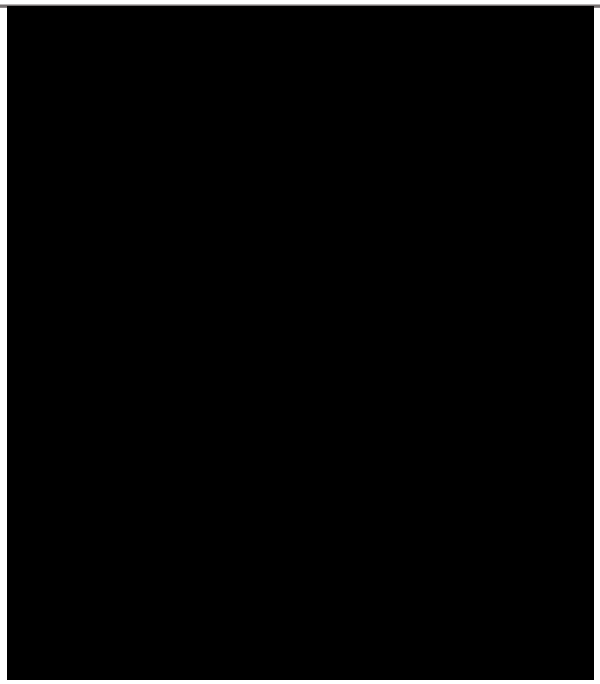


‘Manchester United’ Website Application Project Y3S2

Development/Programming Process

Daniel Wilkins

Contents

Introduction

During the development of the website application regarding Manchester United player statistics/profiles, I explored numerous options to achieve the outcome I wanted to produce. To begin, I thought utilising 'React JS' would be beneficial in helping myself to learn a 'front-end' programming framework as well as the fact that this was a very valued framework within the industry. This therefore influenced the fact that I would need to find another framework relating to 'back-end' website development in order to gather player statistics and information. As will be seen in this document, I explored several options for this but after struggling to find and install a relevant framework and after having discussed with a fellow class colleague, I then decided to utilise the 'PHP' framework called 'Laravel'. This was because this allowed for both 'front-end', through 'blade' files, and 'back-end', meaning there was a higher compatibility and also something that would make it easier for myself trying to learn. The whole process of the development of the Manchester United website application can be viewed in this document including challenges and problems overcome.

Initial Stages of this Process

'React JS' Stage

Installation

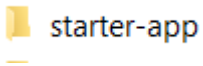
As stated before, I identified 'React JS' as the 'front-end' framework to utilise at first. First of all, I needed to install 'React JS' onto my laptop through different commands on the command line, first of all installing this globally in order to create an actual application within a preferred folder destination:

Installing React JS Globally

```
>npm install -g create-react-app
```

Installing the Application Folder in the Preferred Destination and the Outcome (Without '<>')

```
>create-react-app <starter-app>
```

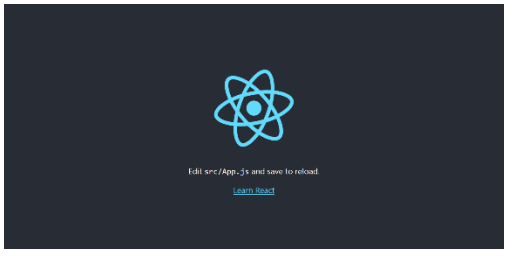


After the installation had been completed, I then needed to execute the following command to begin a server for the application. Once this was complete, this then loaded the application as will be seen below:

Starting the Server for the Application

```
>npm start
```

The Outcome – The Application was now Running Successfully on the Server



Experimenting with 'React JS' with Some Experienced Issues

To begin the whole process of building the application utilising 'React JS', I followed a tutorial of installing an application already created by 'React JS', as shown previously. This would then allow for changing elements, helping myself to learn the programming language quicker. Before I completed this task, I viewed tutorials on 'Codecademy' to help introduce myself to the main concepts of 'React JS' as this was a new language for myself to learn.

After initially changing some elements to reflect my project such as including a relevant title, I then attempted to place 'Font Awesome' icons into the application utilising 'import', as seen on an online resource:

Attempting to Place 'Font Awesome' Icons into the Application through 'import'

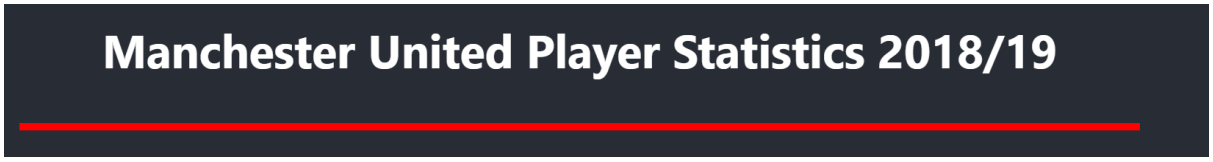
```
import { faHome } from "@fontawesome/free-solid-svg-icons";
import { FontAwesomeIcon } from "@fontawesome/react-fontawesome";
```

Evoking the 'Font Awesome' Icon within the Main Section

```
class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1>
            Manchester United Player Statistics 2018/19
          </h1>
          <FontAwesomeIcon icon={faHome} />
          <hr />
        </header>
        <main>
          <input type="text" placeholder="Search.." name="search" />
          <button type="submit">Submit</button>
        </main>
        <footer className="footer">
          <p>This is the footer section!!!</p>
        </footer>
      </div>
    );
  }
}
```

This, however, didn't successfully work as will be seen below with an example captured at a later stage:

The Outcome of this on the Web Page – This Didn't Successfully Work as the Icon was Expected to Appear between the Title and 'hr' Element



After realising that the previous aspect didn't successfully work and after attempting many different solutions, I then discovered a solution through research undertaken on the Internet. This was by doing something similar to before, however '@fontawesome' was changed to '@fortawesome'. This then worked as will be seen below. Please also note that I needed to install the necessary files from the 'Font Awesome' website in order to import icons as will also be seen below:

Installing the Necessary Files for 'Font Awesome' in the Command Line Interface within the Application Folder

```
npm i --save @fortawesome/fontawesome-svg-core
```

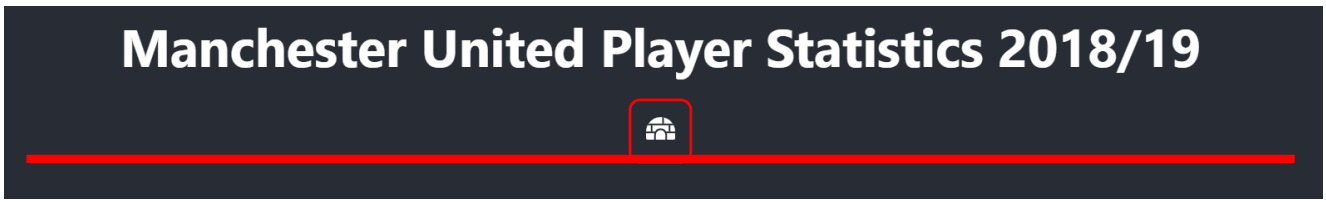
```
npm i --save @fortawesome/free-solid-svg-icons
```

```
npm i --save @fortawesome/react-fontawesome
```

Integrating the Highlighted Aspects of Code into the Application (Using Igloo for Testing Purposes)

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
import { faIgloo } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1>
            Manchester United Player Statistics 2018/19
          </h1>
          <FontAwesomeIcon icon={faIgloo} />
          <hr />
        </header>
        <main>
          <input type="text" placeholder="Search.." name="search" />
          <button type="submit">Submit</button>
        </main>
        <footer className="footer">
          <p>This is the footer section!!!</p>
        </footer>
      </div>
    );
  }
}

export default App;
```

The Outcome of this on the Web Page – This now Worked Successfully



Following on from the previous issue, whilst beginning to structure and style the application properly, I encountered the following problem where the filter sections would overflow past the border/container that they were placed within:

The Current Code for the Structuring and Styling

```
        <main className="App-main">
        <br />
        <div className="user_options_container">
        <div className="search_container">
          <input type="text" placeholder="Search.." name="search" />
          <button type="submit"><FontAwesomeIcon icon={faSearch} /></button>
        </div>
        <div className="filter_container1">
          <select>
            <option value="option1">Option 1</option>
            <option value="option2">Option 2</option>
            <option value="option3">Option 3</option>
            <option value="option4">Option 4</option>
          </select>
        </div>
        <div className="filter_container2">
          <select>
            <option value="option1">Option 1</option>
            <option value="option2">Option 2</option>
            <option value="option3">Option 3</option>
            <option value="option4">Option 4</option>
          </select>
        </div>
        <div className="filter_container3">
          <select>
            <option value="option1">Option 1</option>
            <option value="option2">Option 2</option>
            <option value="option3">Option 3</option>
            <option value="option4">Option 4</option>
          </select>
        </div>
        <div className="filter_container4">
          <select>
            <option value="option1">Option 1</option>
            <option value="option2">Option 2</option>
            <option value="option3">Option 3</option>
            <option value="option4">Option 4</option>
          </select>
        </div>
        </div>
        </main>
        <footer className="footer">
          <p>This is the footer section!!!</p>
        </footer>
      </div>
    );
  }
```

```
.App-main {
  display: flex;
  flex-direction: row;
  width: 100%;
  height: 500px;
  background-color: yellow;
}

.user_options_container {
  display: flex;
  flex-direction: column;
  width: 50%;
  height: auto;
  border-right: 2px solid red;
}

.search_container {
  display: block;
  margin: auto;
  width: 100%;
  height: auto;
  background: black;
  padding: 20px;
}

.filter_container1 {
  display: block;
  margin: auto;
  width: 100%;
  height: auto;
  background: black;
  padding: 20px;
}

.filter_container2 {
  display: block;
  margin: auto;
  width: 100%;
  height: auto;
  background: white;
  padding: 20px;
}
```

```
.filter_container3 {
  display: block;
  margin: auto;
  width: 100%;
  height: auto;
  background: black;
  padding: 20px;
}

.filter_container4 {
  display: block;
  margin: auto;
  width: 100%;
  height: auto;
  background: white;
  padding: 20px;
}
```

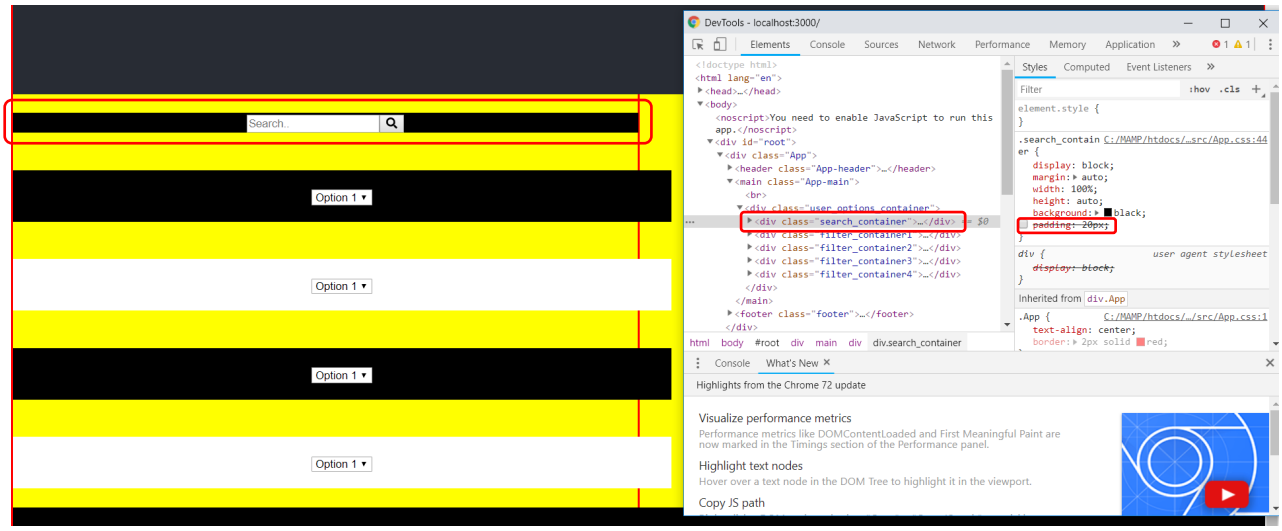
As can be seen above, at this stage the parent container was called 'App-main' where a 'flex-direction' of 'row' was assigned to display containers inside a row format. Within this was a container called 'user_options_container' which was where the search and filters were planning to

be placed with another section planning to be added later regarding displaying the actual players. This is the reason why this container was assigned a width of 50%. This container had 'flex-direction: column' assigned to display all content inside in a vertical format with the other containers relating to the custom search and filters. For each of these containers a width of 100% had been assigned to ensure these aspects fully filled their parent container with alternating background colours being utilised to distinguish different filters. Furthermore 'display: block' and 'margin: auto' were assigned to allow elements inside to be placed centrally with 'padding' being applied to space the elements inside further away from the edges of the containers they were placed within. Please note that online resources were utilised to integrate the custom search and filters. The problem at this stage can be viewed below:

The Problem on the Web Page – The Containers were Overflowing

After utilising the 'DevTools' on 'Google Chrome' to help resolve the issue, I then realised that the 'padding' assigned to each filter container was causing the issue as displayed below:

Highlighting the Issue with 'DevTools' on 'Google Chrome' by Removing the 'padding'



As a result of this, I therefore removed the 'padding' from each of the containers and this resolved the issue as seen below:

The Outcome After Removing the 'padding'

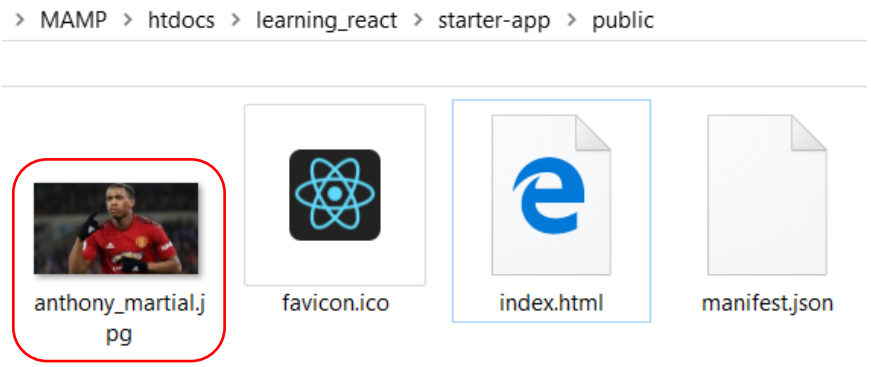
After continuing further in 'React JS', I encountered an issue with importing an image into the page as a placeholder for the current time. This can be viewed below:

The Current Code

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
import { faIgloo } from '@fortawesome/free-solid-svg-icons';
import { faSearch } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import anthony_martial from './anthony_martial.jpg';
```

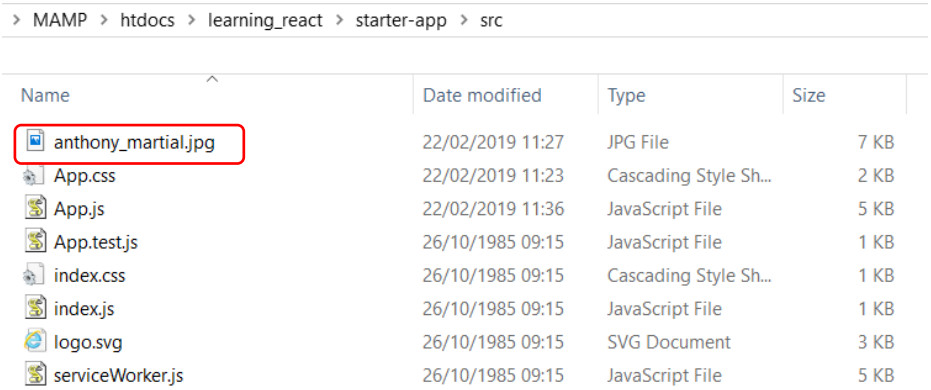
```
<div className="player_container">
  <div className="player_container_title">
    <h2>{"Player Profile"}</h2>
  </div>
  <div className="player_container_content">
    <img src={anthony_martial} alt="anthony_martial" />
  </div>
</div>
```

The Destination of the Image – This was Placed in the 'public' Folder



After then reading further through resources online, I had noticed that I needed to have the image in the same folder for the code to work successfully. The image was therefore moved and the image now appeared when reloading the page:

Placing the Image into the Same Folder as the Other Code Files



The Outcome on the Page – The Image now Successfully Appeared

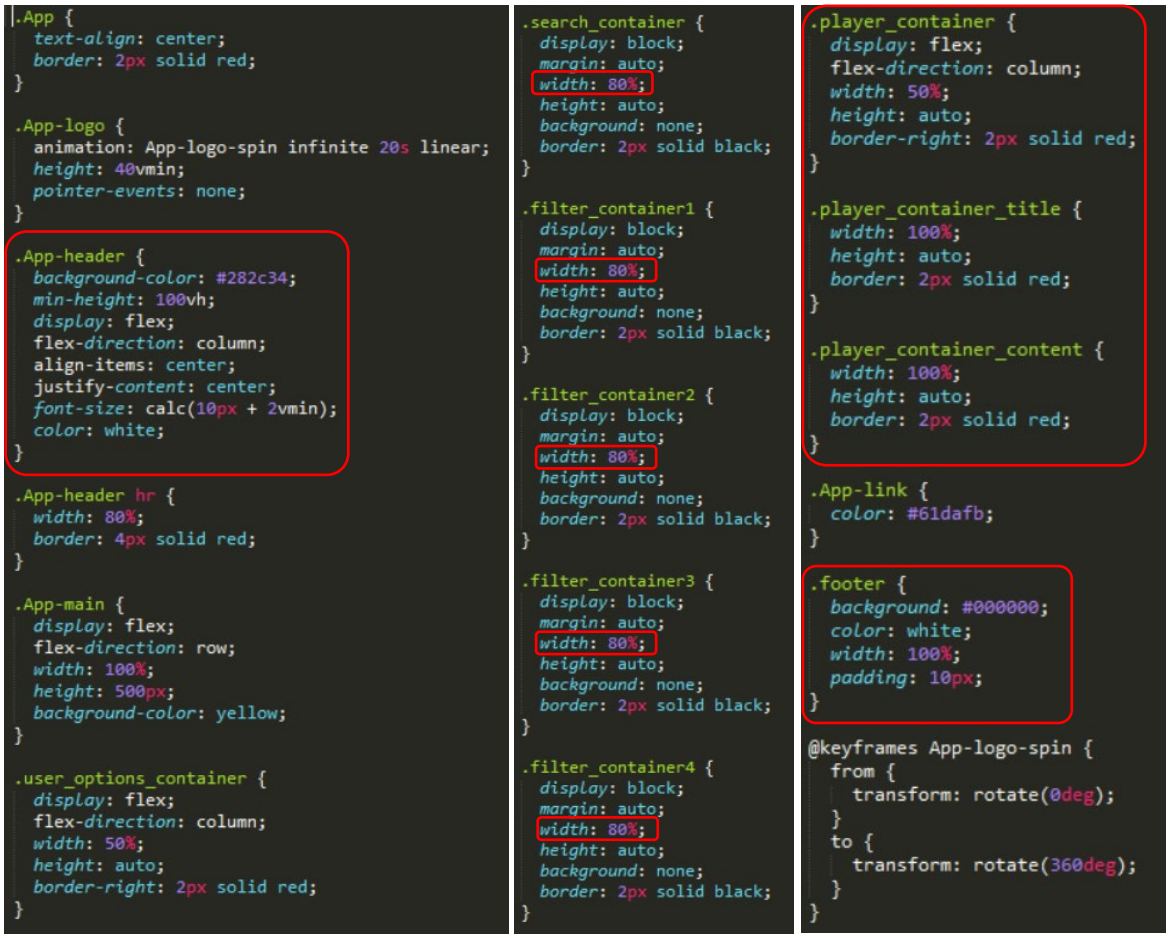
After having solved the issue shown before, I then decided to leave the interface as this was at the current moment in order to allow myself to focus on how I would gather data to insert into the website application. The current code at this stage for the interface can be viewed below:

The Current Structuring and Styling Code for the Interface

App.js file (Horizontally Ordered)

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
import { faIgloo } from '@fortawesome/free-solid-svg-icons';
import { faSearch } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import anthony_martial from './anthony_martial.jpg';


class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1>
            Manchester United Player Statistics 2018/19
          </h1>
          <FontAwesomeIcon icon={faIgloo} />
          <hr />
        </header>
        <main className="App-main">
        <br />
        <div className="user_options_container">
          <div className="search_container">
            <h3>{"Custom Search"}</h3>
            <input type="text" placeholder="Search.." name="search" />
            <button type="submit"><FontAwesomeIcon icon={faSearch} /></button>
          </div>
          <div className="filter_container1">
            <h3>{"Player's Age"}</h3>
            <select>
              <option value="option1">Option 1</option>
              <option value="option2">Option 2</option>
              <option value="option3">Option 3</option>
              <option value="option4">Option 4</option>
            </select>
          </div>
          <div className="filter_container2">
            <h3>{"Player's Nationality"}</h3>
            <select>
              <option value="option1">Option 1</option>
              <option value="option2">Option 2</option>
              <option value="option3">Option 3</option>
              <option value="option4">Option 4</option>
            </select>
          </div>
          <div className="filter_container3">
            <h3>{"Player's Position"}</h3>
            <select>
              <option value="option1">Option 1</option>
              <option value="option2">Option 2</option>
              <option value="option3">Option 3</option>
              <option value="option4">Option 4</option>
            </select>
          </div>
          <div className="filter_container4">
            <h3>{"Weekly Statistics/Full Season Statistics"}</h3>
            <select>
              <option value="option1">Option 1</option>
              <option value="option2">Option 2</option>
              <option value="option3">Option 3</option>
              <option value="option4">Option 4</option>
            </select>
          </div>
        </div>
        <div className="player_container">
          <div className="player_container_title">
            <h2>{"Player Profile"}</h2>
          </div>
          <div className="player_container_content">
            <img src={anthony_martial} alt="anthony_martial" />
          </div>
        </div>
        </main>
        <footer className="footer">
          <p>This is the footer section!!!</p>
        </footer>
      </div>
    );
  }
}

export default App;
```

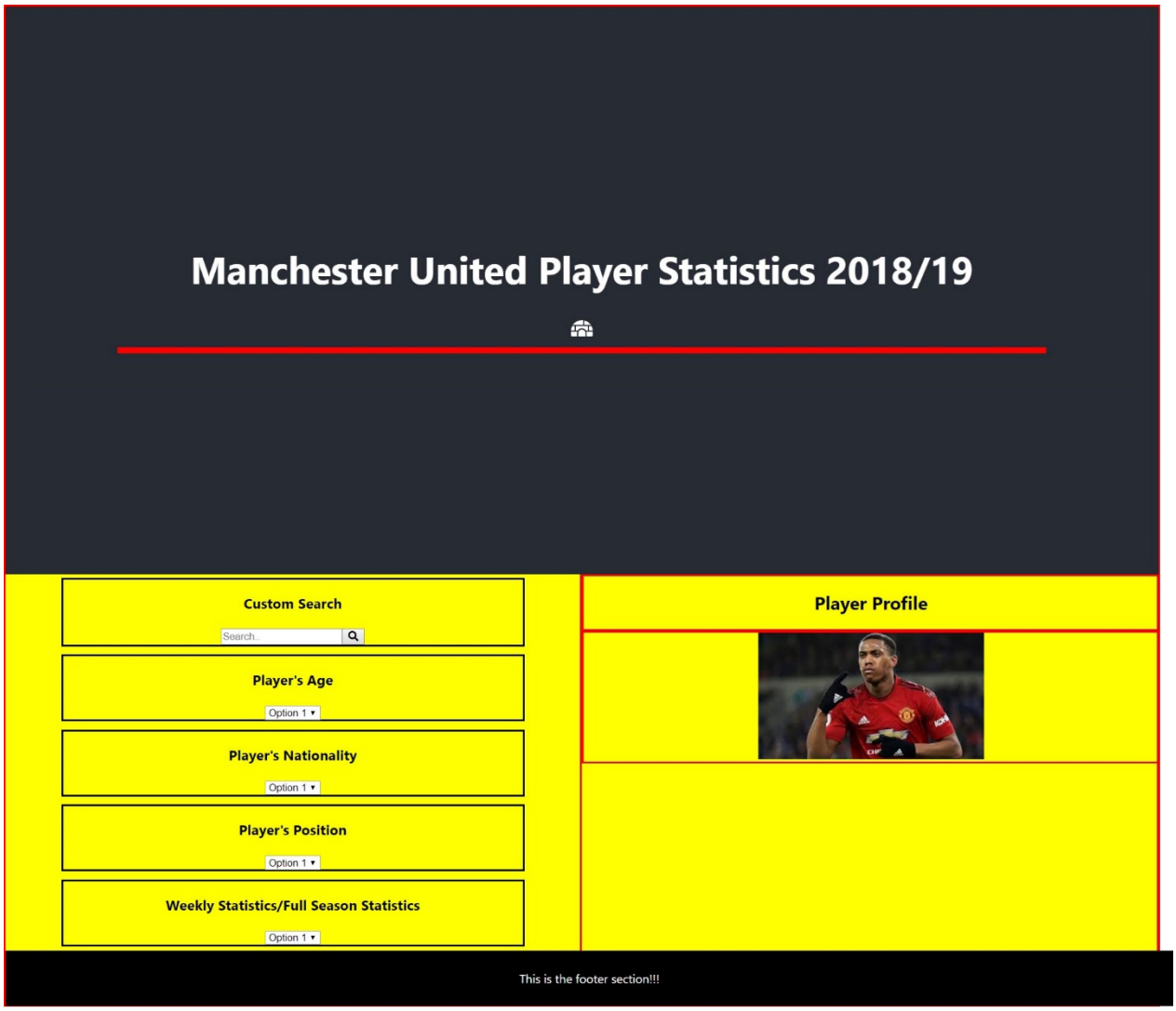
App.css file



```css
.App {
  text-align: center;
  border: 2px solid red;
}

.App-logo {
  animation: App-logo-spin infinite 20s linear;
  height: 40vmin;
  pointer-events: none;
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-header hr {
  width: 80%;
  border: 4px solid red;
}

.App-main {
  display: flex;
  flex-direction: row;
  width: 100%;
  height: 500px;
  background-color: yellow;
}

.user_options_container {
  display: flex;
  flex-direction: column;
  width: 50%;
  height: auto;
  border-right: 2px solid red;
}

.search_container {
  display: block;
  margin: auto;
  width: 80%;
  height: auto;
  background: none;
  border: 2px solid black;
}

.filter_container1 {
  display: block;
  margin: auto;
  width: 80%;
  height: auto;
  background: none;
  border: 2px solid black;
}

.filter_container2 {
  display: block;
  margin: auto;
  width: 80%;
  height: auto;
  background: none;
  border: 2px solid black;
}

.filter_container3 {
  display: block;
  margin: auto;
  width: 80%;
  height: auto;
  background: none;
  border: 2px solid black;
}

.filter_container4 {
  display: block;
  margin: auto;
  width: 80%;
  height: auto;
  background: none;
  border: 2px solid black;
}

.player_container {
  display: flex;
  flex-direction: column;
  width: 50%;
  height: auto;
  border-right: 2px solid red;
}

.player_container_title {
  width: 100%;
  height: auto;
  border: 2px solid red;
}

.player_container_content {
  width: 100%;
  height: auto;
  border: 2px solid red;
}

.App-link {
  color: #61dafb;
}

.footer {
  background: #000000;
  color: white;
  width: 100%;
  padding: 10px;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

As is evident above, some styles relate to what was already included in the application installation. The main aspects to note here are the fact that I changed the widths of each container included in the 'user_options_container' to allow these aspects to be centred on the page through the use of 'display: block' and 'margin: auto'. I had also integrated a container regarding the player's section called 'player_container' which was assigned 'flex-direction: column' to display all items inside in a vertical format. Inside this container, I created containers for the title and actual content, assigning widths of 100% to each to help the content inside fit the whole width of the parent container. Furthermore, I had altered some styles for both the header and footer sections, adding a 'hr' tag for the header and changing colours within the footer. The outcome of the interface at this stage can be viewed on the following page.

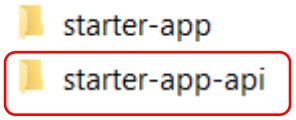
The Outcome on the Web Page



Initial Stages of Attempting to Understand how to Insert Data into the Page

As APIs was an area mentioned by my lecturer and other class colleagues, I therefore undertook research regarding this, creating a separate application to the one shown above to test with exercises online. Examples of these experimentations can be viewed below:

Duplicating the Existing Project Folder to Test with API Aspects



Examples of Code Snippets Used to Experiment with (Ordered by Numbers)

1

```
class List extends Component {
  state = {}

  componentDidMount () {
    fetch('/manifest.json')
      .then(res => res.json())
      .then(this.onLoad);
  }

  parseData (response) {
    return response.data;
  }

  onLoad = (data) => {
    this.setState({
      data: this.parseData(data)
    });
  }

  render () {
    const { data } = this.state;

    return data ?
      this.renderData(data) :
      this.renderLoading()
  }

  renderData (data) {
    if (data && data.length) {
      return (
        <div>
          {
            data.map(item => (
              <div key={item.id}>
                <a href={`mailto:${item.email}`}>{item.name}</a> {item.company}
              </div>
            ))
          }
        </div>
      );
    } else {
      return <div>No items found</div>
    }
  }

  renderLoading () {
    return <div>Loading...</div>
  }
}

export default List;
```

2

```
const Post = ({ body }) => {
  return (
    <div>
      {body.map(post => {
        const { _id, title, content } = post;
        return (
          <div key={_id}>
            <h2>{title}</h2>
            <p>{content}</p>
            <hr />
          </div>
        );
      })}
    </div>
  );
};

class App extends React.Component {
  state = {
    isLoading: true,
    posts: [],
    error: null,
  };
getUsers() {
  // We're using axios instead of Fetch
  axios
    // The API we're requesting data from
    .get("https://randomuser.me/api/?results=5")
    // Once we get a response, we'll map the API endpoints to our props
    .then(response =>
      response.data.results.map(user => ({
        name: `${user.name.first} ${user.name.last}`,
        username: `${user.login.username}`,
        email: `${user.email}`,
        image: `${user.picture.thumbnail}`
      }))
    )
    // Let's make sure to change the loading state to display the data
    .then(users => {
      this.setState({
        users,
        isLoading: false
      });
    })
    // We can still use the `.catch()` method since axios is promise-based
    .catch(error => this.setState({ error, isLoading: false }));
}
  }

  componentDidMount() {
    this.getUsers();
  }
```

3

```
  render() {
    const { isLoading, posts, error } = this.state;
    return (
      <React.Fragment>
        <h1>React Fetch - Blog</h1>
        <hr />
        {!isLoading ? Object.keys(posts).map(key => <Post key={key} body={posts[key]} />) : <h3>Loading...</h3>}
      </React.Fragment>
    );
  }
}


ReactDOM.render(<App />, document.getElementById("root"));
```

From undertaking further reading of this subject area, I then understood that I needed to create my own API in order to relate to the application I would be making. However, after reading of areas regarding this, I was unsure of how to do this. I therefore sought advice from other class colleagues within my group and understood that connecting 'React JS' to a database would be a better option where I would be able to enter data into a database and gather the data in this method. Therefore, I began to undertake research regarding frameworks/technologies to utilise in order to achieve this. From this research, I understood that 'Node JS' would need to be used with 'MySQL' if I were to do this through a more traditional method of which was the method that appealed to myself the most at this stage.

Further Exploration of Technologies to Utilise

After having discussed with the lecturer with regards to which technologies I should utilise after also explaining what I had discovered, it was suggested that one of the aspects I should research around was web APIs. Therefore, I did this and in doing this I highlighted a few options. The first option discovered was a platform from 'GitHub' called 'API Platform' and the second option was a framework called 'Django'.

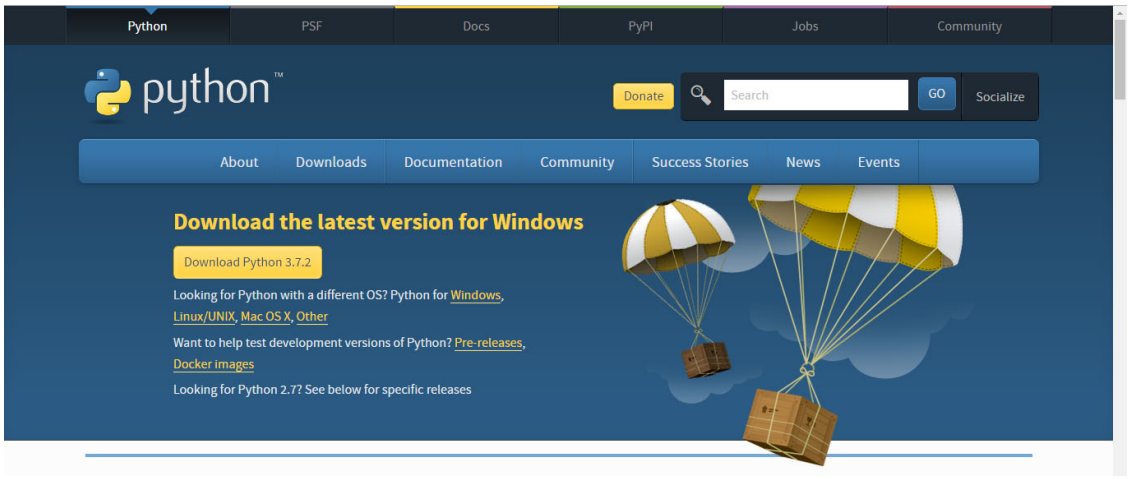
Experimenting with Different Technologies/Platforms

'Django' Framework

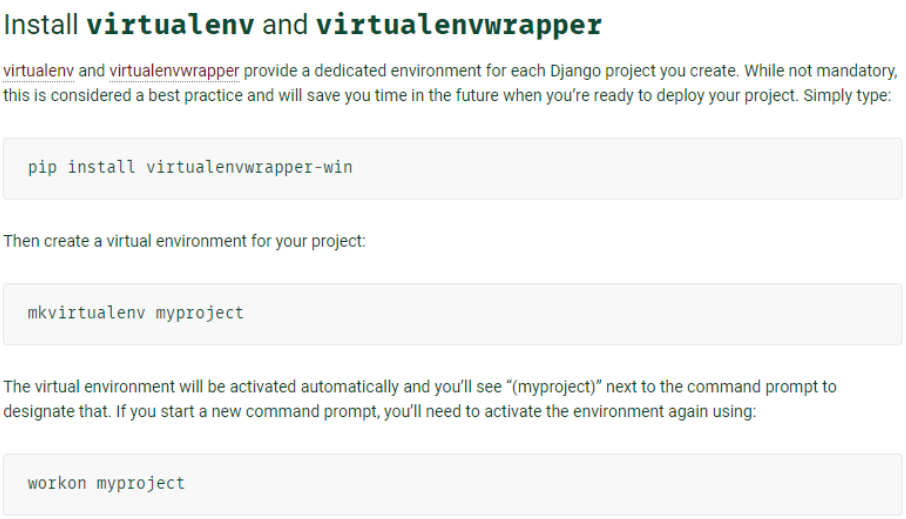
The reason for previously highlighting 'Django' was because, from research, I understood that 'Django' could be utilised with 'React JS' as well as this being a popular choice. This was therefore something I then decided to experiment with as this was more widely used than 'API Platform'. The purpose of using 'Django' would be to create an 'API' to then be able to create the data and integrate this into the 'React JS' application at a later date.

After following some of the installation steps for 'Django', I then became stuck and I wasn't completely sure of what I undertaking as this related to 'Python', a language I had no previous knowledge of. The process of where I was at this point can be seen below:

Installing Python from the Python Website



Installing 'virtualenv' and 'virtualenvwrapper' to Create a Virtual Environment



Install **virtualenv** and **virtualenvwrapper**

virtualenv and virtualenvwrapper provide a dedicated environment for each Django project you create. While not mandatory, this is considered a best practice and will save you time in the future when you're ready to deploy your project. Simply type:

```
pip install virtualenvwrapper-win
```

Then create a virtual environment for your project:

```
mkvirtualenv myproject
```

The virtual environment will be activated automatically and you'll see "(myproject)" next to the command prompt to designate that. If you start a new command prompt, you'll need to activate the environment again using:

```
workon myproject
```

Installing the Framework 'Django' through the use of 'pip'

Install Django

Django can be installed easily using **pip** within your virtual environment.

In the command prompt, ensure your virtual environment is active, and execute the following command:

```
pip install django
```

This will download and install the latest Django release.

After the installation has completed, you can verify your Django installation by executing **django-admin --version** in the command prompt.

See Get your database running for information on database installation with Django.

The Final Aspect where I had then Become Stuck

Get your database running

If you plan to use Django's database API functionality, you'll need to make sure a database server is running. Django supports many different database servers and is officially supported with PostgreSQL, MySQL, Oracle and SQLite.

If you are developing a simple project or something you don't plan to deploy in a production environment, SQLite is generally the simplest option as it doesn't require running a separate server. However, SQLite has many differences from other databases, so if you are working on something substantial, it's recommended to develop with the same database that you plan on using in production.

In addition to the officially supported databases, there are backends provided by 3rd parties that allow you to use other databases with Django.

In addition to a database backend, you'll need to make sure your Python database bindings are installed.

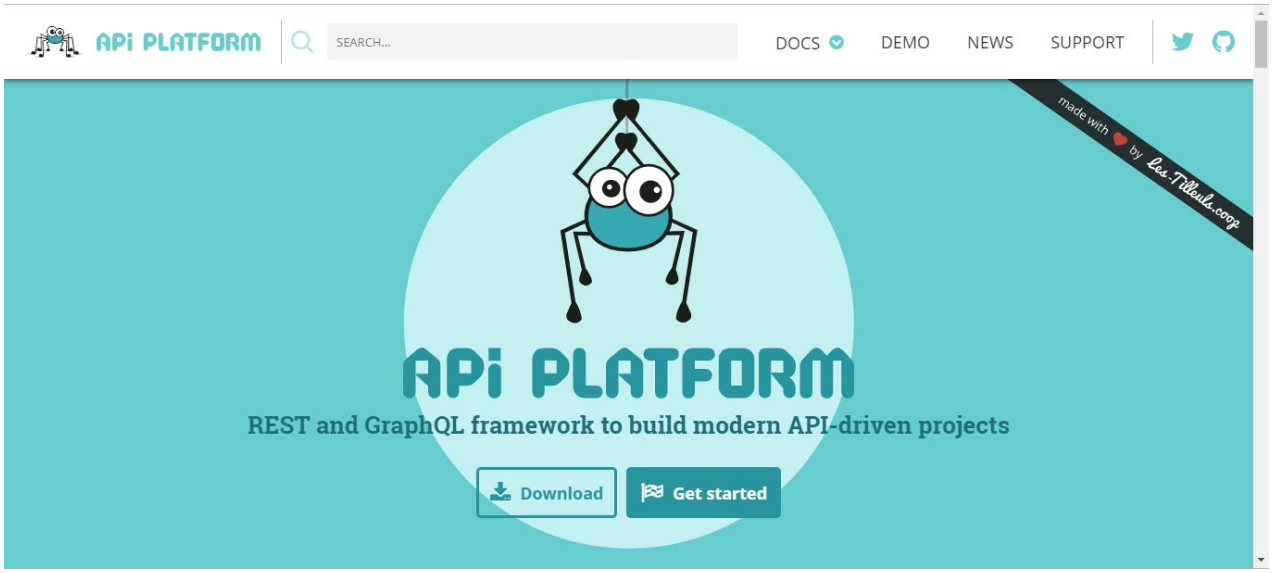
- If you're using PostgreSQL, you'll need the psycopg2 package. Refer to the PostgreSQL notes for further details.
- If you're using MySQL, you'll need a DB API driver like **mysqlclient**. See notes for the MySQL backend for details.
- If you're using SQLite you might want to read the SQLite backend notes.
- If you're using Oracle, you'll need a copy of cx_Oracle, but please read the notes for the Oracle backend for details regarding supported versions of both Oracle and **cx_Oracle**.
- If you're using an unofficial 3rd party backend, please consult the documentation provided for any additional requirements.

As I had now become unsure of what to do and due to the fact that I wanted to progress with the project, this therefore meant that I decided to choose another option instead. This was also due to the fact that 'Python' would be extremely difficult to learn whilst undertaking other projects.

'API Platform'

After the previous issue, I then decided to explore the 'API Platform' from 'GitHub' instead as this looked like an option which would be easier to learn and integrate into the project. However, due to the fact that I would be experimenting with these technologies on my old laptop, this meant I was unable to experiment with this platform. This was because I realised I was required to have 'Windows 10' for when installing something called 'Docker' to use with the platform. My old laptop's Operating System was 'Windows 8'. This process can be seen below:

The API Platform Website



Realising I Needed to Install 'Docker'

Installing the Framework

Using the Official Distribution (recommended)

Start by **downloading the API Platform distribution** and extract its content. The resulting directory contains an empty API Platform project structure. You will add your own code and configuration inside it.

API Platform is shipped with a **Docker** setup that makes it easy to get a containerized development environment up and running. If you do not already have Docker on your computer, **it's the right time to install it**.

On Mac, only **Docker for Mac** is supported. Similarly, on Windows, only **Docker for Windows** is supported. Docker Machine **is not** supported out of the box.

Open a terminal, and navigate to the directory containing your project skeleton. Run the following command to start all services using **Docker Compose**:

```
$ docker-compose pull # Download the latest versions of the pre-built images
$ docker-compose up -d # Running in detached mode
```

Visiting the 'Docker' Website to Install 'Docker' for Windows

Get started with Docker for Windows

Estimated reading time: 17 minutes

Welcome to Docker Desktop for Windows!

Docker is a full development platform for creating containerized apps, and Docker Desktop for Windows is the best way to get started with Docker *on Windows*.

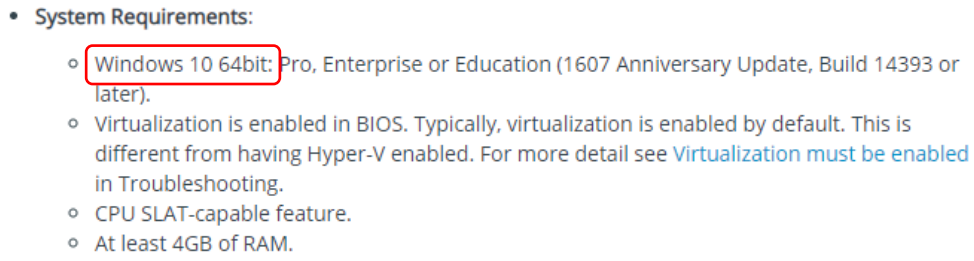
> See Install Docker Desktop for Windows for information on system requirements and stable & edge channels.

Test your installation

1. Open a terminal window (Command Prompt or PowerShell, *but not* PowerShell ISE).
2. Run `docker --version` to ensure that you have a supported version of Docker:

```
> docker --version

Docker version 18.03.0-ce, build 0520e24
```

Realising 'Windows 10' was Required, meaning my Old Laptop Wasn't Compatible



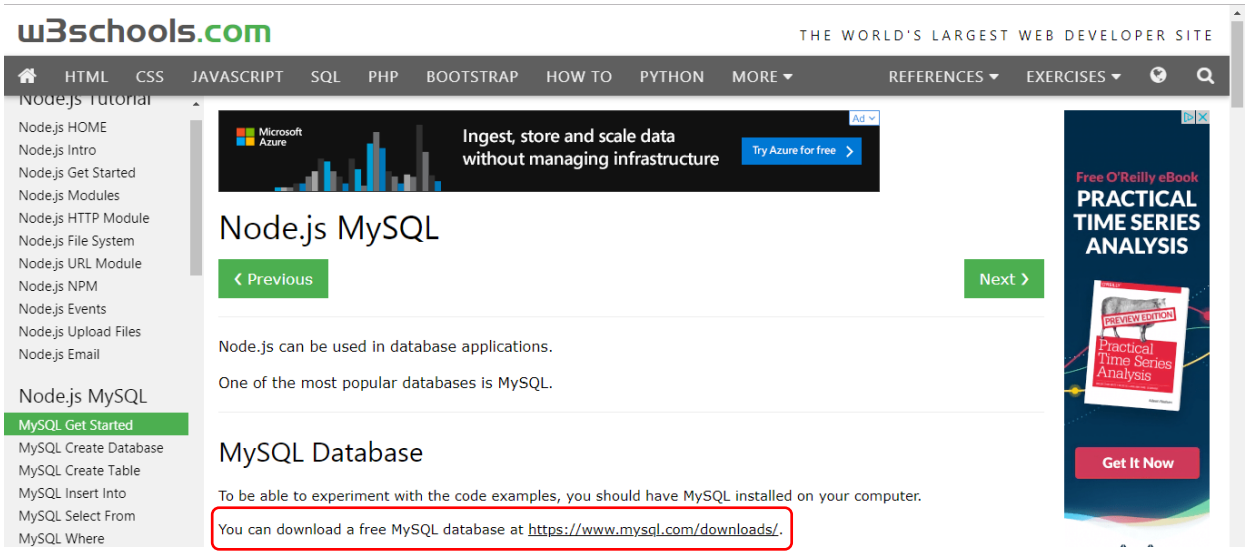
- **System Requirements**:
  - Windows 10 64bit: Pro, Enterprise or Education (1607 Anniversary Update, Build 14393 or later).
  - Virtualization is enabled in BIOS. Typically, virtualization is enabled by default. This is different from having Hyper-V enabled. For more detail see Virtualization must be enabled in Troubleshooting.
  - CPU SLAT-capable feature.
  - At least 4GB of RAM.

Therefore, I then decided to view another option instead which can be viewed on the following page.

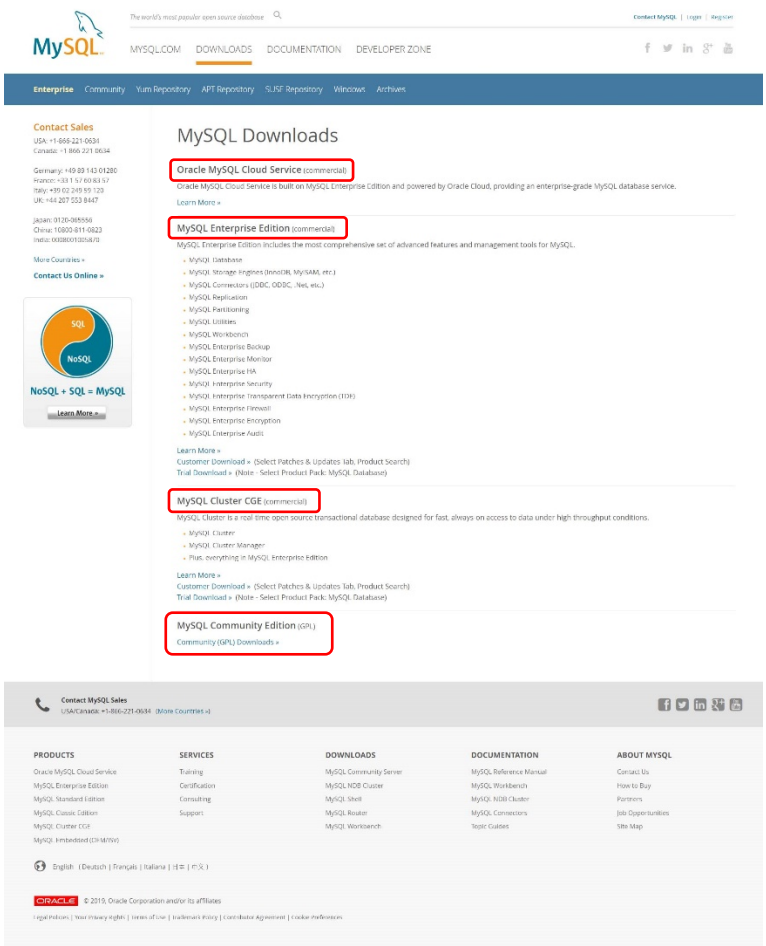
'Node.js' and 'MySQL'

After experiencing the previous issues, I then decided to view the 'MySQL' and 'Node JS' from 'W3Schools' again. I realised I then needed to install a 'MySQL' database which I then visited the provided link. However, I then become confused with this due to the fact that there were multiple options and that only the trial versions were free. This was because I was inexperienced in this area:

Viewing the Instructions on the 'W3Schools' Website for Node JS and MySQL



Selecting the Provided Link and Becoming Stuck with the Available Options



'Laravel'

After again struggling to find and install a framework/technology correctly, I then asked a fellow class colleague what they were using for their 'API' project, as I knew they were undertaking a similar project to mine. I was then informed that they were using 'Laravel' of which I then viewed and this was the framework which then replaced 'React JS' as this had capabilities of undertaking both 'front-end' and 'back-end' tasks.

Utilising 'Laravel'

Installation with Issues

As I had now identified a technology/framework which could be utilised for the project, I then first installed 'MAMP' to allow for 'Laravel' to be able connect to a local server in order to develop the project. I also installed 'composer' in order to be able to install 'Laravel'. To set up the 'composer' file, I selected the 'PHP' command line as that installed in 'MAMP'. After this, I then installed 'Laravel' globally through the following command:

Installing Laravel Globally

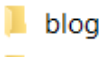
```
composer global require laravel/installer
```

Installing 'Laravel' globally then allowed myself to create a compiled 'Laravel' project through the following command in the command line interface:

Creating a Compiled Project through the Command Line Interface

```
laravel new blog
```

The Outcome – This Created a new Project Folder



After undertaking the previous steps, I then attempted to run the project through the following command within the 'blog' folder:

Attempting to Run the Project

```
php artisan serve
```

However, this didn't work successfully as it was stated that there was a file missing. After undertaking research, I then typed 'composer update' in the folder and this then allowed for the 'php artisan serve' to be executed. This process can be viewed below:

Entering 'composer update' into the folder within the Command Line Interface

```
composer update
```

Attempting to Run the Project and the Outcome

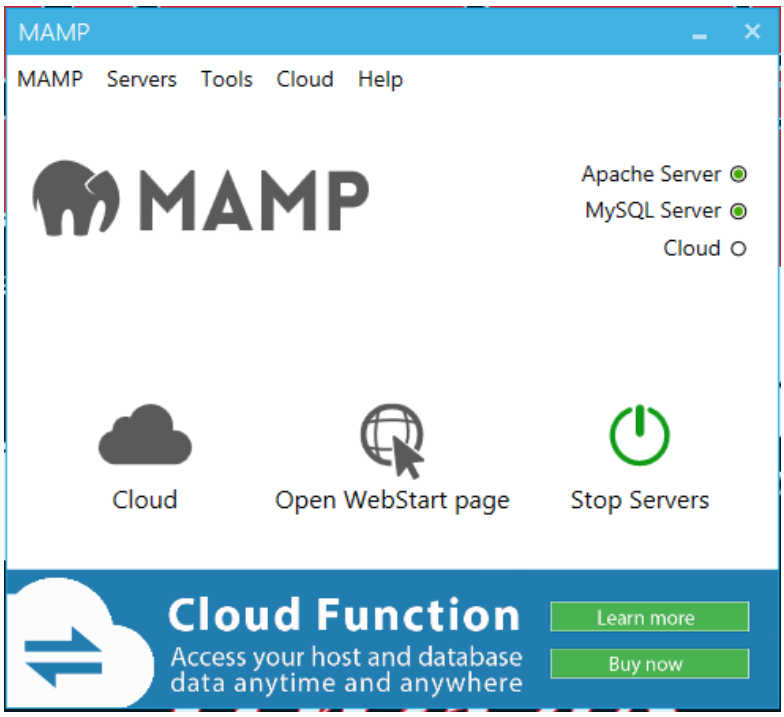
```
php artisan serve
```

500 | Server Error

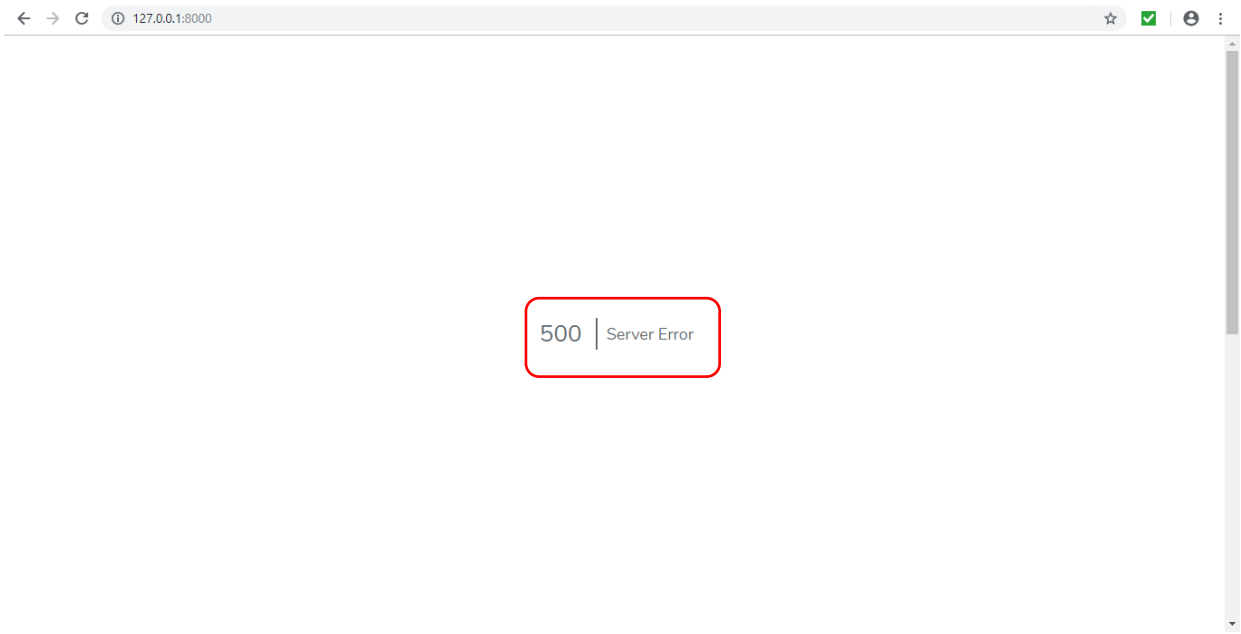
As can be seen above, at this stage 'php artisan serve' was executing correctly as an error appeared in the style of 'Laravel'. However, I now encountered an error.

After seeking advice from the lecturer, I needed to then reinstall 'MAMP' as this couldn't connect to the 'MySQL' server. Therefore, I did this and this now worked with regards to connecting to the 'MySQL' server as will be seen below:

MAMP was now Connecting to the MySQL Server

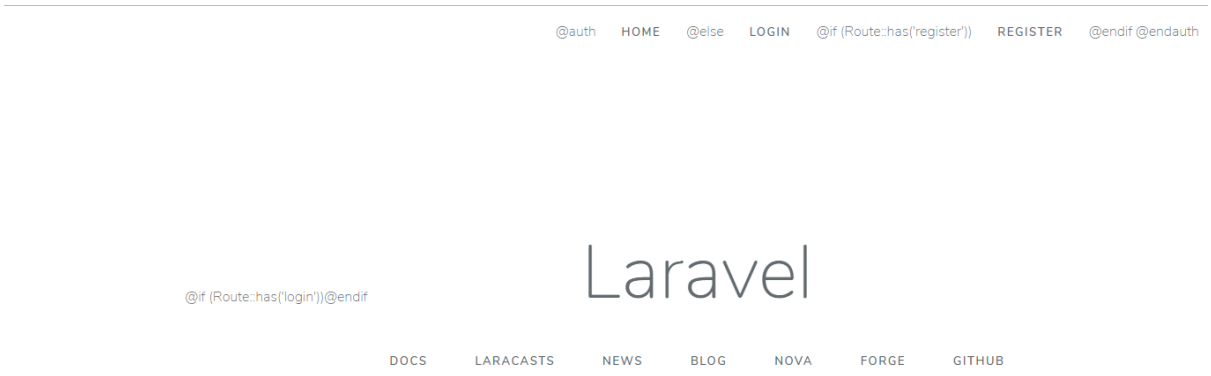


After being informed that the lecturer would help via 'TeamViewer', I then decided to reset the 'composer' file to locate to the 'PHP' command line of the one installed from the 'php.net' website to understand if I could then install 'Laravel' properly. After doing this, I then created a new project called 'blog' through 'laravel new blog', as displayed before, and then entered 'php artisan serve' which caused the following error to appear again. This error can be viewed on the following page.



After encountering this error, I then navigated to the folder on 'MAMP' manually and attempted to see if this would execute properly. However, I now experienced code not executing properly on the page as will be seen below:

The Errors Appearing on the Page



At this stage, I then decided to leave this for the current time to allow the lecturer to help me with later.

Being Introduced to 'Laravel' with Issues

During one of the lectures, I was helped by the other fellow class colleague undertaking a project with 'Laravel', being introduced to areas such as models and migrations. During this process, whilst trying to run the command called 'php artisan migrate' to introduce myself to updating the database with new tables, we encountered an issue with regards to the fact that a 'driver' could not be found. This error can be viewed below:

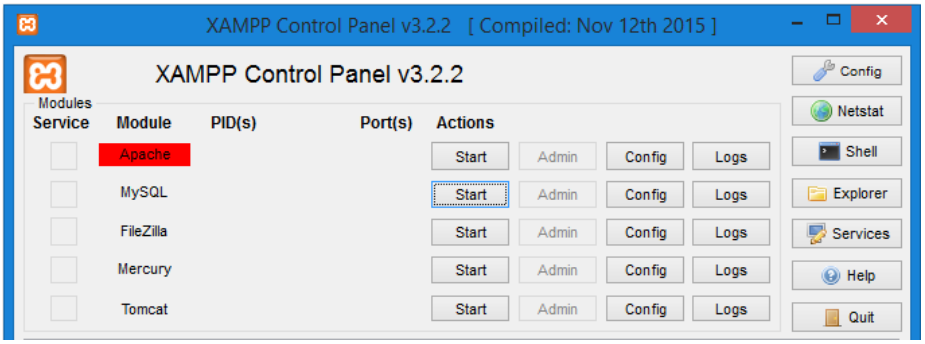
The Error Appearing when Attempting to Execute 'php artisan migrate' in the Project Folder

```
C:\xampp\htdocs\blog>php artisan migrate

   Illuminate\Database\QueryException  : could not find driver (SQL: select * fr
om information_schema.tables where table_schema = mufc_web_app and table_name =
migrations)

  at C:\xampp\htdocs\blog\vendor\laravel\framework\src\Illuminate\Database\Conne
ction.php:664
    660|         // If an exception occurs when attempting to run a query, we'll
 format the error
    661|         // message to include the bindings with SQL, which will make th
is exception a
    662|         // lot more helpful to the developer instead of just the databa
se's errors.
    663|         catch (Exception $e) {
  > 664|             throw new QueryException(
    665|                 $query, $this->prepareBindings($bindings), $e
    666|             );
    667|         }
    668|

  Exception trace:

  1   PDOException::("could not find driver")
      C:\xampp\htdocs\blog\vendor\laravel\framework\src\Illuminate\Database\Conn
ectors\Connector.php:70

  2   PDO::__construct("mysql:host=127.0.0.1;port=3307;dbname=mufc_web_app", "ro
ot", "root", [])
      C:\xampp\htdocs\blog\vendor\laravel\framework\src\Illuminate\Database\Conn
ectors\Connector.php:70

  Please use the argument -v to see more details.
```

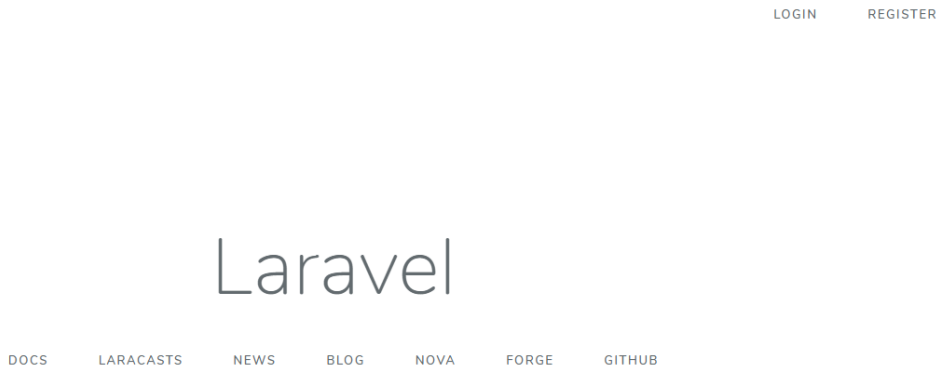
Then when trying to resolve this problem through areas including changing aspects of the 'php.ini' file within 'MAMP', 'MAMP' managed to become unusable as it wouldn't allow a connection to the servers to be stopped. This was still the same after restarting the laptop. Therefore, to resolve this, 'MAMP' was uninstalled and 'XAMPP' was installed instead to attempt to allow 'php artisan migrate' to function properly.

However, the problem with 'MySQL' still remained, not allowing the 'php artisan migrate' to function correctly. Furthermore, 'Apache' was running correctly but this wasn't displaying in the actual interface as shown below:

Apache Not Displaying in the Interface of XAMPP



Although the Application was Running Successfully



After being helped to change settings such as the 'port', this still didn't resolve the issue. Therefore, lecturer advice was sought where we were then informed to view the 'PHP' requirements for 'Laravel' on the official website and to install the 'PDO PHP Extension' as seen below:

Viewing the Requirements for Laravel

Server Requirements

The Laravel framework has a few system requirements. All of these requirements are satisfied by the Laravel Homestead virtual machine, so it's highly recommended that you use Homestead as your local Laravel development environment.

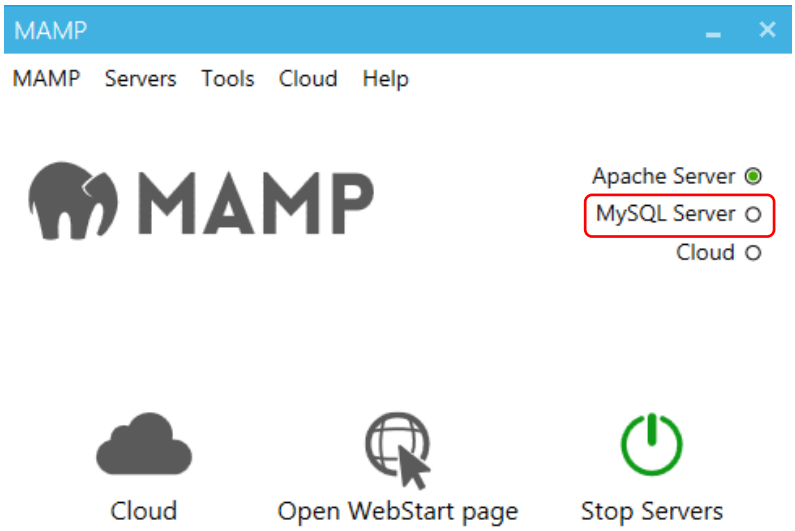
However, if you are not using Homestead, you will need to make sure your server meets the following requirements:

- PHP >= 7.1.3
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- BCMath PHP Extension

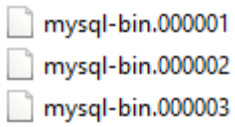
From this, it was understood that the 'PHP' version was correct but that it was definitely something regarding the 'PDO PHP Extension'. I then decided myself to uninstall 'XAMPP' and reinstall 'MAMP' to understand if this would have any effect as I believed that 'php artisan migrate' should have been able to function correctly with any local server platform.

Whilst doing this, I encountered another issue whereby starting and stopping the servers for 'MAMP' and then restarting these again would cause no connection to the 'MySQL' server to be made. This was a problem experienced the first time of using 'MAMP' whilst being helped by the other developer. To resolve this, I was helped to undertake the following process through the assistance of 'Stack Overflow' also. First of all, I was told to navigate to the folder in 'MAMP' where 'mysql-bin' files were held. I was then informed to copy these to another location whilst also deleting these files. After deleting these, I was then told to restart the servers again and ensure that a connection to 'MySQL' could be established. Following on from this, I was then informed I needed to comment out 'log-bin' in the 'my.ini' file and restart the 'MAMP' again, waiting for connections to be established. Finally, I was then told to place the 'mysql-bin' files back into the relevant folder and restart the servers again, checking if a connection could be made again to 'MySQL'. This then did connect successfully and I was now able to continue with resolving the other issue at hand. A visual process of this can be seen below:

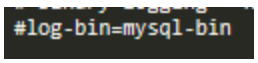
MAMP MySQL Server not Starting After Restarting



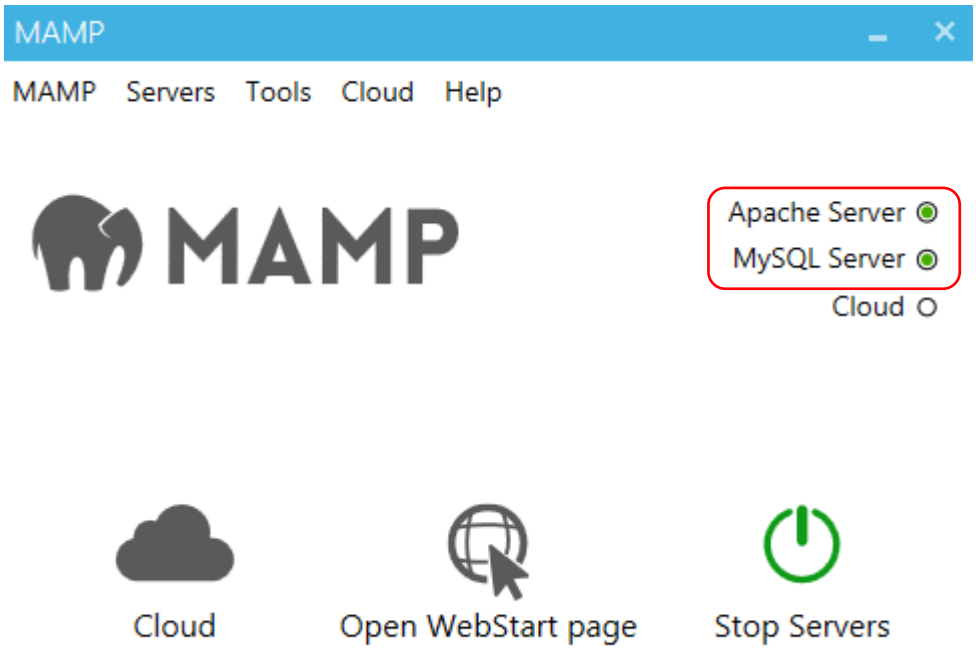
Examples of the 'mysql-bin' files to be Copied with Originals being Deleted



Commenting out 'log-bin' in the 'my.ini' File



The Final Outcome – The MySQL Server was now Connecting



After returning my attention to attempting to allow the 'php artisan migrate' command to work successfully, I initially thought that I could delete the 'PHP' installed from 'php.net'. However, I then thought because the extension was missing regarding 'PDO', that this could be added into the 'php.ini' file being used. I therefore undertook research regarding this, discovering a line of code called 'extension=php_pdo_mysqll.dll'. I believed that adding this would help to resolve the issue and so therefore did this as seen below:

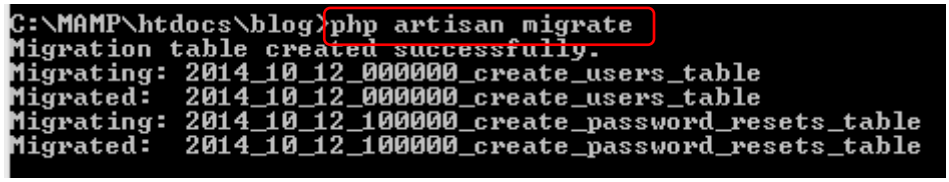
Discovering the Line of Code to add the Extension

```
extension=php_pdo_mysql.dll
```

Adding this to the php.ini file that I had installed from 'php.net'

After doing this, this then resolved the issue which meant I was able to now execute the command 'php artisan migrate', creating tables within the database as displayed below:

Executing the 'php artisan migrate' Successfully



The Tables now Appeared in the Database

Building the Actual Manchester United Website Application

Establishing the Key Initial Areas

After now having been able to set up a framework to utilise regarding data, areas such as 'views' and 'migrations' were then explained to myself by a fellow class colleague.

From these explanations, I then set up another 'Laravel' 'blog' folder but this time, this would be the actual project to be worked on. This process can be seen below:

Setting up the New Folder through the Highlighted Command in the Command Line Interface

```
C:\>cd MAMP/htdocs/mufc_web_app

C:\MAMP\htdocs\mufc_web_app>composer create-project --prefer-dist laravel/larave
l blog
```

This then created the folder in the preferred destination as seen below:

The 'blog' folder had now been Created

To begin, I changed the '.env' file to match the details of the database/connection that I would be utilising. This hasn't been displayed below but this was one of the processes undertaken to allow for data to be passed into the database.

Following on from this, I made and migrated 'auth' through the following command in the command line:

Executing the 'php artisan make' Command

```
C:\MAMP\htdocs\mufc_web_app>php artisan make:auth
```

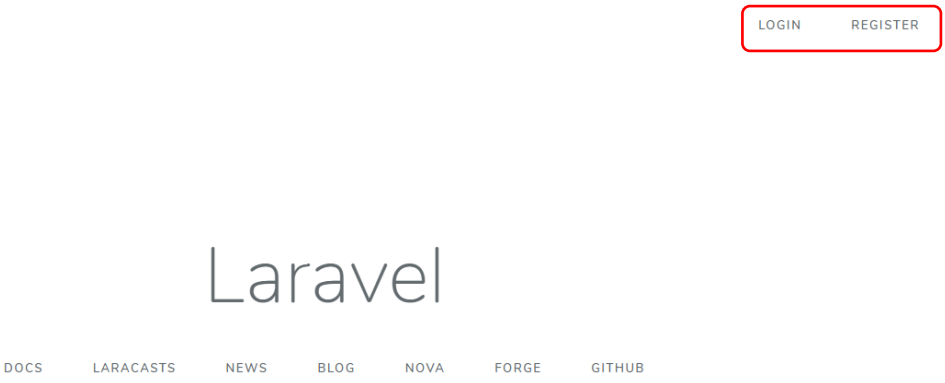
This, however, didn't successfully work but this was because I needed to navigate to the actual 'blog' folder for this to work:

The Original Process and the Process after Navigating to the Correct Folder

```
C:\MAMP\htdocs\mufc_web_app>php artisan make:auth
Could not open input file: artisan

C:\MAMP\htdocs\mufc_web_app>cd blog

C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:auth
```

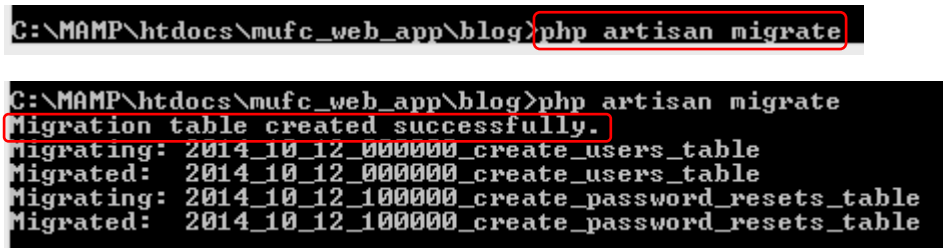
After successfully completing this, the 'login' and 'register' options on the screen then appeared which would allow for registering of users as well as being able to login if required:

The 'login' and 'register' Aspects now Appeared on the Welcome Page



As stated before, I then executed 'php artisan migrate' to update the database which was successful as will be seen below:

Entering the 'php artisan migrate' Command and the Outcome

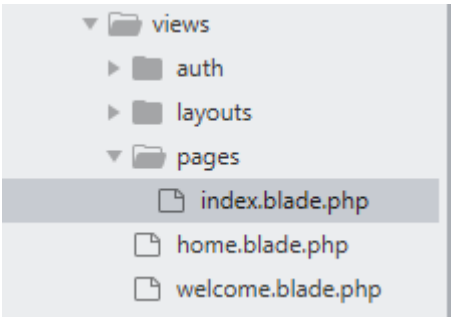


The Tables now Appeared within the Database

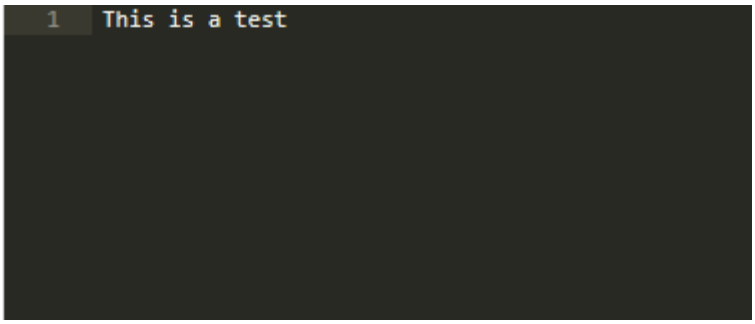
Creating a Basic Structure to Begin

After the previous process, I then decided to create a basic structure of the 'front-end' through aspects such as 'views', 'routes' and 'templates'. In order to create the first 'front-end' aspect of a page, I navigated to the 'views' folder, creating a new folder called 'pages' and within this I created a 'blade' file called 'index.blade.php' which would act as the 'Individual Players Page' for the application. For this stage I entered one line of text to test this at a later stage:

Navigating to the Newly Created 'pages' Folder and 'index.blade.php'



Entering a Line of Text for Testing Purposes at a Later Stage



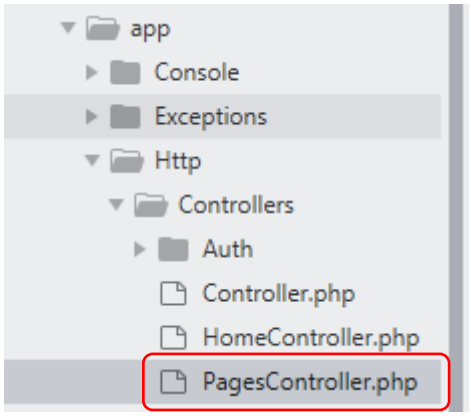
After having undertaken the previous processes, I then changed the 'web.php' file, which was the aspect containing all the 'routes' of the application. I changed the original 'route' by commenting this out in order to include a new 'route' for the newly created 'index.blade.php' file:

Commenting out the Original 'Route' and Creating a new 'Route' for the 'index.blade.php' File

```
/*Route::get('/', function () {
    return view('welcome');
});*/

Route::get('/', 'PagesController@index');
```

The final aspect to add to display the 'index.blade.php' file at this stage was another controller called 'PagesController.php', whereby I integrated the code from another controller and changed this to suit this controller instead by returning the 'view' of 'index.blade.php' instead:

Creating the new 'controller' Called 'PagesController.php'



Integrating the Code from another 'controller' and Changing the Highlighted Aspects

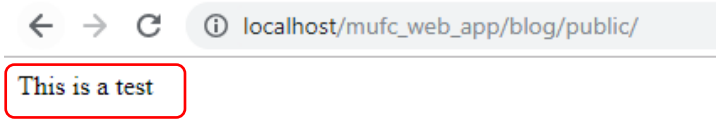
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        return view('home');
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    public function index(){
        return view('pages.index');
    }
}
```

As will be seen above, I needed to change the 'class' aspect to match that of the 'controller' name created as well as changing the 'return view' to include 'pages.index' which acted as a file path with 'pages' being the folder containing the 'index' file. This was also explained previously.

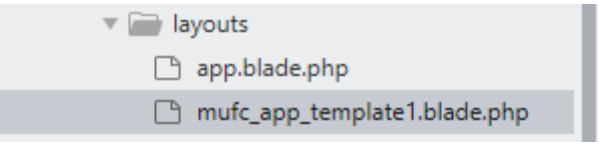
As a result of this, when navigating to the correct 'URL', this then outputted the test message onto the page successfully, meaning this was successfully working:

The 'front-end' Aspect was now Successfully Working



After I had now been able to return the correct 'view' when navigating to the 'public' folder, I then started to create a 'template' for the 'index.blade.php' page to be also applied to any other pages as will be seen below. This needed to be placed within the 'layouts' folder, as previously explained by the other developer. The name of this newly created template was 'mufc_app_template1.blade.php' and the process of this can be viewed below:

Creating the New Template in the 'layouts' Folder



After having created the file, I then added the following code temporarily:

Integrating HTML and Inline CSS Styles as well as Utilising '@yield('content')'

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>MUFC Web App - Home</title>
</head>
<body>
    <div style="width:100%;height:auto;padding:100px;background-color:yellow;border:2px solid red;text-align:center;">
        <h1>Manchester United Website Application</h1>
    </div>
    @yield('content')
    <div style="width:100%;height:auto;padding:10px;background-color:black;color:white;text-align:center;">
        <p>This is the footer section</p>
    </div>
</body>
</html>
```

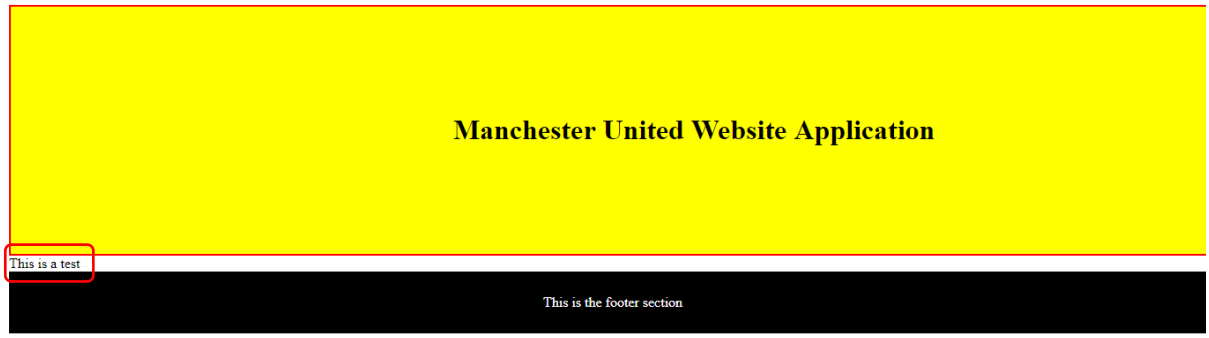
As is evident above, I integrated a very basic structure regarding both header and footer sections due to the fact that I wanted to begin implementing functionality into the page with regards to data. A key aspect to notice is the '@yield('content')' which would indicate where content on each individual page would be placed, allowing unique content to be placed on each page. An example of this will be viewed below where I had utilised '@extends('layout.mufc_app_template1')' to integrate the created template into the 'index.blade.php' file and '@section('content')' to integrate the content for the individual page:

Utilising '@extends' and '@section' in the 'index.blade.php' File

```
@extends('layouts.mufc_app_template1')

@section('content')
This is a test
@endsection
```

The outcome of this at the current stage can be viewed below:

The Template now Applied Successfully with the Unique Entered Content in the 'index.blade.php' File



At this point, I realised that I needed to understand how to implement a stylesheet for the project as using inline styles was unprofessional. Therefore, I undertook research and implemented the following code which then worked. Please note that this was applied to both the 'template' and 'index.blade.php' file, altering the 'h1' style within the stylesheet to test if this worked which it did as will be seen below:

Implementing the Following Code to Include a Stylesheet

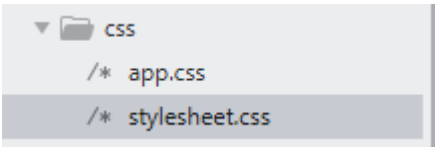
```
<link href="{{ asset('css/stylesheet.css') }}" rel="stylesheet">
```

Altering the 'h1' Style within the Stylesheet

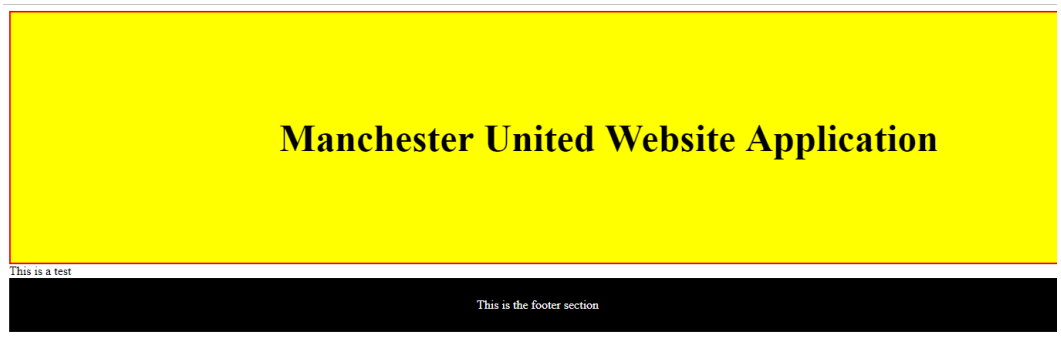
stylesheet.css

```
h1 {
    font-size: 50px;
}
```

Placing the Stylesheet into the 'public' folder Under the Name of 'stylesheet.css'



The Outcome on the Web Page – The Stylesheet was Applying Successfully



After this, I then decided progress with the 'front-end' of the page, creating a very basic page to then start being able to pull data into the page and manipulate this data. This will be evident as this document progresses.

Beginning to Experiment with Manipulating Data on 'Laravel'

In order to understand how to utilise data within a page on 'Laravel', I followed a 'YouTube' tutorial to begin with regards to posts. This process will be able to viewed within this section.

Whilst following the tutorial regarding 'models' and 'database migrations', I first of all created a new 'controller' called 'PostsController' to be able to control the posts individually on the application:

Executing the 'php artisan make:controller' Command in the Command Line Interface

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller PostsController
```

This then Created the 'controller' Successfully

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller PostsController
Controller created successfully.
```

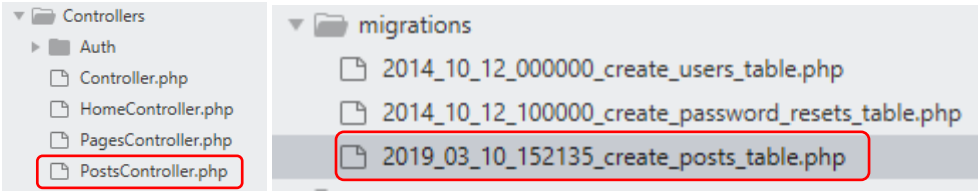
Following on from this, I then created a 'model' through executing the following command which was then successfully created:

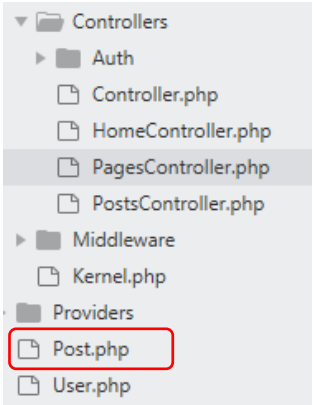
Executing the 'php artisan make:model' Command in the Command Line Interface whilst also Migrating this through '-m'

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:model Post -m
Model created successfully.
Created Migration: 2019_03_10_152135_create_posts_table
```

As these files had now been successfully created, I could now view and edit these within my text editor called 'Sublime Text':

The 'controller', 'migration' and 'model' now Appeared in the Project Folder





Whilst I continued to follow the tutorial, I then added some more fields with their data type such as 'string' into the newly created 'migration' as shown in the tutorial:

Adding more Fields into the 'migration'

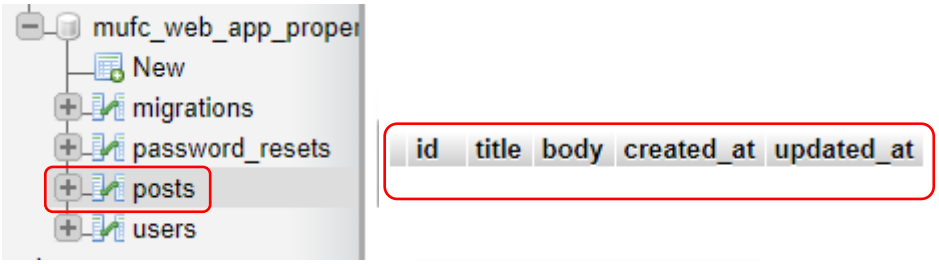
```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('title');
            $table->mediumText('body');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('posts');
    }
}
```

After completing the previous step, I then executed 'php artisan migrate' to sync with the database, adding the table with the fields to this:

Executing 'php artisan migrate' in the Command Line Interface

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan migrate
Migrating: 2019_03_10_152135_create_posts_table
Migrated:  2019_03_10_152135_create_posts_table
```

This then Updated in the Database



Following on from this, I was then shown how to add data to the database through 'tinker' and 'eloquent'. This involved creating a 'variable' called 'post', assigning data to each field and then utilising 'save' to save each data entry:

Utilising 'tinker' and 'eloquent' to add Data to the Database

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan tinker
Psy Shell v0.9.9 (PHP 7.3.2 ôÇö cli) by Justin Hileman
>>> $post = new App\Post();
=> App\Post {#2932}
>>> $post-> title = 'Post One'
=> "Post One"
>>> $post->body = 'This is the post body'
=> "This is the post body"
>>> $post->save();
=> true
>>>
```

This then reflected in the database as shown below:

The Data now Appeared within the Database

+ Options

	id	title	body	created_at	updated_at
Edit Copy Delete	1	Post One	This is the post body	2019-03-10 15:41:01	2019-03-10 15:41:01

Then, after adding another piece of data to the database, I then realised I needed to use 'php artisan make' to create another posts 'controller', this time with 'CRUD' functionality by using '--resource':

Creating the 'controller' Again, adding 'CRUD' Functionality

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller PostsController --r
esource
Controller created successfully.
```

This now Created the 'controller' with 'CRUD' Functionality (Example Shown Below)

PostsController.php × stylesheet.css × 2019_03_10

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        //
    }
```

After this, to be able to create 'routes' for all of the functions, I needed to include the following in the 'web.php' file, as shown by the tutorial:

Including the Following 'route' for all 'CRUD' Functions

```
Route::resource('posts', 'PostsController');
```

Following on from this and after creating a 'posts' folder in the 'view' folder as well as creating a file called 'index.blade.php', I then tested this as shown on the tutorial. However, I had encountered an issue. The issue was regarding the page not being able to be found when navigating to the 'URL':

Adding some Elements shown on the Tutorial to the 'model'

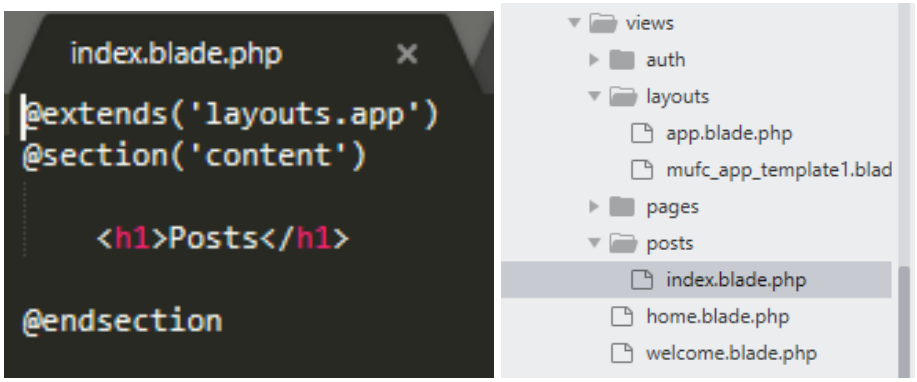
Post.php × web.php

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    //Table Name
    protected $table = 'posts';
    //Primary Key
    public $primaryKey = 'id';
    //Timestamps
    public $timestamps = true;
}
```

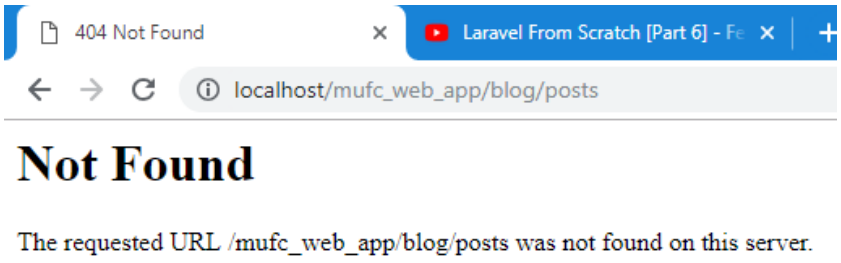
Returning the Newly Created 'index.blade.php' 'view' through the 'controller'

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('posts.index');
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }
```

Including some Test Code for the Newly Created 'index.blade.php' File

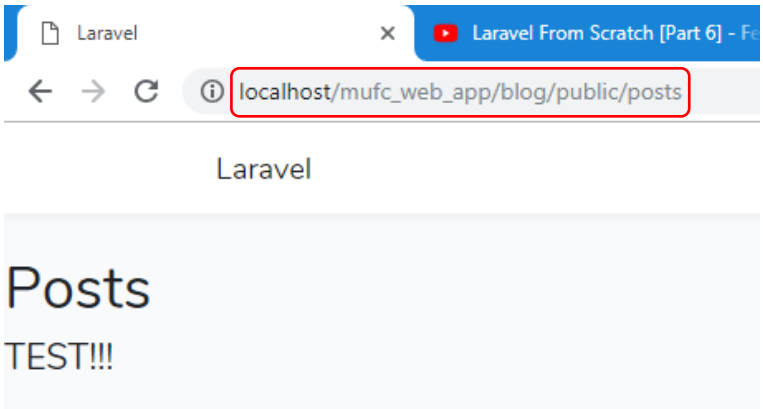


The Outcome on the Web Page – The Page Could not be Found



However, I then navigated to the 'public' folder and typed 'posts' and this then worked as seen below:

The Page now Appeared with the Correct Content



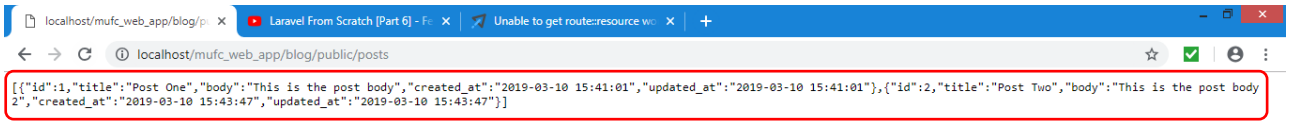
After now knowing that the 'view' was returning, I then entered code into the 'controller' that would return all posts to the page, following the tutorial:

Entering the Code to Return all Posts onto the Page

```
use Illuminate\Http\Request;
use App\Post;

class PostsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return Post::all();
        return view('posts.index');
    }
```

This then successfully outputted the posts onto the page into an 'array' format:

The Posts were now Appearing



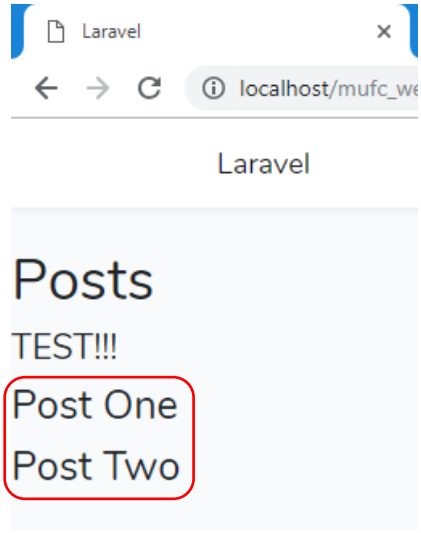
As the above wasn't visually appealing, I then added 'if' and 'else' and 'foreach' statements, as shown on the tutorial, which allowed for the post titles, as displayed in the database, to be placed onto the page for each post if the posts were greater than one. Otherwise, no posts would be displayed onto the page with the message 'No posts found' appearing instead:

Adding the 'if' and 'else' and 'foreach' Statements into the 'index.blade.php' File

index.blade.php | Post.php

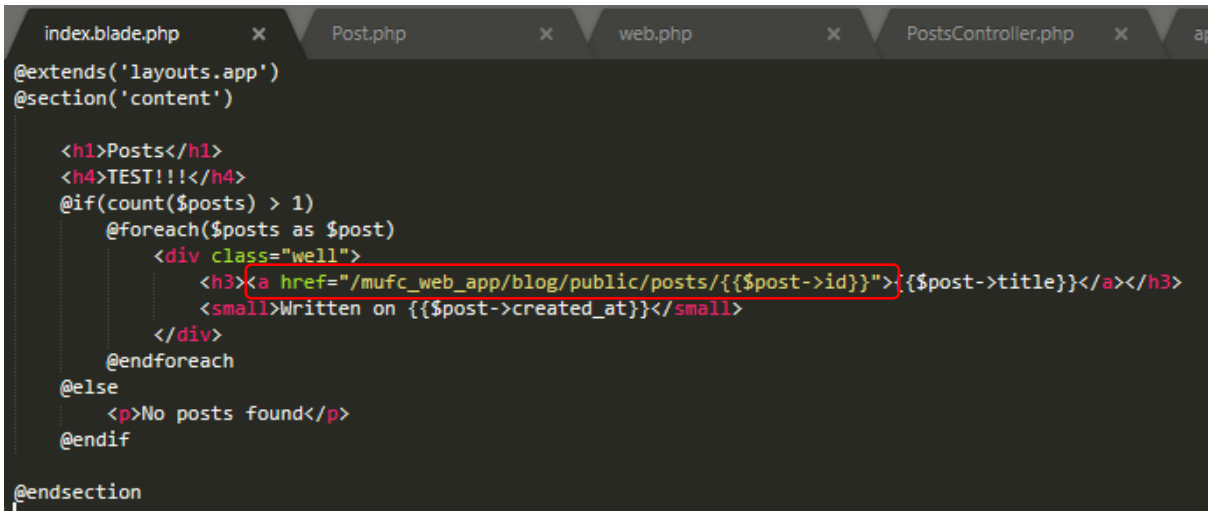
```
@extends('layouts.app')
@section('content')

    <h1>Posts</h1>
    <h4>TEST!!!</h4>
    @if(count($posts) > 1)
        @foreach($posts as $post)
            <div class="well">
                <h3>{{$post->title}}</h3>
            </div>
        @endforeach
    @else
        <p>No posts found</p>
    @endif

@endsection
```

The Outcome on the Page – The Posts now Appeared with their Titles

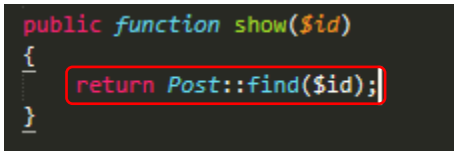


Following on from this, I then followed the tutorial again, this time collecting data when selecting a specific post as shown below:

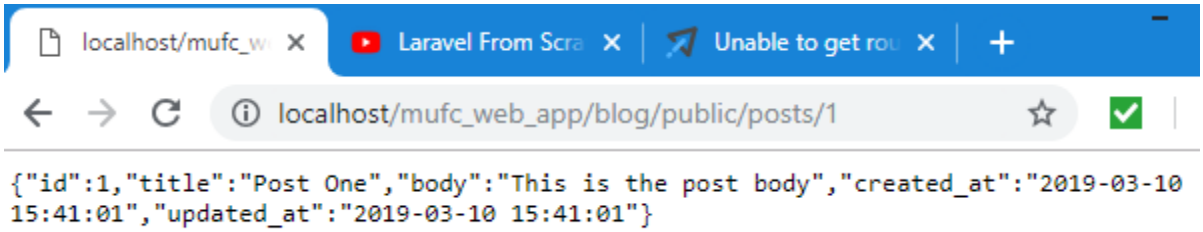
Entering the 'URL' to Collect each Selected Post's Data through it's Specific 'ID' within the 'index.blade.php' File



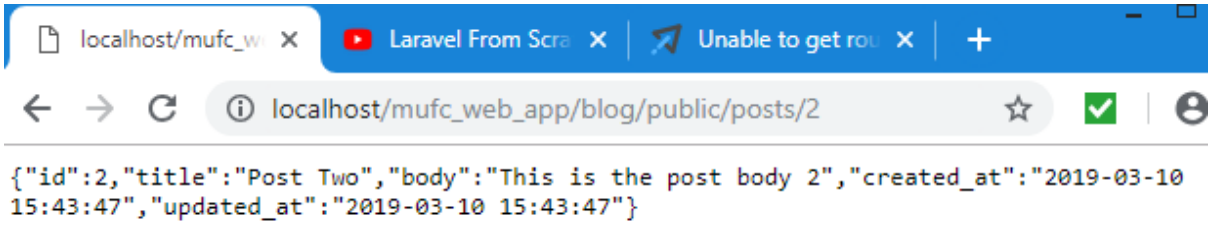
Making the 'controller' Find the Post 'ID' within the 'show' Function to Show Data for a Specific Post



The Result After Selecting the Post 1 Link – The Data Displayed Successfully

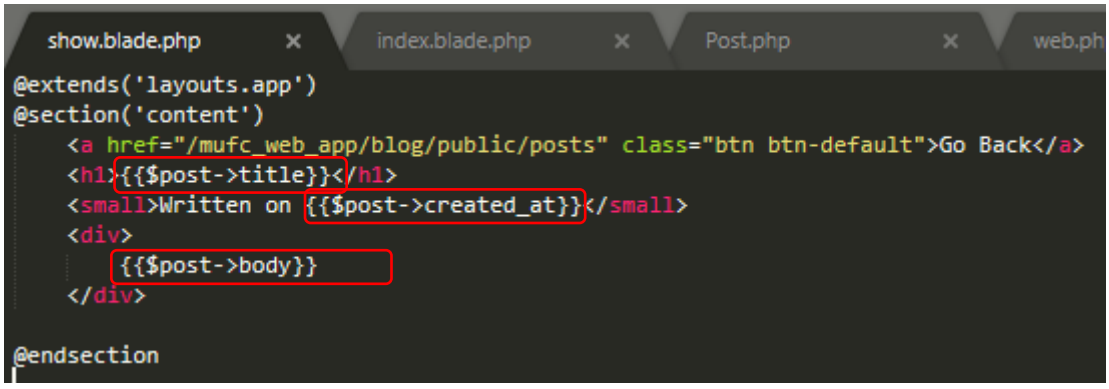


The Result After Selecting the Post 2 Link – The Data Displayed Successfully



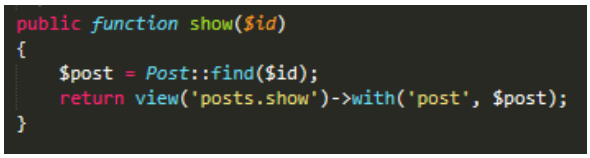
Following on from the previous step, I then created another file called 'show.blade.php' as shown on the tutorial which would allow for displaying the posts in a more user-friendly format. This is evident below:

Creating the 'show.blade.php' File and Collecting Data to Show



As is evident above, a link was included to allow users to return to the previous page with aspects within curly braces relating to collecting data from different fields within the table in the database. After creating the file, this was then set to be returned within the public function called 'show', returning the more user-friendly format to users when selecting a post to view. This is evident below:

Applying 'return view' to Show the File Created before



Following on from this, I then followed the tutorial further, changing the 'if' statement to show posts if the count was higher than '0' as this had caused an issue to occur.

Changing the 'if' Statement in the 'index.blade.php' File

```
@extends('layouts.app')
@section('content')

    <h1>Posts</h1>
    <h4>TEST!!!</h4>
    @if(count($posts) > 0)
        @foreach($posts as $post)
            <div class="well">
                <h3><a href="/mufc_web_app/blog/public/posts/{{$post->id}}">{{$post->title}}</a></h3>
                <small>Written on {{$post->created_at}}</small>
            </div>
        @endforeach
    @else
        <p>No posts found</p>
    @endif

@endsection
```

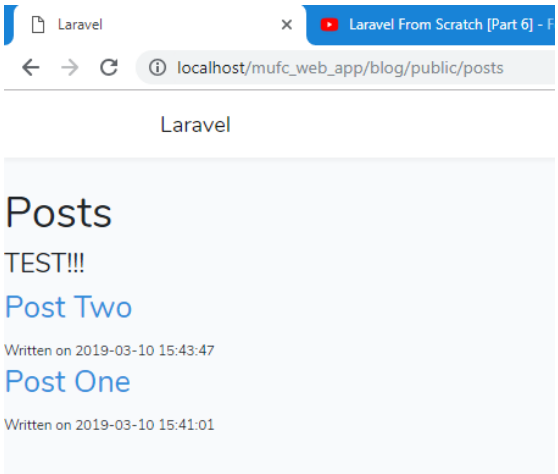
Furthermore, within the 'PostsController', the order was set to descending for the 'title' aspect of each post, displaying the most current post first on the page. This is evident below:

Adding 'desc' to Order Posts in Descending Order by their Title

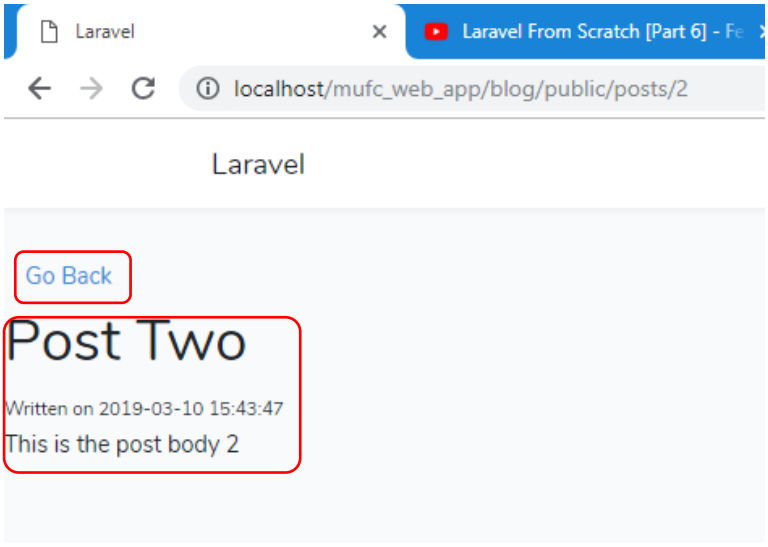
```
public function index()
{
    $posts = Post::orderBy('title','desc')->get();
    return view('posts.index')->with('posts', $posts);
}
```

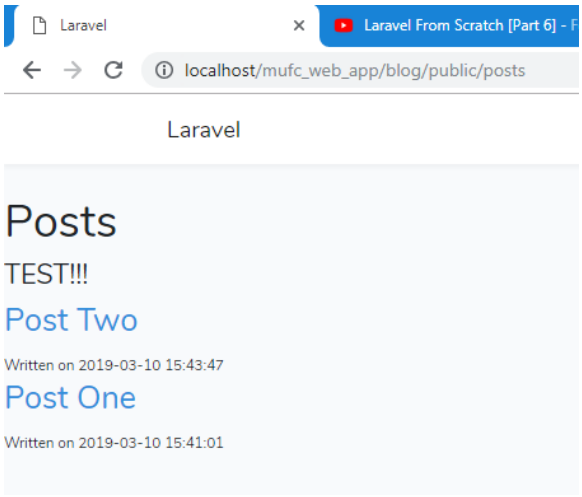
After completing the previous steps, this then produced the required outcome of being able to view each post more visually with posts initially being ordered in descending order, as explained above:

The Posts were now Being Displayed in Descending Order



An Example of Selecting a Post with the Ability to Select the 'Go Back' button to Navigate to the Original Page





At this stage, I believed to have a good understanding to be able to transfer this knowledge across to my project. This will be viewable on the following page.

Transferring Obtained Knowledge Across to the Manchester United Website Application

Establishing Areas of the Application

Initial Process

After undertaking a series of actions following tutorials, I then decided to begin implementing aspects such as tables with the data I required. To begin, I listed the tables/data I would need to show which are viewable below:

- Statistics table for the players (goals, assists, etc.)
- Player profiles table which includes information about each player as well as an image of each player

After the previous process, I then thought that I could create a couple of 'controllers', one for the player profiles and one for the player statistics. However, whilst doing this I realised that it would be better to create one controller to make it easier for the migrations/models and data pulling aspect of the application:

The Process of Creating a Player Profile 'Controller' through the Command Line Interface

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller PlayerProfileContro
ller --resource
Controller created successfully.
```

Deleting the 'Controller' Above After Realising it would be more Beneficial to Utilise one 'Controller' and Creating a new 'Controller' Called 'PlayersController'

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller PlayersController -
-resource
Controller created successfully.
```

Creating a Model and Migration Relating to the Players

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:model Player -m
Model created successfully.
Created Migration: 2019_03_11_101413_create_players_table
```

After then creating the 'migration', I then entered some fields for the table into this and migrated this to be able to experiment and test with before adding more aspects. This can be viewed below:

Adding new Fields to the Migration with their Data Types (e.g. integer)

```
public function up()
{
    Schema::create('players', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->timestamps();
        $table->string('name');
        $table->integer('age');
        $table->string('nationality');
        $table->string('image');
    });
}
```

Executing 'php artisan migrate' in the Command Line to Update the Database

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan migrate
Migrating: 2019_03_11_103642_create_players_table
Migrated:  2019_03_11_103642_create_players_table
```

The Table now Appeared in the Database with the Created Fields

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop
☐	2	created_at	timestamp			Yes	NULL			Change Drop
☐	3	updated_at	timestamp			Yes	NULL			Change Drop
☐	4	name	varchar(191)	utf8mb4_unicode_ci		No	None			Change Drop
☐	5	age	int(11)			No	None			Change Drop
☐	6	nationality	varchar(191)	utf8mb4_unicode_ci		No	None			Change Drop
☐	7	image	varchar(191)	utf8mb4_unicode_ci		No	None			Change Drop

Following on from the previous step, I then utilised 'tinker' to add a couple of players to the table to act as tests, using the 'Premier League' website to help with the information:

Adding the Players to the Database via 'tinker' in the Command Line Interface

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan tinker
Psy Shell v0.9.9 (PHP 7.3.2 OCo cli) by Justin Hileman
>>> $player = new App\Player();
=> App\Player {#2939}
>>> $player-> name = 'Anthony Martial';
=> "Anthony Martial"
>>> $player-> age = '23';
=> "23"
>>> $player->save();
Illuminate/Database/QueryException with message 'SQLSTATE[HY000]: General error:
 1364 Field 'nationality' doesn't have a default value (SQL: insert into `player
s` (`name`, `age`, `updated_at`, `created_at`) values (Anthony Martial, 23, 2019
-03-11 10:52:28, 2019-03-11 10:52:28))'
>>> $player-> nationality = 'France';
=> "France"
>>> $player->save();
Illuminate/Database/QueryException with message 'SQLSTATE[HY000]: General error:
 1364 Field 'image' doesn't have a default value (SQL: insert into `players` (`n
ame`, `age`, `updated_at`, `created_at`, `nationality`) values (Anthony Martial,
 23, 2019-03-11 10:52:28, 2019-03-11 10:52:28, France))'
>>> $player-> image = 'Test';
=> "Test"
>>> $player->save();
=> true
```

```
>>> $player = new App\Player();
=> App\Player {#2944}
>>> $player-> name = 'Marcus Rashford';
=> "Marcus Rashford"
>>> $player-> age = '21';
=> "21"
>>> $player-> nationality = 'England';
=> "England"
>>> $player-> image= 'Test 2';
=> "Test 2"
>>> $player->save();
=> true
>>> quit
Exit:  Goodbye

C:\MAMP\htdocs\mufc_web_app\blog>
```

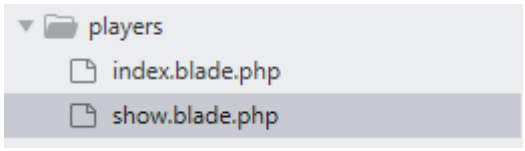
This then Reflected within the Table

	id	created_at	updated_at	name	age	nationality	image
Edit Copy Delete	1	2019-03-11 10:52:28	2019-03-11 10:52:28	Anthony Martial	23	France	https://e1.365dm.com/18/12/ antho...
Edit Copy Delete	2	2019-03-11 10:57:18	2019-03-11 10:57:18	Marcus Rashford	21	England	https://s.hs-data.com/bilder/spieler/gross

Please note that at a later stage I inputted 'URLs' relating to images of the players as this would allow for the images to display on the pages.

After having now established a basic table with a couple of players, I now wanted to begin collecting the players through filters on the current web page. Therefore, I undertook the same approach as that shown in the tutorial followed before, creating both 'index.blade.php' and 'show.blade.php' files:

Creating a 'players' Folder within the 'views' Folder with both the 'index.blade.php' and 'show.blade.php' Files



After doing this, I then currently added the following code to the 'index.blade.php' file, initially integrating filters from 'W3Schools':

The Current 'index.blade.php' File

```
<link href="{{ asset('css/stylesheet.css') }}" rel="stylesheet">

@extends('layouts.mufc_app_template1')

@section('content')
<!--This is a test-->

<div class="player_statistics_container">
    <div class="psc_filter_container">
        <div class="typed_search_container">
            <h3>Custom Search</h3>
            <input type="text" placeholder="Search..">
        </div>
        <div class="age_filter_container">
            <h3>Age</h3>
            <select>
                <option value="">Volvo</option>
                <option value="saab">Saab</option>
                <option value="mercedes">Mercedes</option>
                <option value="audi">Audi</option>
            </select>
        </div>
        <div class="nationality_filter_container">
            <h3>Nationality</h3>
            <select>
                <option value="">Volvo</option>
                <option value="saab">Saab</option>
                <option value="mercedes">Mercedes</option>
                <option value="audi">Audi</option>
            </select>
        </div>
    </div>
    <div class="psc_player_container">
    @if(count($players) > 0)
        @foreach($players as $player)
            <div class="well">
                <h3><a href="/mufc_web_app/blog/public/players/{{$post->id}}">{{$post->title}}</a></h3>
                <small>Written on {{$post->created_at}}</small>
            </div>
        @endforeach
    @else
        <p>No posts found</p>
    @endif
    </div>
</div>

@endsection
```

As is evident, I integrated a template through '@extends' as well as specifying where the content of this page would be placed through '@section('content')'. Furthermore, a parent container of 'player_statistics_container' was created to contain both the sections for the filters and actual players themselves with 'psc_filter_container' and 'psc_player_container' relating to these sections. It is also worth noting that I integrated the same 'if' statement as that utilised in the tutorial with a 'foreach' statement as well to format each player in specific format. After doing this, I also then added code to the 'show.blade.php' file as will be seen below:

The Current 'show.blade.php' File

```
@extends('layouts.mufc_app_template1')
@section('content')
    <a href="/mufc_web_app/blog/public/players">Go Back</a>
    <h1>{{$player->title}}</h1>
    <small>Written on {{$player->created_at}}</small>
    <div>
        {{$player->nationality}}
    </div>

@endsection
```

As is evident above, I also integrated the same template as that utilised for the 'index.blade.php' file through '@extends' to create a consistent appearance, utilising '@section('content')' to specify where the content would be placed for the page. Furthermore, again, I integrated the same code as that utilised in the tutorial to help which would collect some of the player's information and display this on the page.

After the previous process, in the 'PlayersController', I then added the same as that shown in the tutorial undertaken before, utilising 'return view' to return the relevant 'blade' files to the user, ordering players in descending order by their 'name'. This would allow for displaying the selected player through 'show', relating to specific players through their 'ID'.

Adding the Required Code to the 'PlayersController' for both the Public Functions of 'index' and 'show'

```
public function show($id)
{
    $player = Player::find($id);
    return view('players.show')->with('player', $player);
}
```

```
public function index()
{
    $players = Player::orderBy('name','desc')->get();
    return view('players.index')->with('players', $players);
}
```

After the previous process, although not required, I then added the same code to the 'player' file in case this was needed:

Adding the Same Code as that Shown in the Tutorial Before to the 'player' File

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Player extends Model
{
    //Table Name
    protected $table = 'players';
    //Primary Key
    public $primaryKey = 'id';
    //Timestamps
    public $timestamps = true;
}
```

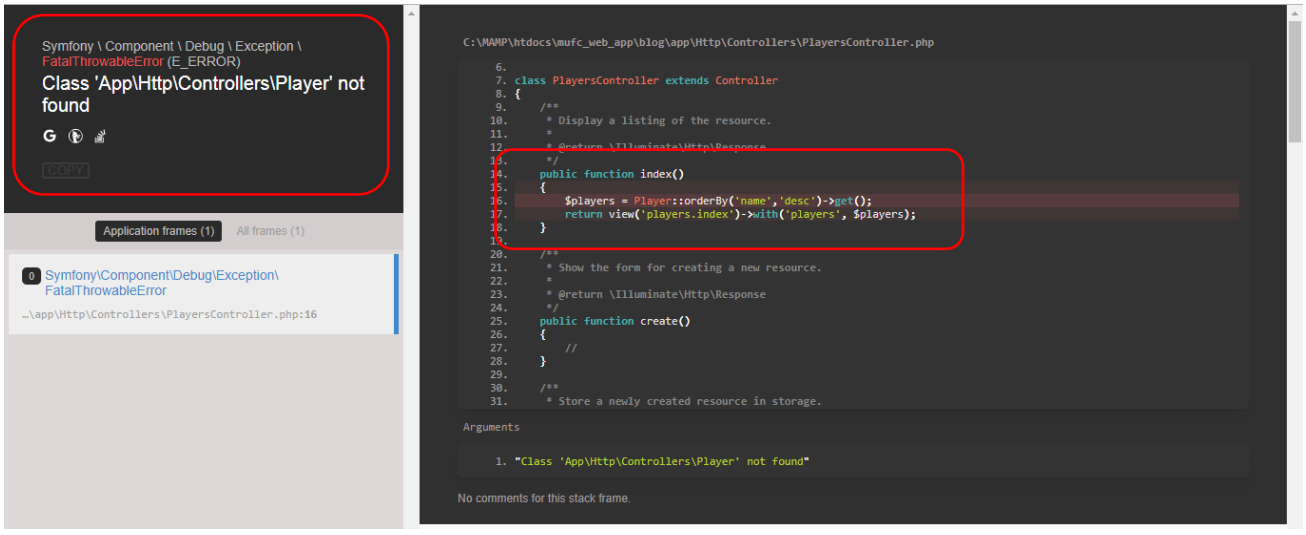
Furthermore, in the file relating to the 'routes', I specified a 'route' relating to the previously shown aspects to be able to allow the page to function correctly:

Adding the Following 'route' to the Web.php File

```php
Route::resource('players', 'PlayersController');
```

After attempting to test the page, I then encountered the following issue where a 'class' could not be found:

The Encountered Error



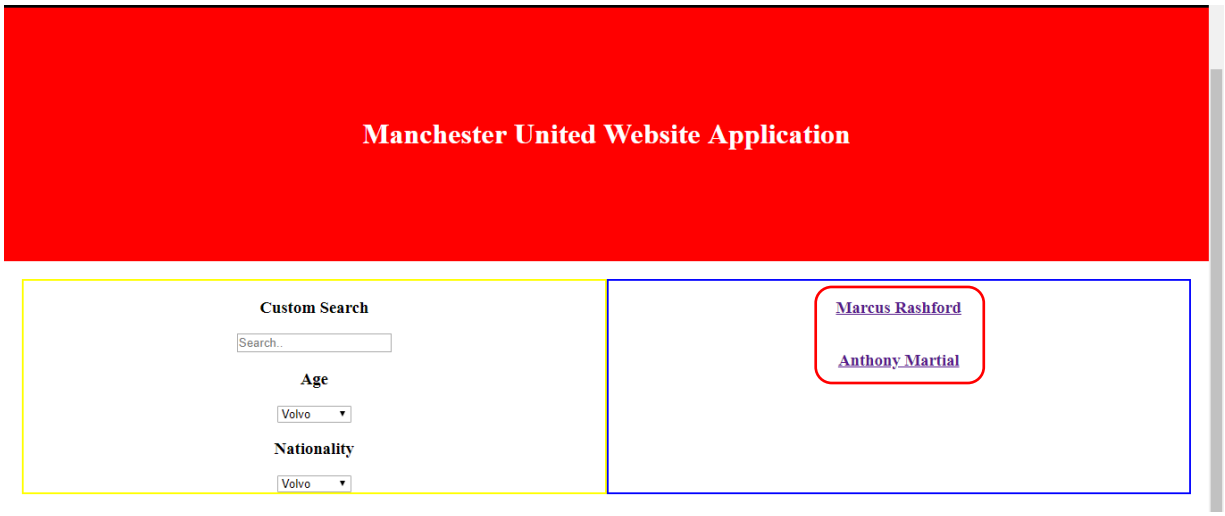
However, I had then realised that I had forgotten to add the highlighted code to the 'Controller' file shown below:

Adding the Forgotten Code to the 'Controller' File

```
use Illuminate\Http\Request;
use App\Player;

class PlayersController extends Controller
{
```

After adding the missing code, this then resolved the issue and now the players were showing on the page through their names. This is viewable on the following page.

The Page now Appeared with the Data from the Database



Adding more Aspects to the Database

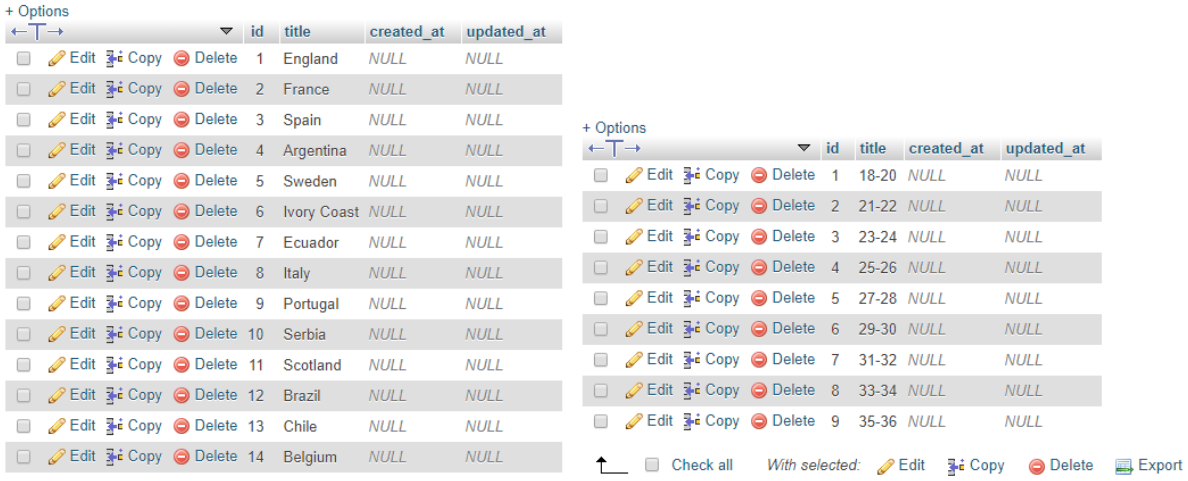
As at one stage, I couldn't make the filters aspect function correctly, as will be seen later on, I therefore decided to add the data to the database for each player instead regarding their name, nationality, age and image as well as updating the nationalities/categories and ages tables on 'PHPMyAdmin':

Examples of Inserted Players into the Players Table

+ Options

	id	created_at	updated_at	name	age	nationality	image
Edit Copy Delete	1	2019-03-11 10:52:28	2019-03-11 10:52:28	Anthony Martial	23	France	https://e1.365dm.com/18/
Edit Copy Delete	2	2019-03-11 10:57:18	2019-03-11 10:57:18	Marcus Rashford	21	England	https://s.hs-data.com/bilde
Edit Copy Delete	3	NULL	NULL	Paul Pogba	26	France	https://upload.wikimedia.o
Edit Copy Delete	4	NULL	NULL	David De Gea	28	Spain	https://www.standardmedi
Edit Copy Delete	5	NULL	NULL	Sergio Romero	32	Argentina	https://upload.wikimedia.o
Edit Copy Delete	6	NULL	NULL	Victor Lindelof	24	Sweden	https://upload.wikimedia.o
Edit Copy Delete	7	NULL	NULL	Eric Bailly	24	Ivory Coast	https://e0.365dm.com/17/
Edit Copy Delete	8	NULL	NULL	Phil Jones	27	England	https://upload.wikimedia.o
Edit Copy Delete	9	NULL	NULL	Marcos Rojo	28	Argentina	https://upload.wikimedia.o
Edit Copy Delete	10	NULL	NULL	Chris Smalling	29	England	https://upload.wikimedia.o
Edit Copy Delete	11	NULL	NULL	Ashley Young	33	England	https://upload.wikimedia.o
Edit Copy Delete	12	NULL	NULL	Luke Shaw	23	England	https://upload.wikimedia.o
Edit Copy Delete	13	NULL	NULL	Antonio Valencia	33	Ecuador	https://www.eluniverso.co

Updating the Tables Relating to Nationalities and Ages



+ Options

	id	title	created_at	updated_at
Edit Copy Delete	1	England	NULL	NULL
Edit Copy Delete	2	France	NULL	NULL
Edit Copy Delete	3	Spain	NULL	NULL
Edit Copy Delete	4	Argentina	NULL	NULL
Edit Copy Delete	5	Sweden	NULL	NULL
Edit Copy Delete	6	Ivory Coast	NULL	NULL
Edit Copy Delete	7	Ecuador	NULL	NULL
Edit Copy Delete	8	Italy	NULL	NULL
Edit Copy Delete	9	Portugal	NULL	NULL
Edit Copy Delete	10	Serbia	NULL	NULL
Edit Copy Delete	11	Scotland	NULL	NULL
Edit Copy Delete	12	Brazil	NULL	NULL
Edit Copy Delete	13	Chile	NULL	NULL
Edit Copy Delete	14	Belgium	NULL	NULL

+ Options

	id	title	created_at	updated_at
Edit Copy Delete	1	18-20	NULL	NULL
Edit Copy Delete	2	21-22	NULL	NULL
Edit Copy Delete	3	23-24	NULL	NULL
Edit Copy Delete	4	25-26	NULL	NULL
Edit Copy Delete	5	27-28	NULL	NULL
Edit Copy Delete	6	29-30	NULL	NULL
Edit Copy Delete	7	31-32	NULL	NULL
Edit Copy Delete	8	33-34	NULL	NULL
Edit Copy Delete	9	35-36	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

After undertaking the previous process, all the players now appeared on the page as well as the nationality filters. This can be seen on the following page.


Manchester United Website Application

Furthermore, during the latter stages of the project, after having added all that was required to create a prototype, I then currently had each of the following tables with data viewable below. Please note that the images shown were from an exported 'PDF' as it was difficult to capture all information from the actual tables in 'PHPMyAdmin':

The 'players' Table

id	created_at	updated_at	name	age	nationality	position	image	category_id	age_id	position_id	full_season_statistic_id	goals	assists	passes	shots	tackles	fouls	offsides	minutes_played	yellow_cards	red_cards	saves
1	2019-03-11 10:52:28	2019-03-11 10:52:28	Anthony Martial	23	France	Forward	https://e1.365dm.com/18/12/768x432/skysports-anthony-martial-manchester-united_4527686.jpg?20181222185138	2	3	4	1	9	5	731	49	15	17	8	1581	1	0	0
2	2019-03-11 10:57:18	2019-03-11 10:57:18	Marcus Rashford	21	England	Forward	https://s.hs-data.com/bilder/spieler/gross/328453.jpg	1	2	4	2	7	5	576	61	14	21	9	1807	3	0	0
3			Paul Pogba	26	France	Midfielder	https://upload.wikimedia.org/wikipedia/commons/f/f0/Paul_Pogba_2017-03-09.jpg	2	4	3	3	6	10	1721	76	33	44	3	2151	5	1	0
4			David De Gea	28	Spain	Goalkeeper	https://www.standardmedia.co.ke/images/sunday/man_united_de_gea_t5c8def1d72f8e.jpg	3	5	1	4	0	0	942	0	0	0	0	3330	0	0	115
5			Sergio Romero	32	Argentina	Goalkeeper	https://upload.wikimedia.org/wikipedia/commons/f/f0/Sergio_Romero_%28ManUtd%29.jpg	4	7	1	5	0	0	20	0	0	0	0	90	0	0	3
6			Victor Lindelof	24	Sweden	Defender	https://upload.wikimedia.org/wikipedia/commons/c/cc/CSKA-MU_2017_%286%29.jpg	5	3	2	6	0	0	751	1	17	10	0	1284	1	0	0
7			Eric Bailly	24	Ivory Coast	Defender	https://e0.365dm.com/17/08/768x432/skysports-eric-baily-manchester-united-fbotball_4076443.jpg?20170818085211	6	3	2	7	1	0	426	3	17	10	0	1000	2	0	0
8			Phil Jones	27	England	Defender	https://upl	1	5	2	8	0	0	1083	2	25	10	0	1975	2	0	0

id	created_at	updated_at	name	age	nationality	position	image	category_id	age_id	position_id	full_season_statistic_id	goals	assists	passes	shots	tackles	fouls	offsides	minutes_played	yellow_cards	red_cards	saves
							edia.org/wikipedia/commons/5/56/Cskamu_43.jpg															
9			Marcos Rojo	28	Argentina	Defender	https://upload.wikimedia.org/wikipedia/commons/7/7f/Marcos_Rojo_vs_Rostov.jpg	4	5	2	9	0	0	358	0	7	8	0	640	6	0	0
10			Chris Smalling	29	England	Defender	https://upload.wikimedia.org/wikipedia/commons/6/65/CSKA-MU_2017_%2815%29.jpg	1	6	2	10	4	0	1137	13	40	24	2	2535	4	0	0
11			Ashley Young	33	England	Defender	https://upload.wikimedia.org/wikipedia/commons/b/ba/Ashley_Young_2017-03-09.jpg	1	8	2	11	2	4	1285	18	53	29	1	2445	7	0	0
12			Luke Shaw	23	England	Defender	https://upload.wikimedia.org/wikipedia/commons/thumb/4/4a/Shaw_-_July_2015_%28cropped%29.jpg/359px-Shaw_-_July_2015_%28cropped%29.jpg	1	3	2	12	0	0	477	12	18	4	0	788	2	0	0
13			Antonio Valencia	33	Ecuador	Defender	https://www.eluniverso.com/sites/default/files/styles/powgallery_1024/public/fotos/2019/03/16411206.jpg?itok=QLUfR7Ba	7	8	2	13	3	1	1602	16	57	38	4	2742	7	0	0
14			Matteo Darmian	29	Italy	Defender	https://e0.365dm.com/18/07/768x432/skysports-football-matteo-darmian_43754	8	6	2	14	0	0	195	2	16	5	0	444	0	0	0

id	created_at	updated_at	name	age	nationality	position	image	category_id	age_id	position_id	full_season_statistic_id	goals	assists	passes	shots	tackles	fouls	offsides	minutes_played	yellow_cards	red_cards	saves
							20.jpg?2018072925643															
15			Diogo Dalot	20	Portugal	Defender	https://e2.365dm.com/18/12/768x432/skysports-diogo-dalot-manchester-united_4510605.jpg?20181206123402	9	1	2	15	0	0	0	0	0	0	0	0	0	0	0
16			Juan Mata	30	Spain	Midfielder	https://e2.365dm.com/16/10/768x432/juan-mata-manchester-united_3805541.jpg?20161010163358	3	6	3	16	3	5	1215	28	18	12	18	1930	1	0	0
17			Jesse Lingard	26	England	Midfielder	https://static.independent.co.uk/s3fs-public/thumbnails/image/2019/01/14/12/lingard.jpg	1	4	3	17	8	5	932	56	30	34	4	1823	2	0	0
18			Ander Herrera	29	Spain	Midfielder	https://upload.wikimedia.org/wikipedia/commons/0/04/Ander_Herrera_2017-03-09.jpg	3	6	3	18	0	2	1043	12	56	28	2	1260	5	0	0
19			Nemanja Matic	30	Serbia	Midfielder	https://s.hs-data.com/bilder/spieler/gross/95200.jpg	10	6	3	19	1	1	2601	18	70	33	2	3119	6	0	0
20			Scott McTominay	22	Scotland	Midfielder	https://images2.minutemediacdn.com/image/upload/c_fill,w_912,h_516,f_auto,q_auto,g_auto/shape/cover/sport/tottenham-hotspur-v-manchester-united-premier-league-5c45e	11	2	3	20	0	0	343	1	11	7	0	629	2	0	0

id	created_at	updated_at	name	age	nationality	position	image	category_id	age_id	position_id	full_season_statistic_id	goals	assists	passes	shots	tackles	fouls	offsides	minutes_played	yellow_cards	red_cards	saves
							01ae04b229b90000003.jpg															
21			Fred	26	Brazil	Midfielder	https://images.performgroup.com/di/library/GOAL/bb/5e/fred-manchester-united_p6r3qeroj0xv1dwfu9za9hh11.jpg?t=815824391&quality=100&w=640&h=360	12	4	3	21	0	0	0	0	0	0	0	0	0	0	0	0
22			Andreas Pereira	23	Brazil	Midfielder	https://e2.365dm.com/18/08/768x432/skysports-andreas-pereira-manchester-united_4393325.jpg?20180818120009	12	3	3	22	0	0	0	0	0	0	0	0	0	0	0	0
23			Alexis Sanchez	30	Chile	Forward	https://e1.365dm.com/18/12/768x432/skysports-alexis-sanchez-manchester-united_4526355.jpg?20181812211522 51	13	6	4	23	2	3	582	19	21	10	10	1046	1	0	0
24			Romelu Lukaku	25	Belgium	Forward	https://upload.wikimedia.org/wikipedia/commons/6/60/Romelu_Lukaku_27_September_2017_cropped.jpg	14	4	4	24	16	7	721	86	7	31	18	2869	4	0	0

As is evident above, I had added the actual statistics for each of the players such as 'goals' and 'shots' to be able to use this within the application. Please note that I had also added a field called

'full_season_statistic_id' as I wanted to be able to allow users to view individual matchday statistics and full season statistics by selecting filters for each player. Therefore, I thought adding this would allow for filtering of the full season statistics aspect. This was decided to not be added to the final prototype due to time constraints, as will be explained later in this document.

The 'ages' Table

id	title	created_at	updated_at
1	18-20		
2	21-22		
3	23-24		
4	25-26		
5	27-28		
6	29-30		
7	31-32		
8	33-34		
9	35-36		

As is evident above, the 'ages' table remained the same and the purpose of this table was to allow for titles of age groups such as '18-20' to be displayed as filters on various pages of the website application.

The 'categories' Table (Relating to Nationalities)

id	title	created_at	updated_at
1	England		
2	France		
3	Spain		
4	Argentina		
5	Sweden		
6	Ivory Coast		
7	Ecuador		
8	Italy		
9	Portugal		
10	Serbia		
11	Scotland		
12	Brazil		
13	Chile		
14	Belgium		

This table remained the same as well with the purpose of allowing the titles of different nationality groups such as 'Sweden' to be displayed as filters on various pages of the website application.

The 'positions' Table (Relating to Nationalities)

id	title	created_at	updated_at
1	Goalkeeper		
2	Defender		
3	Midfielder		
4	Forward		

As is evident above, I had also created a 'positions' table to act as the same purpose of the previous two tables. This was to allow the titles of different position groups such as 'Forward' to be displayed as filters on various pages of the website application.

Regarding the use and functionality of the data/tables shown above, this will have been explained within different sections of this document.

Utilising Data within the Website Application

Creating the Custom Search Aspect

As I had now successfully managed to collect data and place this into the page, now I could start manipulating the data through the filters shown to the left. To begin this process, I integrated code from an online source to allow for a custom search box to function. However, I experienced the following problem displayed below:

Integrating the Code into the 'index.blade.php' File from the Online Resource

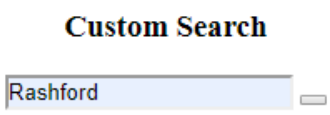
```
<div class="player_statistics_container">
    <div class="psc_filter_container">
        <div class="typed_search_container">
            <h3>Custom Search</h3>
            <form action="/mufc_web_app/blog/public/players" method="POST" role="search">
                {{ csrf_field() }}
                <div class="input-group">
                    <input type="text" class="form-control" name="q"
                        placeholder="Search users"> <span class="input-group-btn">
                        <button type="submit" class="btn btn-default">
                            <span class="glyphicon glyphicon-search"></span>
                        </button>
                    </span>
                </div>
            </form>
        </div>
```

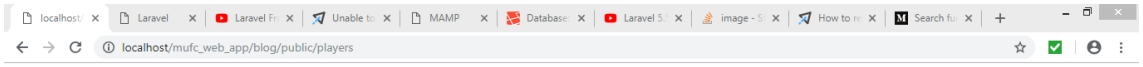
Integrating the Code into the 'web.php' File from the Online Resource

```
Route::any('mufc_web_app/blog/public/players',function(){
    $q = Input::get ( 'q' );
    $player = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($player) > 0)
        return view('players.index')->withDetails($player)->withQuery ( $q );
    else return view ('players.index')->withMessage('No Details found. Try to search again !');
});
```

As is evident above, the main aspect to note was the 'variable' called 'q' which would obtain the user's input and analyse if any data in the database matched this. If it did, then the 'index.blade.php' file would be returned with players relating to the term entered by the user. If it didn't, then the message of 'No Details found. Try to search again !' would appear instead. Furthermore, 'POST' was utilised due to the fact that the user would be sending data to the database as opposed to requesting for certain data through a dropdown filter/category. The outcome of this can be viewed below with the error:

The Outcome Before Searching and After – A Blank Page Appeared





After experiencing the previous issue, I then decided to research further, integrating other code to understand if this would work but this didn't due to the fact that the 'URL' wasn't one recognised by the application:

Creating a new 'Controller' called 'SearchController' and Integrating further Discovered Code

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class SearchController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
if (Input::has('query')) {
                $query = Input::get('query');

            $results = Model::whereRaw("MATCH(name,content) AGAINST(? IN BOOLEAN MODE)", array($query))->remember(1440)->orderBy
                ('id','ASC')->get();

            if ($results->count() > 0) {
                return View::make('search.main')->with('results',$results)->with('query', $query)->with('title', 'Search Result
                    for '.$query);
            } else {
                return '<h3>Sorry, No results for ' . $query . ' </h3>';
            }
        }

        return Redirect::to('/');
    }
    }
```

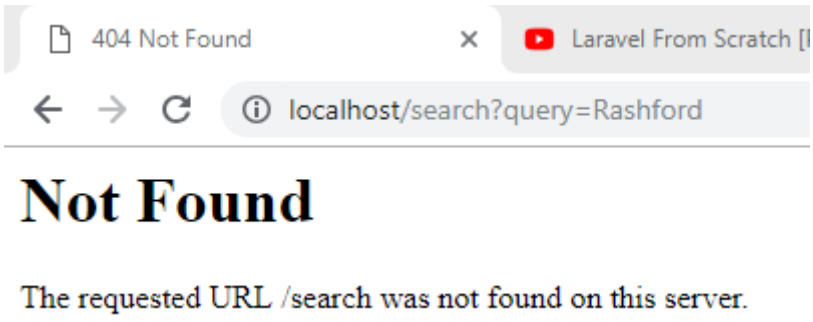
Integrating further Discovered Code into the 'index.blade.php' File

```
<div class="player_statistics_container">
    <div class="psc_filter_container">
        <div class="typed_search_container">
            <h3>Custom Search</h3>
            <form action="/search" method="get">
<label for="query"></label>
    <input type="query" id="query" name="query">
    <input type="submit">
            </form>
```

Creating a new 'route' for the Search Aspect, as Understood from Research

```
Route::get('search', array(
    'as'    =>  'search',
    'uses'  =>  'SearchController@index'
));
```

The Outcome on the Web Page – This Caused a Non-existent 'URL' to Appear



After struggling with this aspect, I then reverted back to the originally displayed code but struggled with this also.

Therefore, as a result, I then sought advice and was helped to resolve the issue by a fellow class colleague. Within the 'web.php' file, the search 'route' was changed to 'any' as well as passing '$input' into the function and changing 'Input::get' to include the 'query' aspect. Furthermore, the view returned also was changed to match that of the required page with the 'index.blade.php' file of the players folder. This changed code can be viewed below:

The Changed 'web.php' File

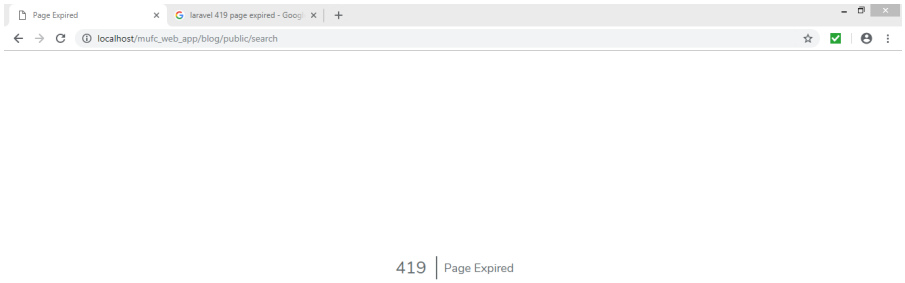
```
Route::any('/search', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($players) > 0){
        return view('players.index')->with('players', $players)->withQuery ( $q );
    }
    else return view ('players.index')->withMessage('No Details found. Try to search again !');
});
```

Additionally, the 'index.blade.php' form was changed so that the method would be 'POST' as the user would be sending information rather than requesting it, as explained by the class colleague. The 'query' aspect matched that of the 'query' in the 'web.php' file and a key line of code, which was missing, was integrated which was the 'csrf_field'. Before this, I was experiencing the following error:

Making Alterations to the 'index.blade.php' with Assistance

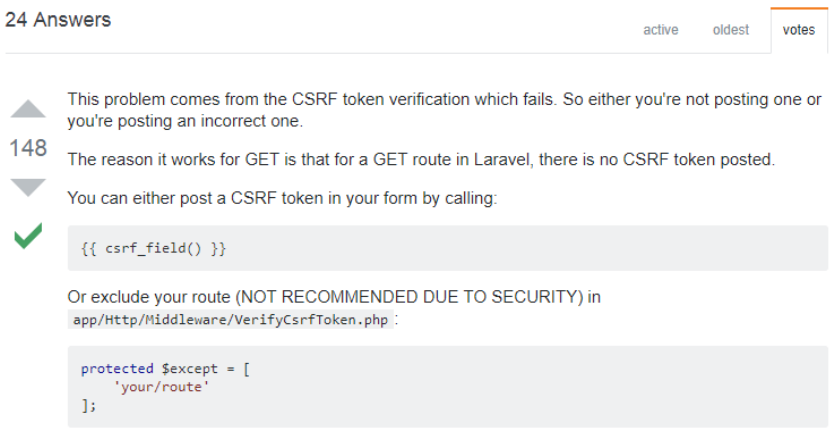
```
<div class="player_statistics_container">
    <div class="psc_filter_container">
        <div class="typed_search_container">
            <h3>Custom Search</h3>
            <form action="http://localhost/mufc_web_app/blog/public/search" method="post">
                {{ csrf_field() }}
<label for="query"></label>
    <input type="query" id="query" name="query">
    <input type="submit">
            </form>
        </div>
```

The Issue Before Integrating the 'csrf_field' Aspect

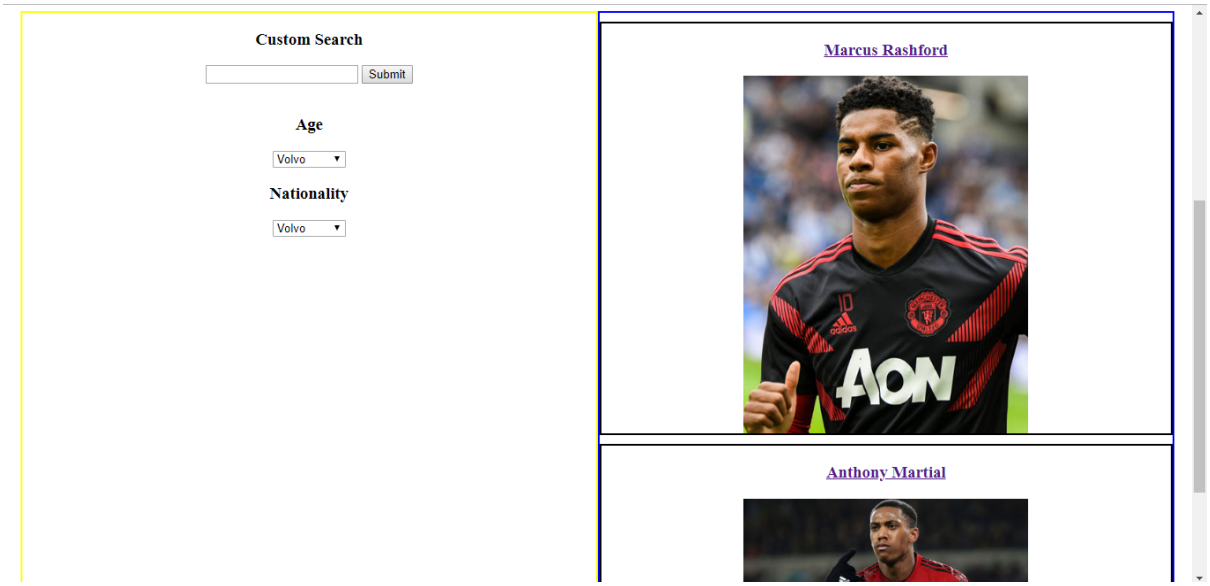


After adding in the 'csrf_field' aspect, as shown above, this then resolved the issue and the search was now working with players now appearing with the same phrases as that entered into the custom search:

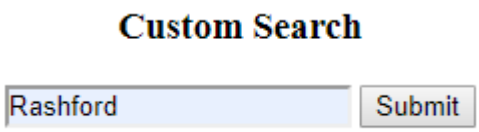
Finding the Solution on 'Stack Overflow' and Confirming with the Class Colleague



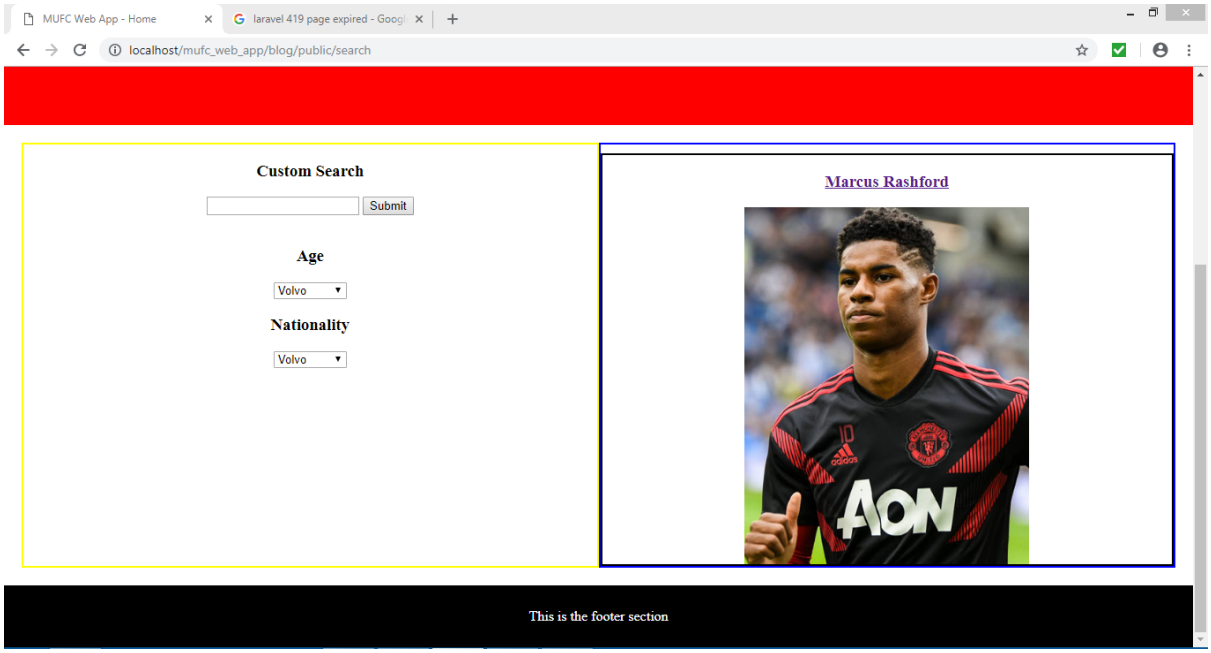
Before Searching for a Player - All Players were Displayed



Whilst Searching for a Player



After Searching for a Player – Those Players Matching the Search Only now Appeared on the Page



Creating Filters to Specify Data

After having now managed to allow the custom search to function with assistance, I now wanted to allow the filters to work underneath also. Therefore, to begin, I undertook research regarding this area. First of all, I integrated the following discovered code but was experiencing some issues with the fact that this was causing an error as displayed below:

Integrating the Highlighted Code into the 'index.blade.php' File

```
    <div class="psc_player_container">
<!--    @if(count($players) > 0)
        @foreach($players as $player)
            <div style="width: 100%; height: auto; border: 2px solid black; margin-top: 10px;">
                <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}">{{$player->name}}</a></h3>
                <img src="{{$player->image}}" style="width: 50%; height: auto;">
            </div>
        @endforeach
    @else
        <p>No players found</p>
    @endif-->
    @foreach ($categories as $category)
        <select onchange="filter(this.value)">
            @if($category->id == Input::get('category')
                <option selected="selected" value="{{ $category->id }}">{{ $category->name }}</option>
            @else
                <option value="{{ $category->id }}">{{ $category->name }}</option>
            @endif
        </select>
    @endforeach
    </div>
</div>

<script>
    function filter(id)
    {
        window.location.href = {{ URL::action('Controller@filter') }} + '/' + id;
</script>
```

As is evident above, I integrated a 'foreach' statement where the filter was meant to change when selecting an option. The '$category->id' would allow for changing options with the '$category->name' placing the different names of each option as text onto the screen. Furthermore, the 'JavaScript' code would allow for linking to a 'controller' to allow for the whole aspect to function correctly.

After adding the code above, I then created a new 'controller', adding some more code from resources online, as seen below:

Creating a new 'Controller' and Integrating Code into this

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller MyController --reso
urce
Controller created successfully.
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class MyController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
$images = Image::all();

    $categories = ..\Models\Category::all();

    return View::make('index', array('images' => $images, 'categories' => $categories));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

public function filter($category){

    $images = Image::where('category_id', '=', $category);

    $categories = ..\Models\Category::all();

    return View::make('index', array('images' => $images, 'categories' => $categories));

}
```

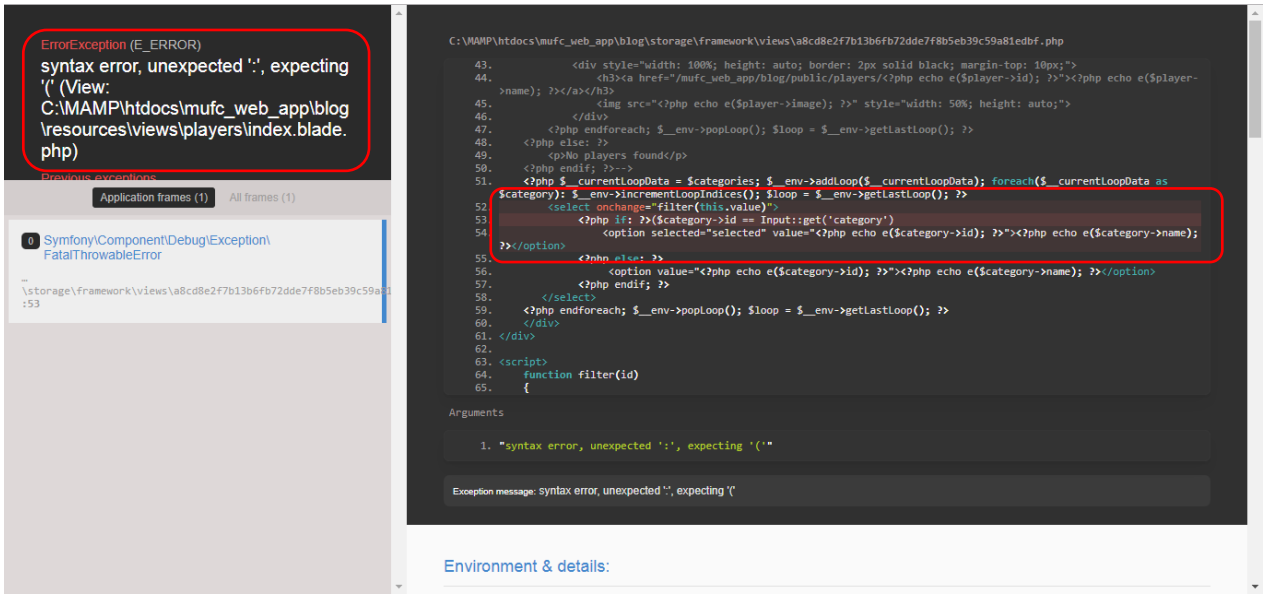
Here a public function of 'filter' was created with a 'variable' being defined for the categories to allow for images to be displayed of players relating to the 'category_id'. Furthermore, the categories 'variable' was also placed in the public function of 'index' to allow this to fully function. The final aspect to note is that I then added new 'routes' to the 'web.php' file through online resource help, as seen below:

Adding 'routes' to the 'web.php' File

```php
Route::get('/', array('uses' => 'MyController@index'));
Route::get('/{category?}', array('uses' => 'MyController@filter'));
```

As stated before, I then encountered an error which is viewable below:

The Current Error I was Experiencing



Despite attempting to resolve the error through changing ':' to '(' and also experimenting with other aspects, I couldn't understand why this wasn't functioning correctly. Therefore, I integrated other discovered code instead to be able to progress faster with the project. However, whilst doing this, I was unable to allow for this to work successfully either. The current code at this stage can be viewed below:

The Current 'index.blade.php' File with the Newly Integrated Code

```
    </div>
    <div class="psc_player_container">
    @if(count($players) > 0)
        @foreach($players as $player)
            <div style="width: 100%; height: auto; border: 2px solid black; margin-top: 10px;">
                <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}">{{$player->name}}</a></h3>
                <img src="{{$player->image}}" style="width: 50%; height: auto;">
            </div>
        @endforeach
    @else
        <p>No players found</p>
    @endif
    @if (request->get('age') === '18-20')
        @foreach($players as $player->orderBy('age', 'desc');
            <div style="width: 100%; height: auto; border: 2px solid black; margin-top: 10px;">
                <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}">{{$player->name}}</a></h3>
                <img src="{{$player->image}}" style="width: 50%; height: auto;">
            </div>
        @endforeach
    @else
        <p>No players found</p>
    @endif
    </div>
</div>
```

As is evident above, a similar approach was undertaken as before regarding adding 'foreach' statements. However, regarding the age aspect, an 'if' statement was integrated where I believed this would collect a certain age group and display players in this age group. After this, with help from online resources, I integrated the following code into the 'PlayersController' file:

Integrating Code into the 'PlayersController' File

```
public function index()
{
    $players = Player::orderBy('name','desc')->get();
        $players->orderBy(request->get('age'));
    return view('players.index')->with('players', $players);
}
```

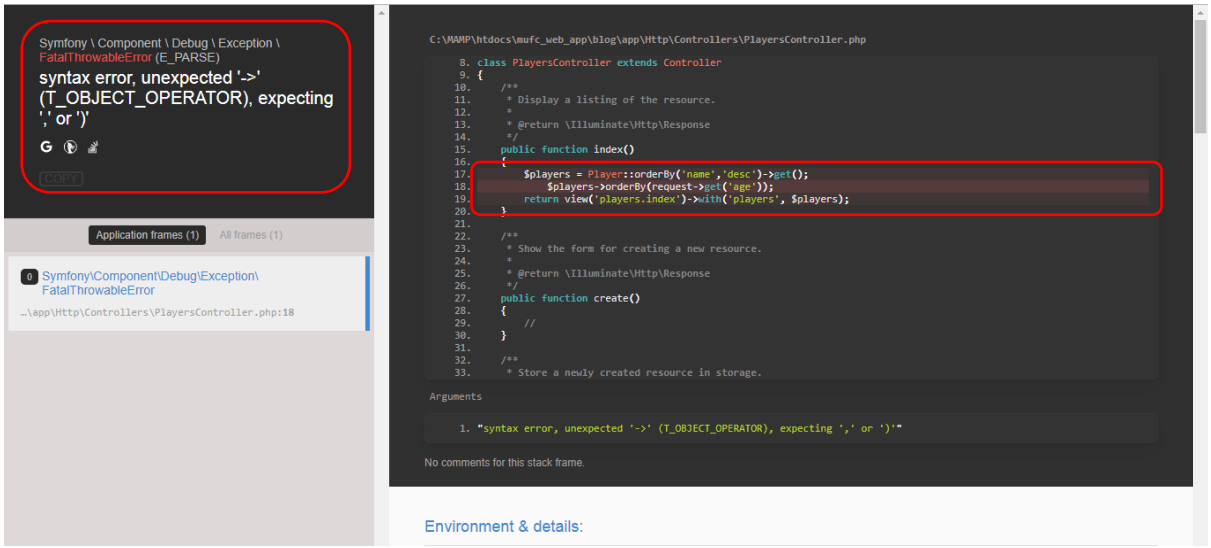
As can be seen above, the default would display all players, ordering these in descending order by name. However, 'orderBy(request->get('age'));' was also included to allow for filtering by age group, returning the view of the 'index.blade.php' file. Following on from this, I then also integrated new code into the 'web.php' file as can be seen on the following page.

Integrating Code into the 'web.php' File

```
Route::any('/search', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($players) > 0){
        return view('players.index')->with('players', $players)->withQuery ( $q );
    }
    else return view ('players.index')->withMessage('No Details found. Try to search again !');
});

Route::any('/search', function(Input $input){


});
```

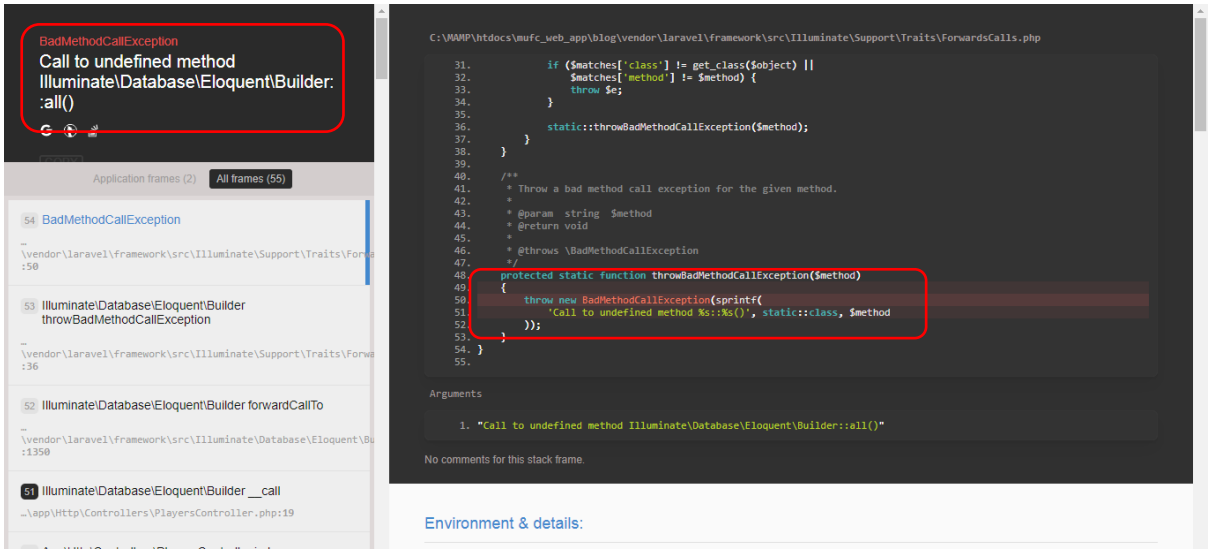
For this, I added a 'route' for the previously integrated aspect. As stated before, I then experienced issues with an error, as displayed below:

The Error After Testing the Previously Integrated Code

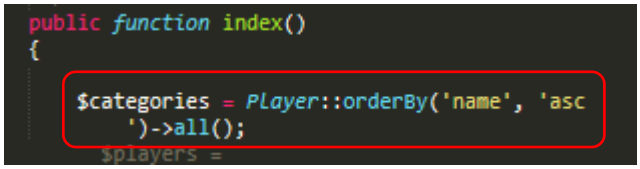


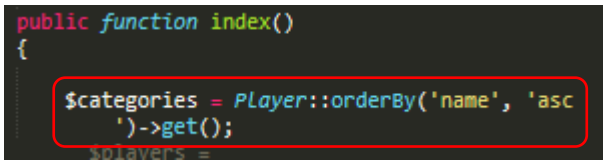
After the previously failed attempt, I then decided to view a tutorial on 'YouTube' to help myself understand how to integrate filters instead. Whilst following this tutorial, I encountered issues with examples shown below. Some of these issues were those shown in the tutorial:

An Example of an Issue Experienced



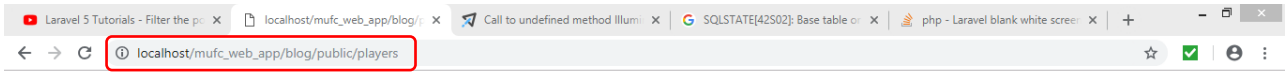
Changing 'all' to 'get' within the 'Controller', as Displayed on the Tutorial





At a later stage, I then realised that I needed to return a view as I was experiencing a blank page as seen on the following page.

The Issue with the Blank Page Appearing

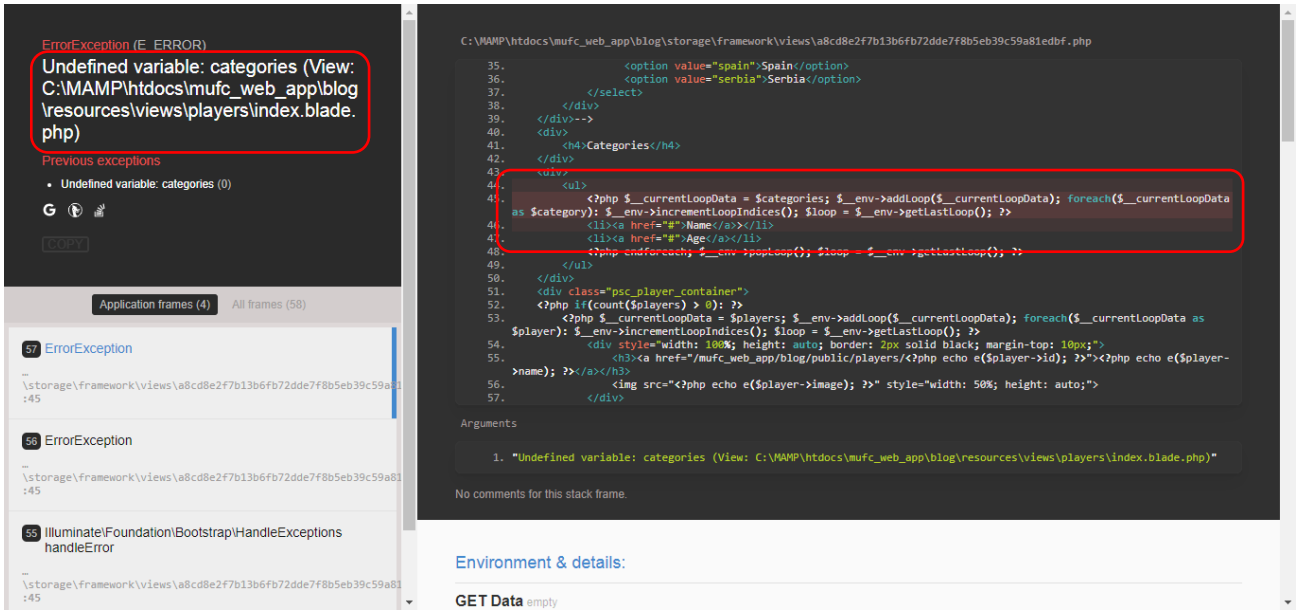


Adding 'return view' to the same Place in the 'Controller'

```
$players = Player::orderBy('name', 'asc')
    ->get();

return view('players.index')->with('
    players', $players);
```

However, after undertaking the process shown above, I then encountered another error, as displayed below:

The Error Experienced



After identifying that the 'variable' was the issue, I then ensured that this was an area resolved by removing the following 'variables':

Removing the 'Variables' and 'foreach' Loop

```
<div>
    <ul>
        @foreach ($categories as $category)
        <li><a href="#">Name</a>></li>
        <li><a href="#">Age</a></li>
        @endforeach
    </ul>
</div>
```

However, I then realised that in order for the links and shown players to work successfully, I needed to create another 'migration' to create a table named 'categories' in the database. I therefore undertook the following process, also adding a new 'controller' called 'CategoriesController':

Adding a new 'controller' and 'migration' through the Command Line Interface

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller CategoriesControlle
r --resource
Controller created successfully.

C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:model Category -m
Model created successfully.
Created Migration: 2019_03_16_101332_create_categories_table
```

The Data Fields to be Entered into the Table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('title');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

Executing 'php artisan migrate' in the Command Line Interface to Update the Database

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan migrate
Migrating: 2019_03_16_101332_create_categories_table
Migrated:  2019_03_16_101332_create_categories_table
```

The Table now Appeared with all Fields

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop
☐	2	title	varchar(191)	utf8mb4_unicode_ci		No	None			Change Drop
☐	3	created_at	timestamp			Yes	NULL			Change Drop
☐	4	updated_at	timestamp			Yes	NULL			Change Drop

After completing the previous process, I then added the nationalities to test this first, as seen below:

Adding each Nationality to the Table to Test

	id	title	created_at	updated_at
Edit Copy Delete	1	England	NULL	NULL
Edit Copy Delete	2	France	NULL	NULL

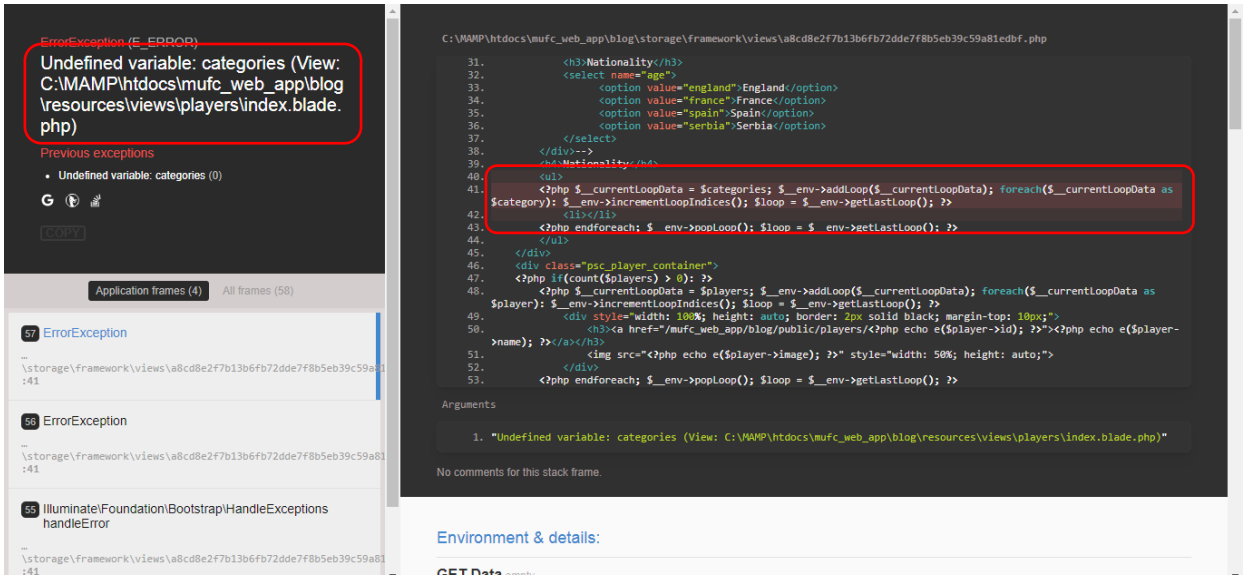
Whilst following the tutorial again, I encountered the following issue shown below:

The Current Code in the 'CategoriesController'

```
    public function index()
    {

        $players = Player::orderBy('name', 'asc')->get();
        $categories = Category::orderBy('title', 'asc')->get();

//          return view('players.index')->with('players', $players);
        return view('players.index')->with('players', $players)->with('categories', $categories);
/*          $players = Player::orderBy('name','desc')->get();
        return view('players.index')->with('players', $players);*/

    }
```

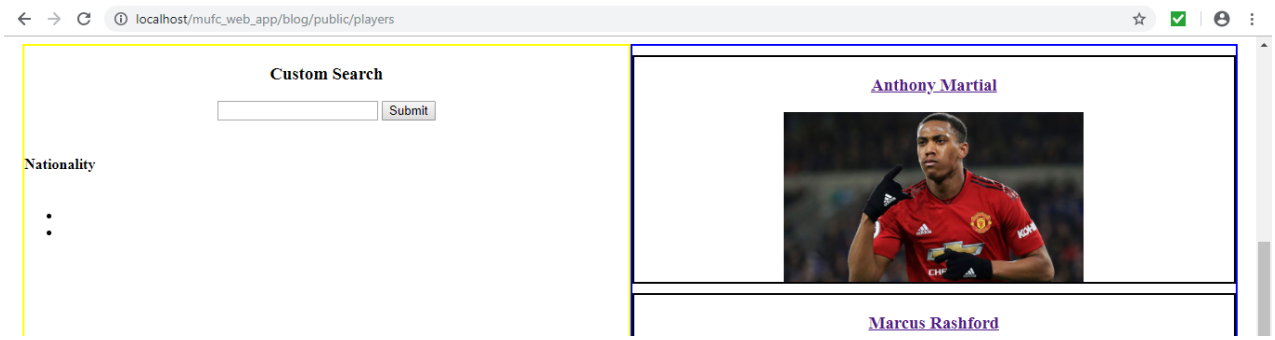
As is evident above, both the players and categories would be ordered in ascending order by name and title through 'orderBy', utilising 'get' to obtain each player and category. Furthermore, the view would then be returned with players ordered by their selected categories. In this case, this would be the nationalities. As stated before, I encountered the following error:

The Current Experienced Error



As the 'categories' 'variable' was the aspect causing the issue, I then reloaded the page without the 'categories' 'variable' to understand if this was true. Whilst undertaking this process, I then realised that this wasn't working because it was using the wrong 'URL' and I needed to navigate back to the original page before the search:

Loading the Page without the 'Variable'

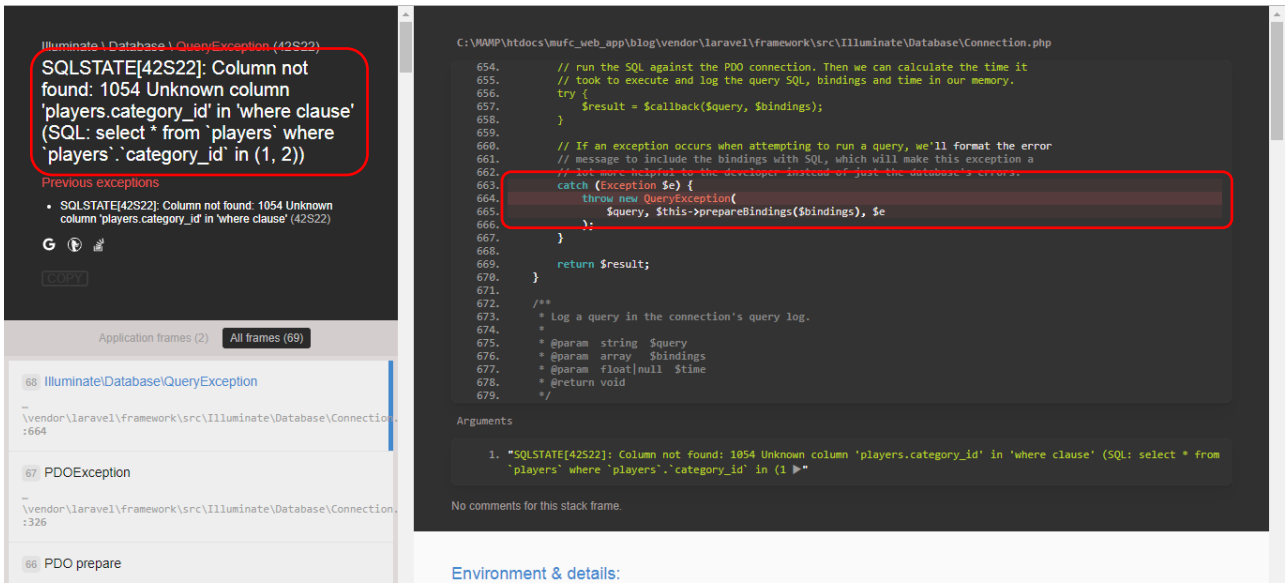


Whilst continuing further, I then encountered another issue regarding not being able to find a 'column'. This can be viewed below:

The Current Public Function within the 'PlayersController'

```
$players = Player::orderBy('name', 'asc')->get();
$categories = Category::with('players')->orderBy('
    title', 'asc')->get();
```

The Error Experienced Regarding a 'Column'



To resolve this issue, I then changed the 'column' name within the players table from 'nationality' to 'category_id' to allow for the 'column' to be found. I also then added a 'foreach' statement into the 'index.blade.php' file to allow for this filter to then show and successfully function:

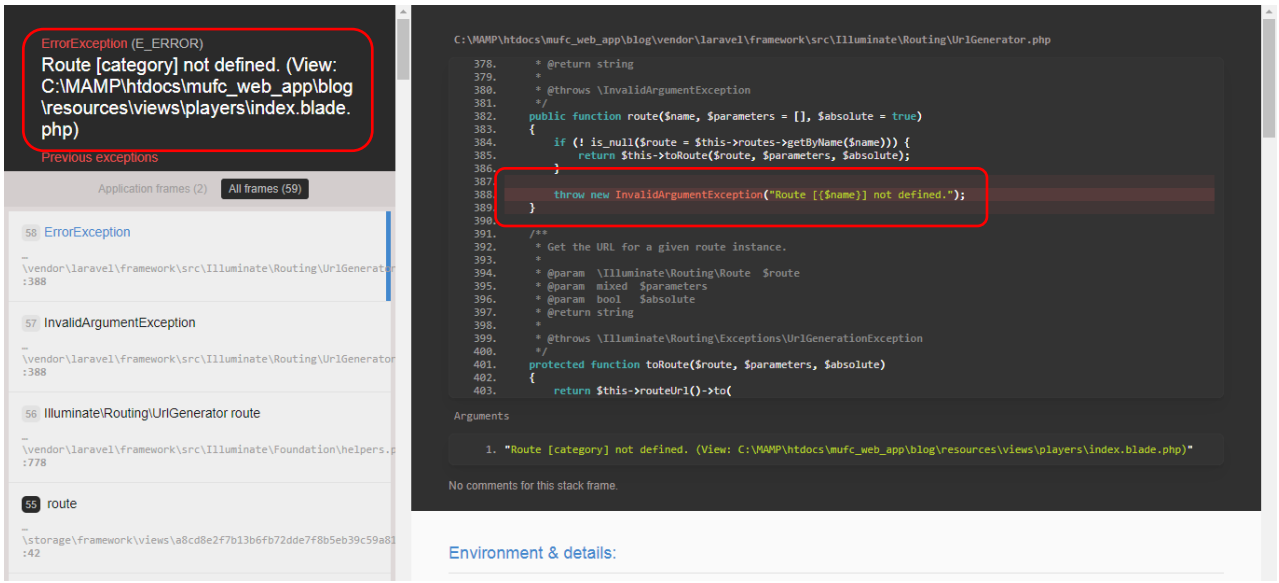
Changing the 'Column' Name within the Players Table

Adding the 'foreach' Statement into the 'index.blade.php' File

```
<h4>Nationality</h4>
<ul>
@foreach ($categories as $category)
    <a href="{{ route('category', $category->id)
    }}"><li>{{ $category->title }}</li></a>
@endforeach
</ul>
```

However, I then experienced another issue regarding the 'route' not being defined, as displayed below:

The Error Encountered Regarding the 'route' not Being Defined



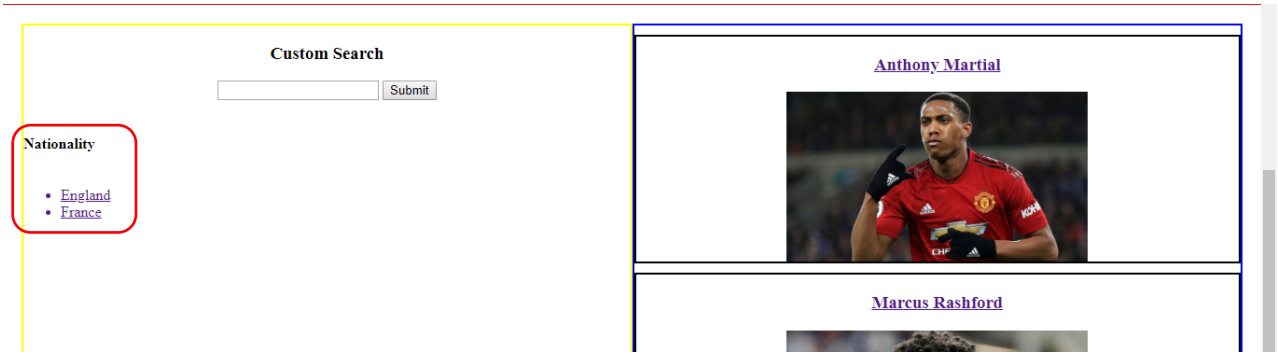
As shown on the tutorial I was following, I then added the 'route' to resolve this issue:

Adding the 'route' to the 'web.php' File

```
Route::get('/players/{player}', [
    'uses' => 'PlayersController@category',
    'as' => 'category'

]);
```

After undertaking the previous process, the page now displayed, as seen below:

The Page now Displayed Successfully with the Filters



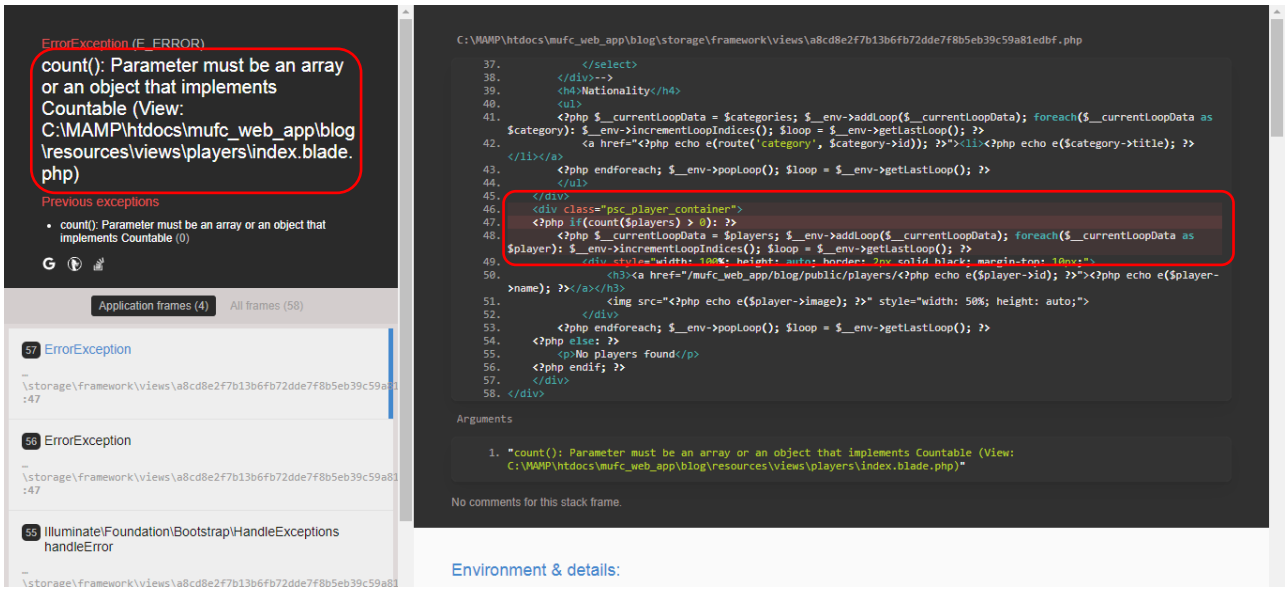
Whilst duplicating the public function of index to ensure that when selecting nationalities, the players would appear, I encountered the following error:

Duplicating the Function, Changing the name to 'category'

```
public function category($id)
{

    $categories = Category::with('players')->orderBy(
        'title', 'asc')->get();

    $players = Player::with('name')
        ->where('category_id', $id);

    return view("players.index", compact('players', '
        categories'));

}
```

As is evident above, 'compact' was utilised to return the two 'variables' to the view, allowing for all aspects to work and display. The error I encountered can be viewed below:

The Error Experienced

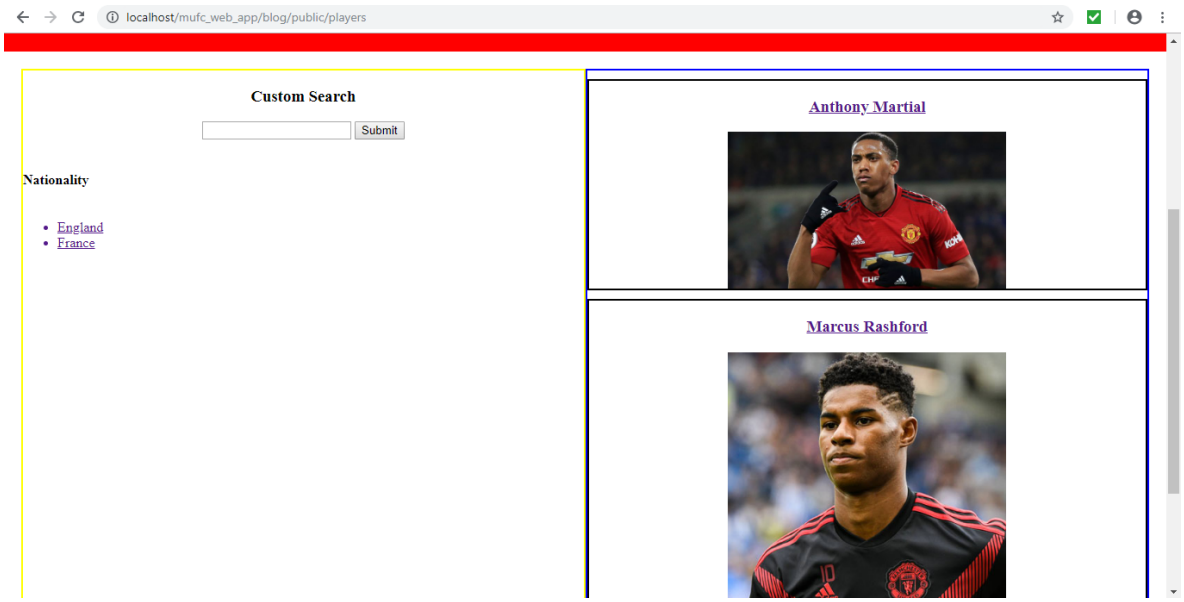


I then altered the code to include the 'get' aspect regarding the players variable and reloaded the page which now caused for no errors to appear:

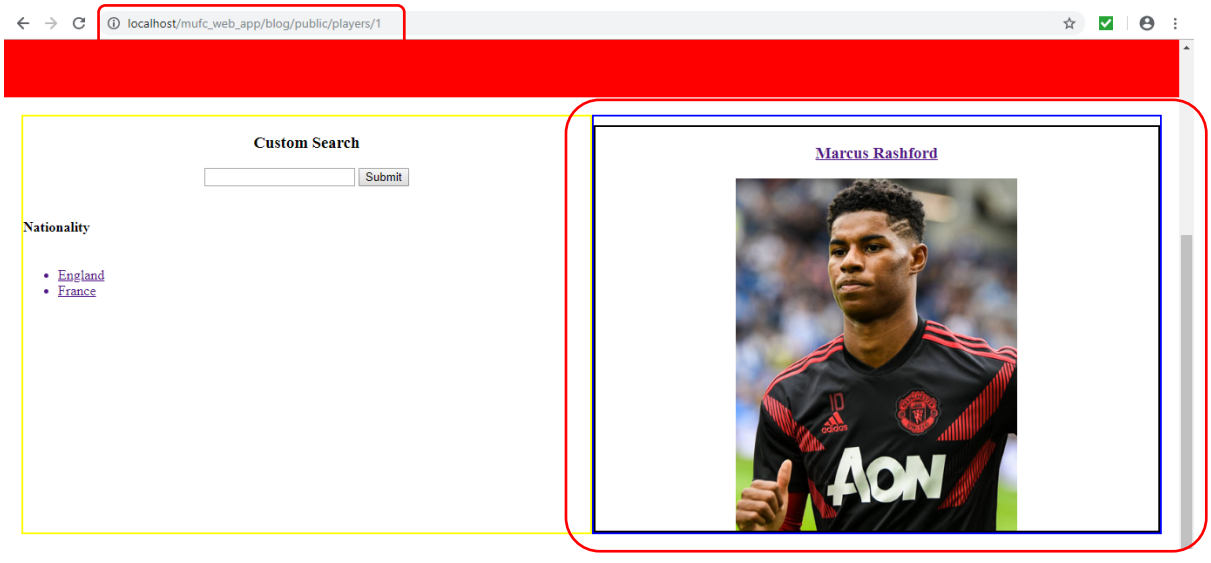
Altering the Code

```
public function category($id)
{

    $categories = Category::with('players')->orderBy(
        'title', 'asc')->get();

      $players = Player::with('player')
          ->where('category_id', $id);

    $players = Player::orderBy('name')->get()->where(
        'category_id', $id);

    return view("players.index", compact('players', '
        categories'));

}
```

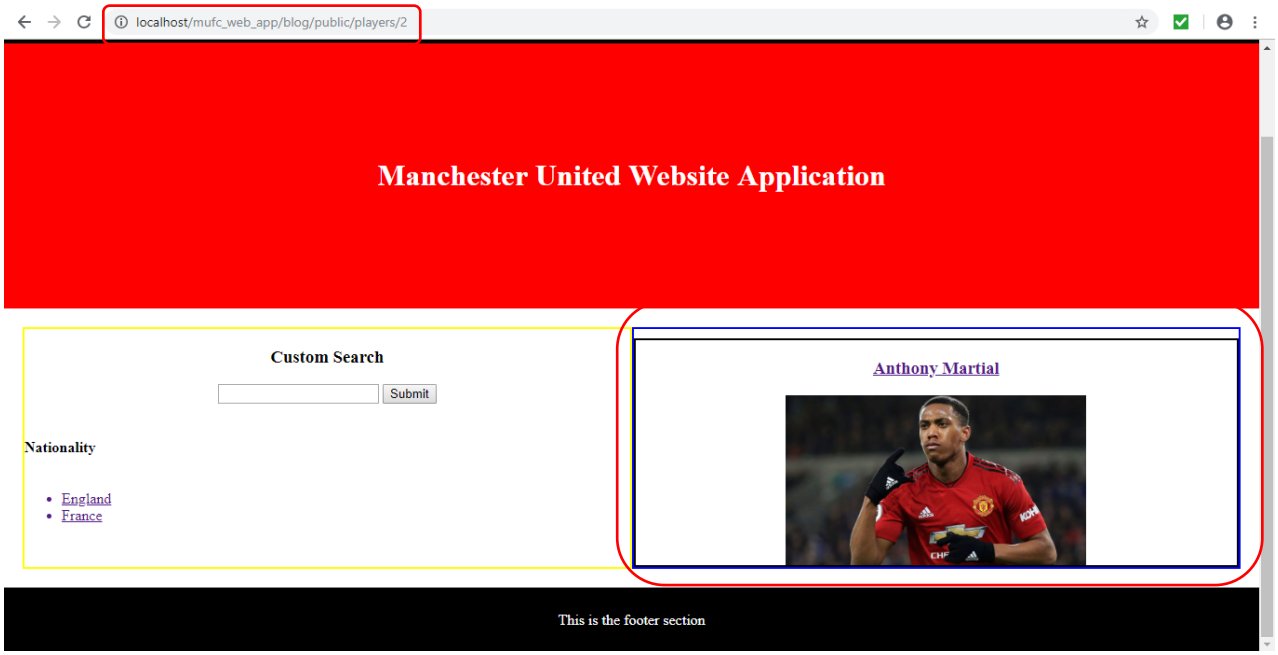
The Outcome – This now Successfully Worked



After Selecting the Nationality of England, the Player Appeared Relating to this



After Selecting the Nationality of France, the Player Appeared Relating to this



After successfully managing to allow the filter to work for the nationalities, I then undertook the same process again for the age filter:

Making the 'Model' and 'Controller' for the Age Aspect as well as Creating a 'Migration' (Not Displayed Below)

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:model Age -m
Model created successfully.
Created Migration: 2019_03_16_115511_create_ages_table

C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller AgesController --re
source
Controller created successfully.
```

The 'Migration' File – Adding the 'title' Aspect

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateAgesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('ages', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('title');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('ages');
    }
}
```

Adding the Following Code to the 'Model' File

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Age extends Model
{
    public function players() {
        return $this->hasMany(Player::class);
    }
}
```

Adding the Following Code to the 'Model' File Relating to the Players

```php
public function age() {
    return $this->belongsTo(Age::class);
}
```

Adding a 'route' to the 'web.php' File

```php
Route::get('/players/{player}', [
    'uses' => 'PlayersController@age',
    'as' => 'age'

]);
```

Integrating a 'foreach' Statement into the 'index.blade.php' File

```php
<h4>Age</h4>
<ul>
@foreach ($ages as $age)
    <a href="{{ route('age', $age->id) }}"><li>{{ $age->title }}</li></a>
@endforeach
</ul>
```

Executing 'php artisan migrate' in the Command Line Interface to Update the Database

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan migrate
Migrating: 2019_03_16_115511_create_ages_table
Migrated:  2019_03_16_115511_create_ages_table
```

Please note that after creating the table, I then entered the relevant data into the database as will be seen below:

Adding the 'age_id' 'Column' with 'Integers' into the Existing Players Table

age_id
3
2

The new Created Ages Table with all Relevant Data

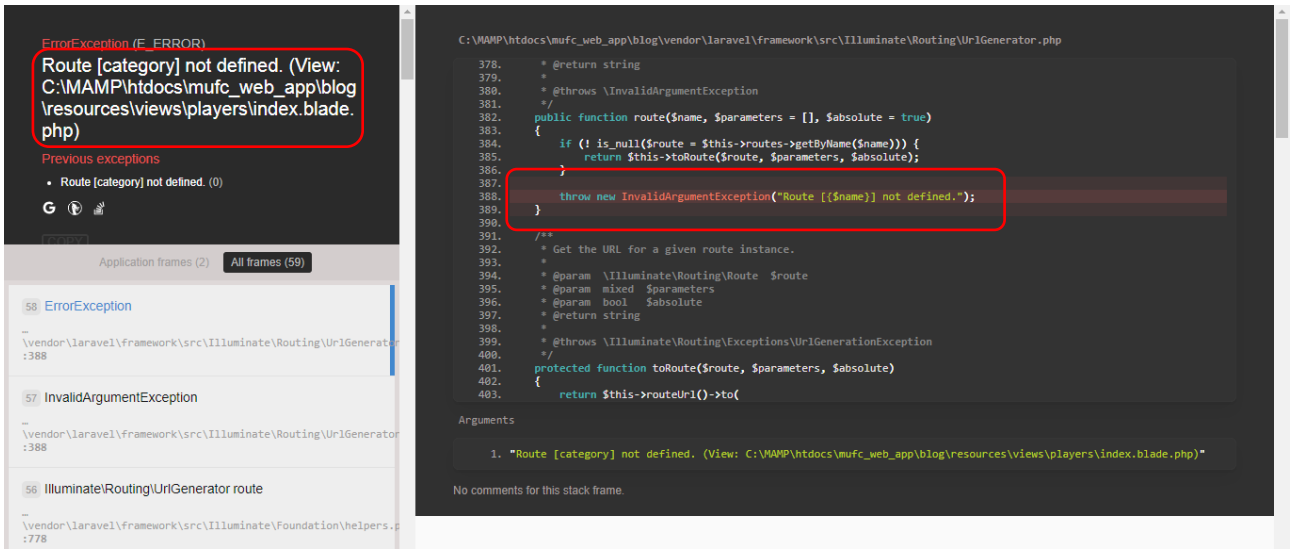
	id	title	created_at	updated_at
Edit Copy Delete	1	18-20	NULL	NULL
Edit Copy Delete	2	21-22	NULL	NULL
Edit Copy Delete	3	21-22	NULL	NULL
Edit Copy Delete	4	23-24	NULL	NULL
Edit Copy Delete	5	25-26	NULL	NULL

However, whilst trying to load page after the previous processes, I experienced the following issue where 'routes' were overwriting each other, even after implementing some new code as seen below:

Integrating new Code to Place each Filter Aspect into one 'route'

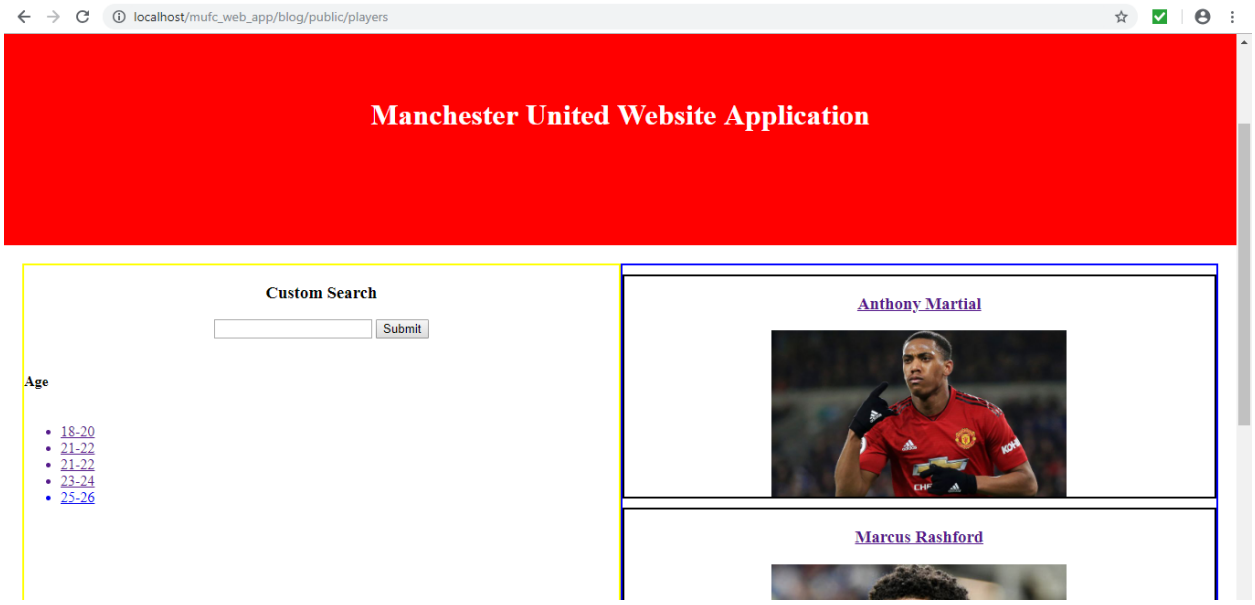
```
Route::match(['get'], '/players/{player}', [
    'uses' => 'PlayersController@category',
    'as' => 'category',
    'uses' => 'PlayersController@age',
    'as' => 'age'

]);
```

The Error with the 'routes' Overwriting each other

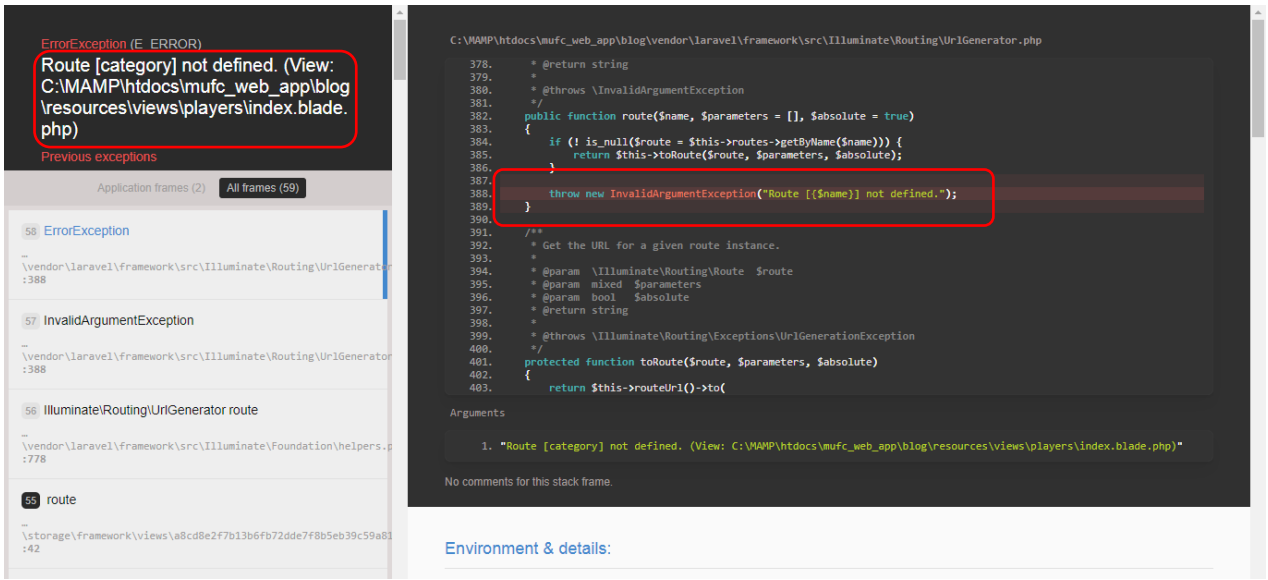


However, after removing the code for both the category route and age route, I realised there was a problem with the category route as will be seen below:

Removing the Category Route (Relating to Nationalities) – The Page Displayed



Removing the Age Route – The Error now Appeared



Following on from this, I then explored the category route further to identify the problem. I then removed the categories and ages variables from relevant parts within the controller. I believed removing those not relating to the public functions would help to resolve the issue. However, this had no effect on the outcome:

Removing the Relevant Variables from each Public Function

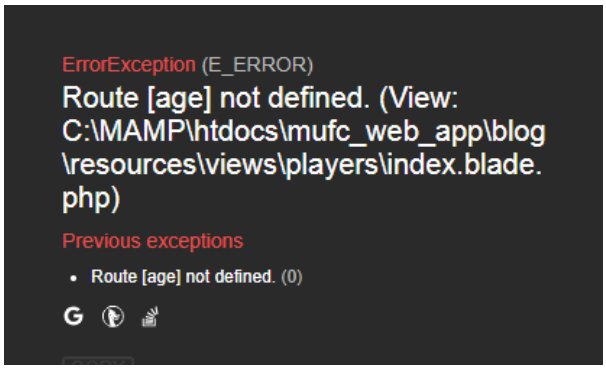
```
public function category($id)
{
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
      $players = Player::with('player')
          ->where('category_id', $id);
    $players = Player::orderBy('name')->get()->where('category_id', $id);
    return view("players.index", compact('players', 'categories'));
}
public function age($id)
{
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
      $players = Player::with('player')
          ->where('category_id', $id);
    $players = Player::orderBy('name')->get()->where('age_id', $id);
    return view("players.index", compact('players', 'ages'));
}
```

I then realised that the routes were actually overwriting each other, as initially thought. This was because when changing the position of each route, I now encountered the same error for the age route:

Changing the Position of each Route in the 'web.php' File

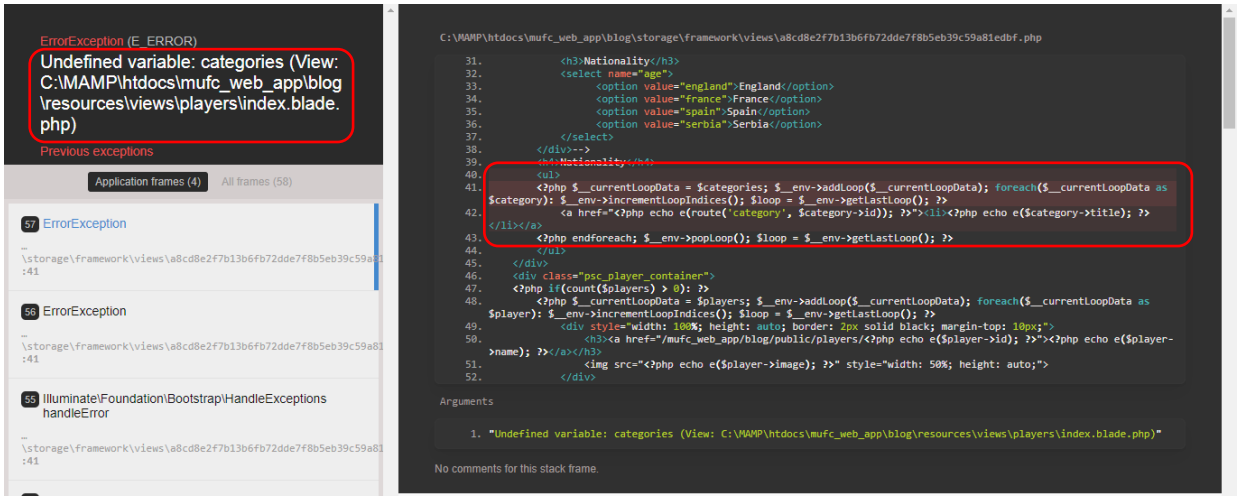
```
Route::match(['get'], '/players/{player}', [
    'uses' => 'PlayersController@age',
    'as' => 'age',
    'uses' => 'PlayersController@category',
    'as' => 'category'

]);
```

The Error now Appeared for the age Route



As I was currently struggling with the previous aspect, I therefore decided to resolve another encountered issue regarding the custom search which was now not functioning correctly when entering search terms. Whilst attempting to resolve this issue, I changed the 'URL' of the search route and form action as I believed this would help direct the user to the correct place:

The Error Experienced Whilst Searching



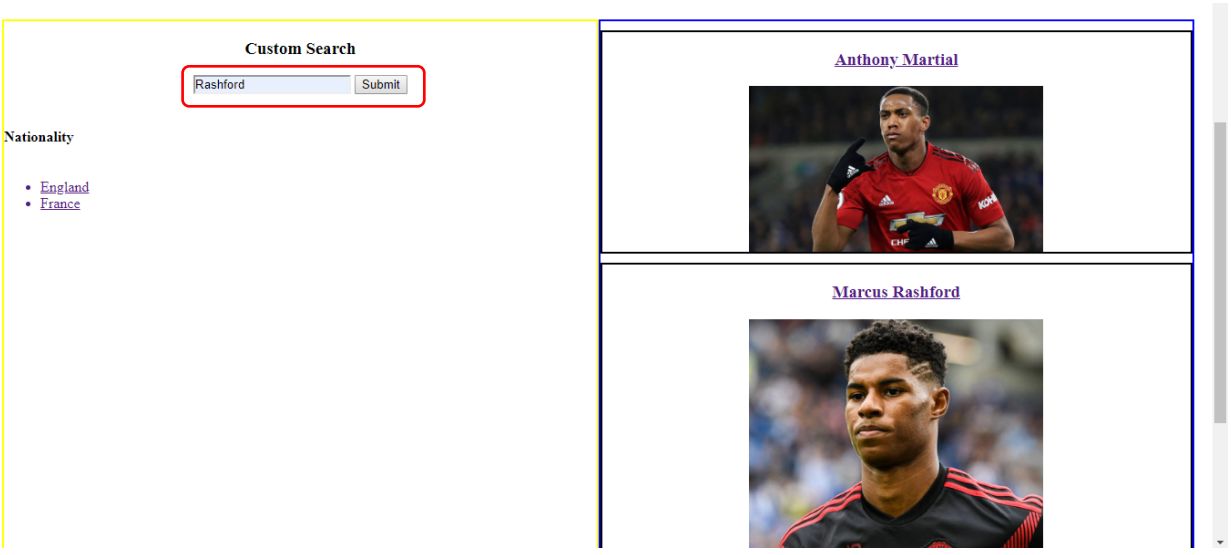
Modifying the 'URL' of Both the Form Action and Route

```
<form action="http://localhost/mufc_web_app/blog/public/players/search" method="post">
```

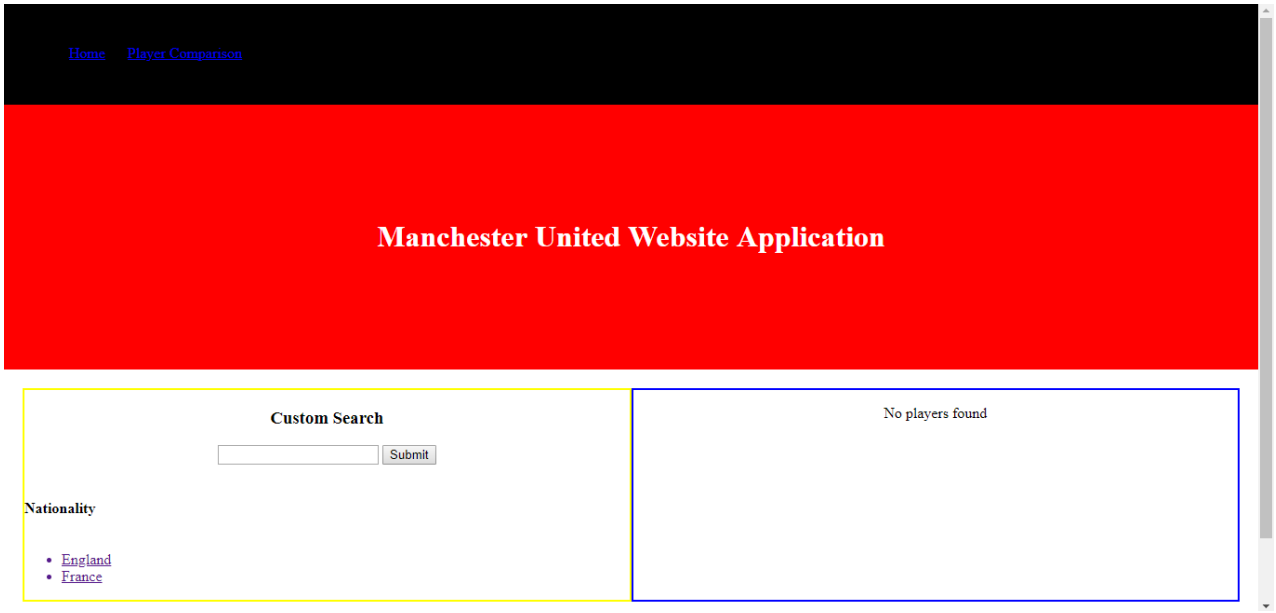
```
Route::any('http://localhost/mufc_web_app/blog/public/players/search', function(Input $input){
```

This didn't cause the error to appear but also caused no players to appear:

The Outcome – Whilst Searching



The Outcome – After Searching (No Players Appeared)



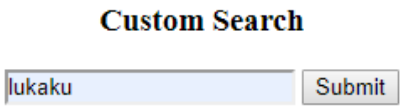
From undertaking research online, I managed to resolve the issue by applying the following:

Making Modifications to the 'web.php' File

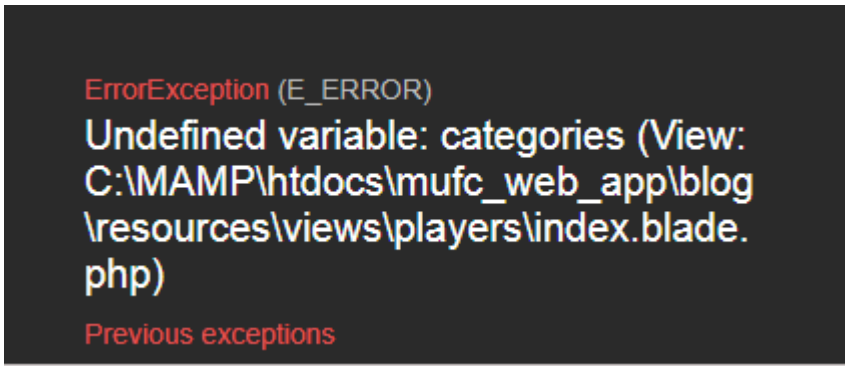
```php
<?php

use Illuminate\Support\Facades\Input;
use App\Player;
use App\Category;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

/*Route::get('/', function () {
    return view('welcome');
});*/

Route::get('/', 'PagesController@index');

//Route::resource('posts', 'PostsController');

Route::resource('players', 'PlayersController');

Route::get('/', function(){
    return view('welcome');
});

Route::any('/search', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
//    $categories = Category::with('players')->orderBy('title', 'asc')->get();
//    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    if(count($players) > 0){
        return view('players.index')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories', Category::orderBy('title', 'asc')->get());
//        return view('players.index')->with('ages', $ages);
//        return view('players.index')->with('categories', $categories);
    }
    else return view ('players.index')->withMessage('No Details found. Try to search again !');
});
```

However, now I needed to resolve the issue with the fact that I was receiving the following error after entering a term that wouldn't be able to find the data as this currently didn't exist in the database:

Entering a Player Name which Didn't Exist in the Database



The Outcome – An Error Occurred



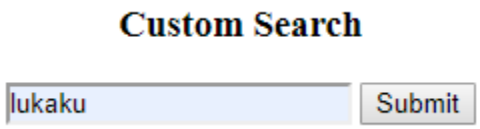
I then entered the same code as that utilised as before after realising that I needed to add the players aspect in otherwise the players variable couldn't be found. This was applied to the 'else' aspect as I was currently searching for a player which wouldn't be able to be found:

Applying the Same Code as Before to the 'else' Aspect in the 'web.php' File

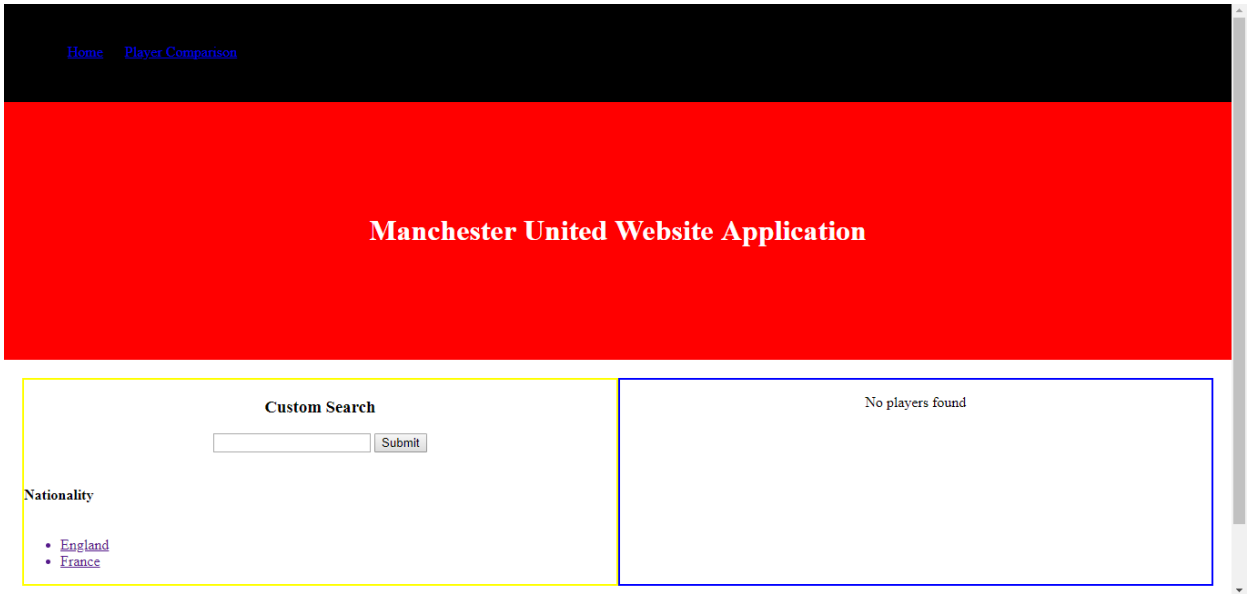
```
Route::any('/search', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
//    $categories = Category::with('players')->orderBy('title', 'asc')->get();
//    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    if(count($players) > 0){
        return view('players.index')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories', Category::orderBy('title', 'asc')->get());
//        return view('players.index')->with('ages', $ages);
//       return view('players.index')->with('categories', $categories);
    }
    else return view ('players.index')
        ->withMessage('No Details found. Try to search again !')
        ->with('players', $players)
        ->with('categories', Category::orderBy('title', 'asc')->get());
});
```

This then resolved the issue as now the page displayed with 'No players found':

Entering a Player which Didn't Exist in the Database



The Outcome – This now Successfully Worked



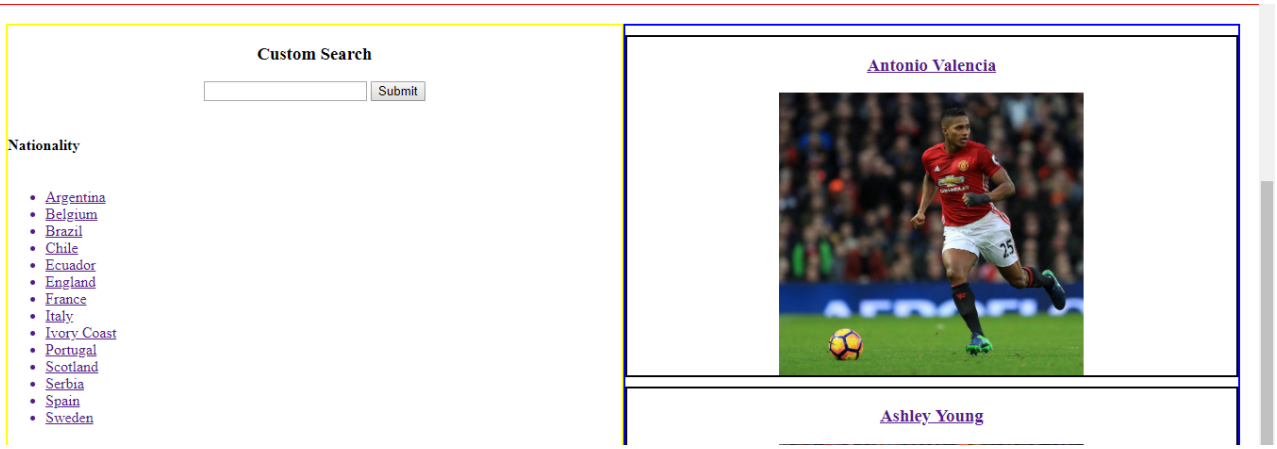
As now these issues had been resolved, I then continued with attempting to resolve the multiple filters issue. After I had integrated the same code from the 'PlayersController' into the 'CategoriesController' and had changed the route to include 'CategoriesController@category', I still experienced the same issue of the age route not being defined.

After experiencing the issues with the fact that the route could not be defined on the main page, I then added two new aspects relating to each individual filter. This then caused for the page to show with both filter sets. However, when selecting the age and nationality, this wasn't relating to the correct ages or nationalities for players but displayed all players instead as will be seen on the following page.

Modifying the 'web.php' to Assign Individual Routes to each Filter Set

```
Route::any('/players/{player}', [
    'uses' => 'PlayersController@category',
    'as' => 'category',
    'uses' => 'PlayersController@age',
    'as' => 'age'

]);

Route::any('/players/{category}', [
    'uses' => 'PlayersController@category',
    'as' => 'category',
]);

Route::any('/players/{age}', [
    'uses' => 'PlayersController@age',
    'as' => 'age'

]);
```

Selecting the 'Italy' Nationality to Test – This Displayed all Players

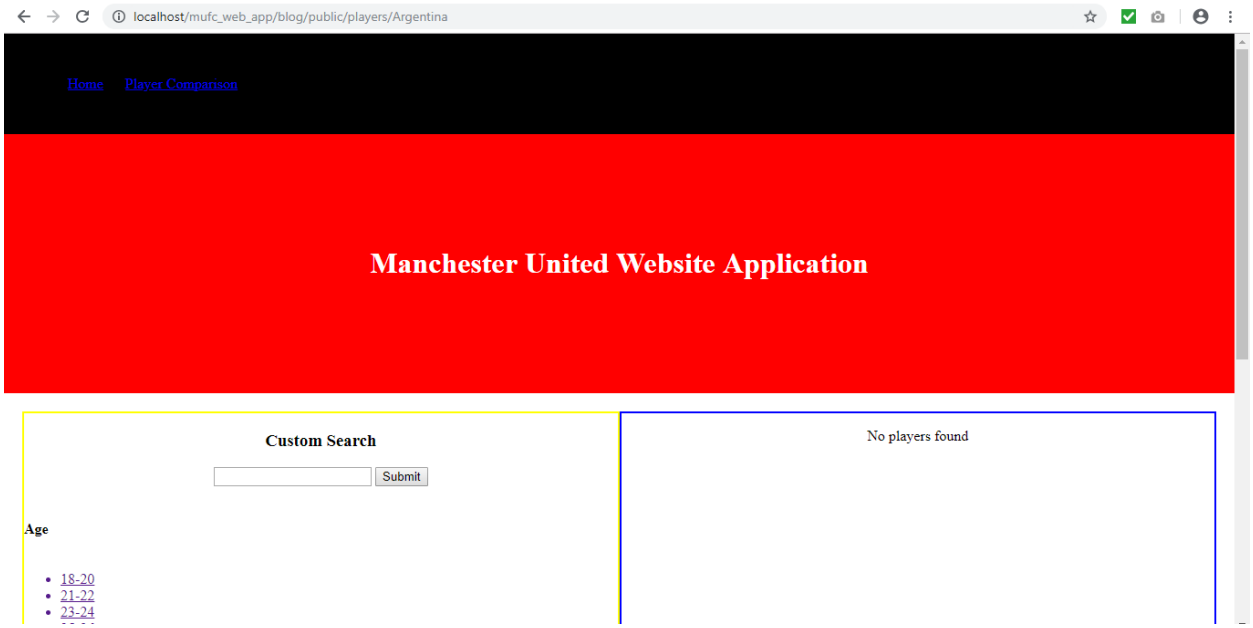


However, I then realised that when selecting the nationalities that this was relating to the age rather than the nationality. After realising that I could change the link to change to the title of the nationality, I then believed if I set the link to obtain the 'category_id' field, this would work in displaying the relevant players on the page. This process can be viewed below:

Obtaining the 'title' Aspect within the Link Initially

```
@foreach ($categories as $category)
    <a href="{{ route('category', $category->title) }}"><li>{{ $category->title }}</li></a>
@endforeach
```

The Outcome on the Page – This Displayed no Players After Selecting Argentina



Altering the 'index.blade.php' File to Collect the 'category_id' Instead within the Link

```
<h4>Nationality</h4>
<ul>
@foreach ($categories as $category)
@foreach ($players as $player)
    <a href="{{ route('player', $player->category_id) }}"><li>{{ $category->title }}</li></a>
@endforeach
@endforeach
</ul>
```

Adding a new Separate Route for the Players Aspect in the 'web.php' File

```
Route::any('/players/{player}', [
    'uses' => 'PlayersController@category',
    'as' => 'category',
    'uses' => 'PlayersController@age',
    'as' => 'age',
    'uses' => 'PlayersController@player',
    'as' => 'player'

]);

Route::any('/players/{player}', [
    'uses' => 'PlayersController@player',
    'as' => 'player'
]);

Route::any('/players/{category}', [
    'uses' => 'PlayersController@category',
    'as' => 'category'
]);

Route::any('/players/{age}', [
    'uses' => 'PlayersController@age',
    'as' => 'age'

]);
```

Something that wasn't mentioned before regarding the code above is that I thought placing two 'foreach' statements would help to allow for the nationality title to be displayed whilst also allowing for the 'category_id' to be collected from the players table in the database. This also meant I then applied the same process as before in the 'web.php' file to allow for the page to display.

The page did display, however, when selecting a nationality and age option, I experienced the following error:

The Error Encountered

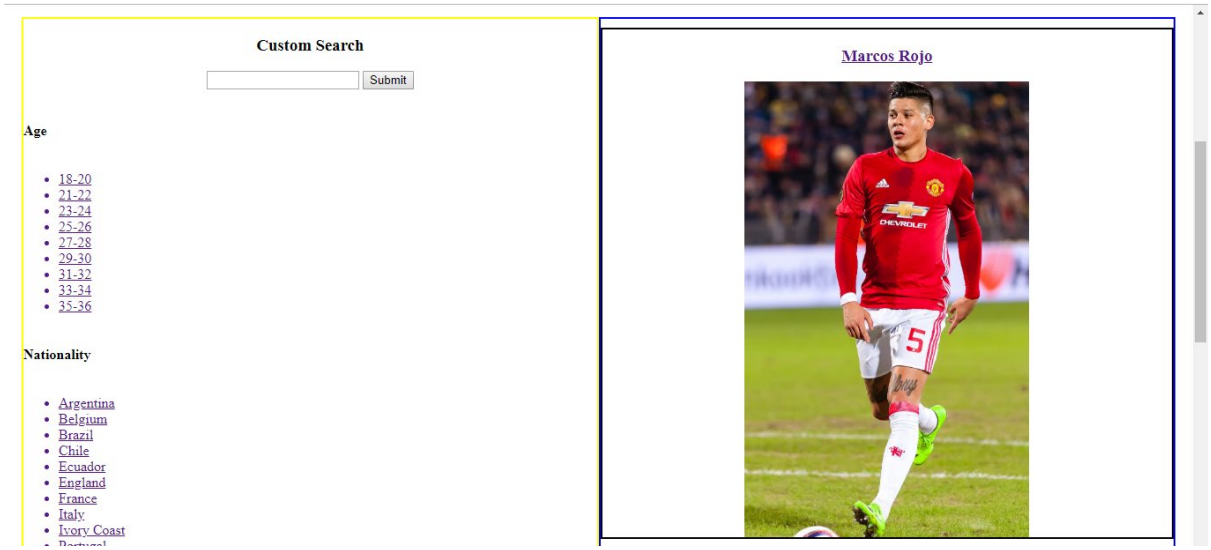
I then realised that I may have needed to create a new function called player so therefore I undertook the process of assigning the same function as that used for the category function, changing the name whilst also altering a few aspects:

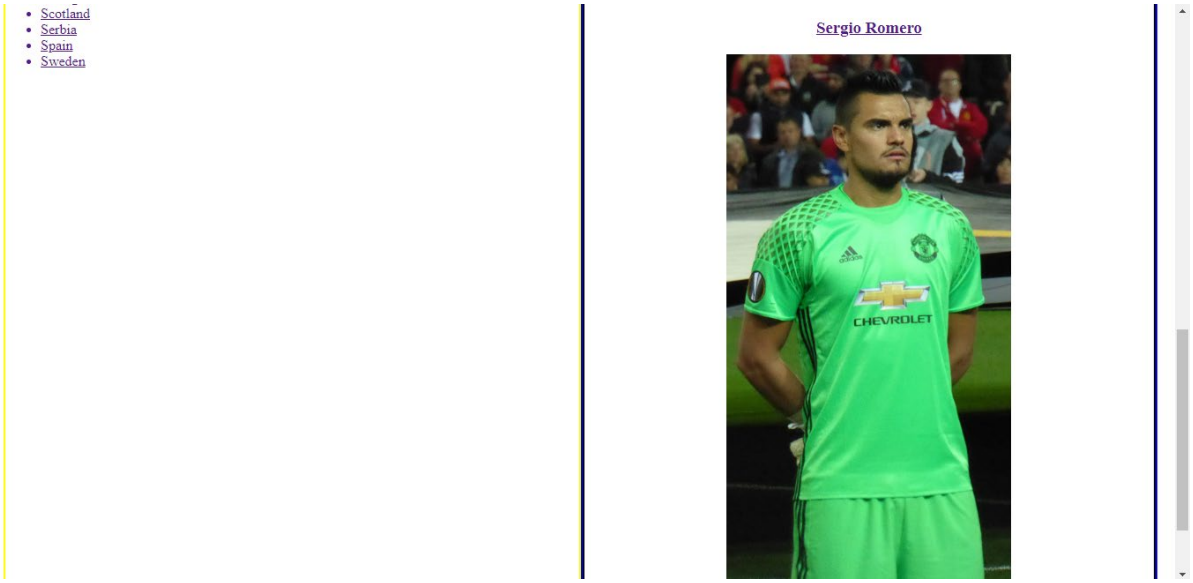
Creating the new Player Function in the 'PlayersController' File

```
public function player($id) {

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

      $players = Player::with('player')
          ->where('category_id', $id);

    $players = Player::orderBy('name')->get()->where('category_id', $id);

    return view('players.index', compact('players', 'categories', 'ages'));

}
```

After applying this, this then worked as selecting Argentina caused the correct players to appear on the page:

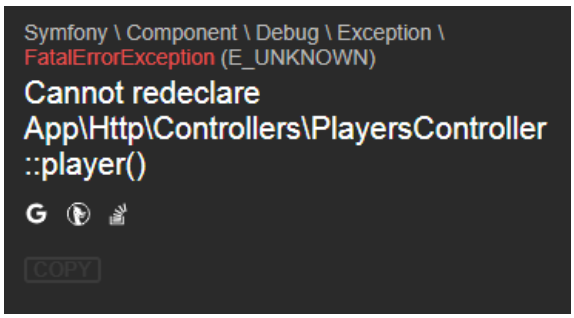
The Players with the Nationality of Argentina now Appeared on the Web Page





I then believed I could undertake the same process again regarding the age options. However, I experienced the following error due to utilising the same named public function twice:

The Error I Encountered



I then tried passing in variables as parameters, obtaining both the nationality and age through 'where'. This was because I thought I could allow for these two aspects to be inputted into one function. However, this was unsuccessful also as will be displayed below:

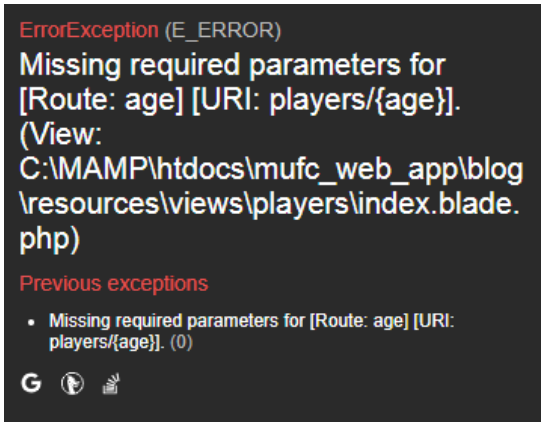
Passing in Variables as Parameters in the Controller

```
public function player($id, $id2) {

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

      $players = Player::with('player')
          ->where('category_id', $id);

    $players = Player::orderBy('name')->get()
    ->where('category_id', $id)
    ->where('age_id', $id2);

    return view('players.index', compact('players', 'categories', 'ages'));

}
```

Adding 'foreach' Statements for both the Nationality and Age Aspects

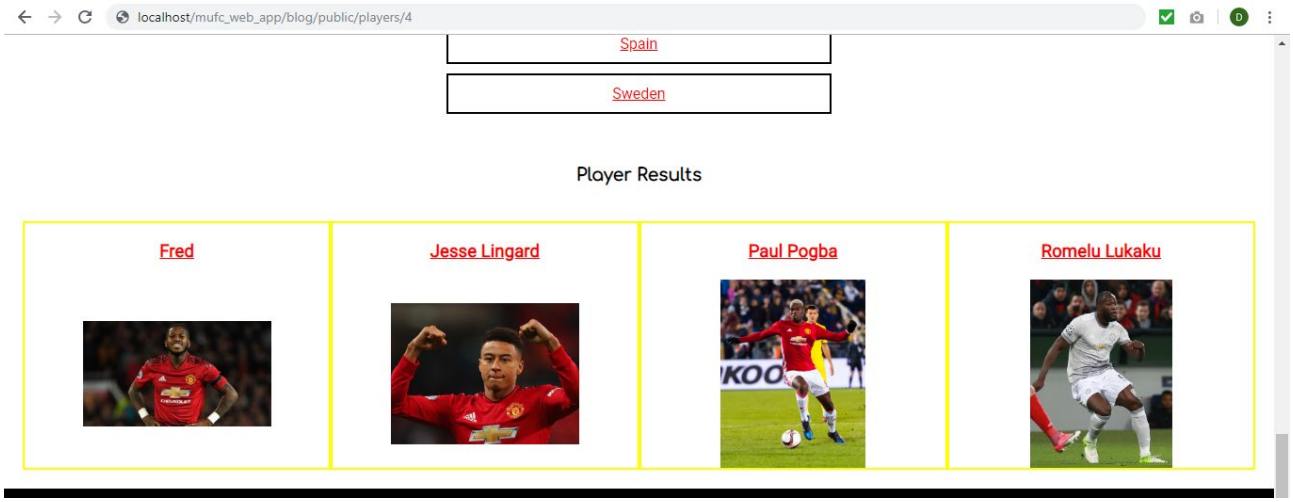
```
<h4>Age</h4>
<ul>
@foreach ($ages as $age)
    <a href="{{ route('age', $age->id2) }}"><li>{{ $age->title }}</li></a>
@endforeach
</ul>

<h4>Nationality</h4>
<ul>
@foreach ($categories as $category)
    <a href="{{ route('category', $category->id) }}"><li>{{ $category->title }}</li></a>
@endforeach
</ul>
```

The Error which Appeared

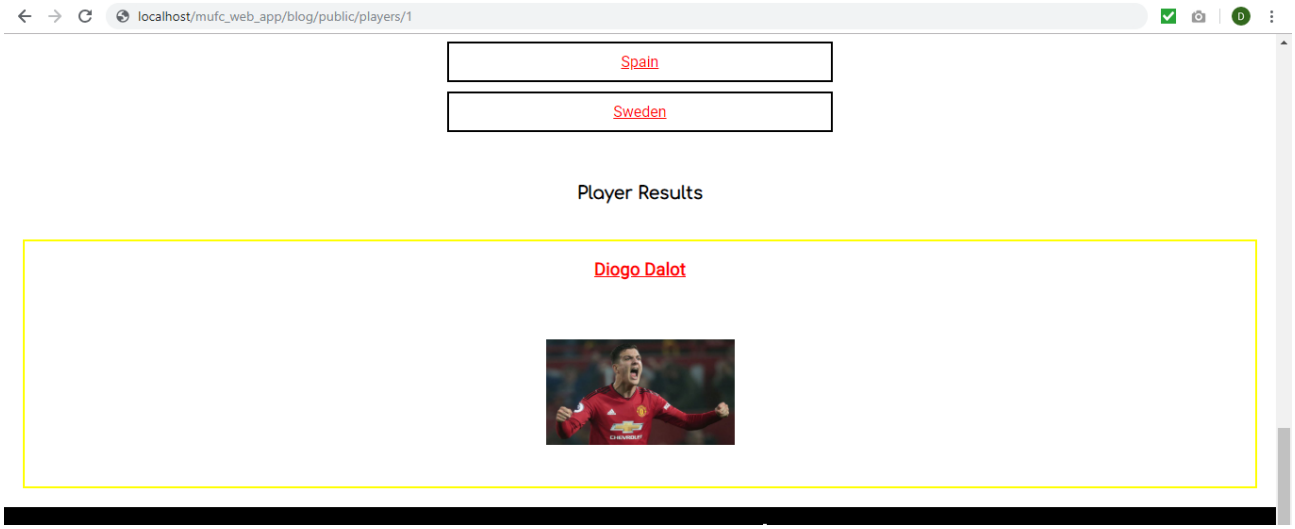


I then reverted back to what I had originally. To reiterate the issue I was experiencing at this current stage, when selecting an age filter, the correct players appeared on the screen. However, when selecting a nationality filter, the players were appearing that were related to a specific age filter and not the nationality filter itself. Examples of this issue can be viewed again below. Please note that the images supplied below are of the application at a later stage which is why the appearance is different:

Selecting Players with the Nationality of Argentina – This Displayed Players with the Ages 25-26 Years

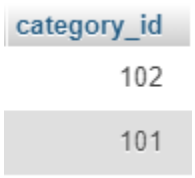


Selecting Players with the Nationality of England – This Displayed Players with the Ages 18-20 Years



One aspect I tried changing, because I thought that the 'IDs' were conflicting with each other for both the nationalities and ages, was changing the 'IDs' for nationalities in the database as seen with examples below:

Examples of Changing the 'IDs' of the Nationalities in the Database 'Players' Table



However, this proved to not alter the outcome and the same issue occurred as displayed before. After attempting many different aspects, I then decided to seek lecturer advice. I was informed that I shouldn't have been using one controller for both filter sets. As a result of this, I was then informed to upload this to 'GitHub' for the lecturer to assist.

After focusing elsewhere, I then decided to return to this at a later date. At this stage, I had the following code which was causing the previously explained issue. Please note I had altered this several times to try to the resolve the issue which is why the code appeared untidy:

The Relevant Code in the 'index.blade.php' File

```
        <h4>Age</h4>
        <ul>
        @foreach ($ages as $age)
            <a href="{{ route('age', $age->id) }}"><li>{{ $age->title }}</li></a>
        @endforeach
        </ul>

        <h4>Nationality</h4>
        <ul>
        @foreach ($categories as $category)
            <a href="{{ route('category', $category->id) }}"><li>{{ $category->title }}</li></a>
        @endforeach
        </ul>

    </div>
    <div class="psc_player_container">
    @if(count($players) > 0)
        @foreach($players as $player)
            <div style="width: 100%; height: auto; border: 2px solid black; margin-top: 10px;">
                <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}" target="_blank">{{$player->name}}</a></h3>
                <img src="{{$player->image}}" style="width: 50%; height: auto;">
            </div>
        @endforeach
    @else
        <p>No players found</p>
    @endif
    </div>
</div>
```

As is evident above, I was utilising 'foreach' statements for both the age and nationality aspects which was supposed to locate the user to the correct players through the 'route' aspect, using the 'ID' of the selected filter. This would then return the players regarding the players section which is the last highlighted section above. Each player would be returned with their name and image which would then navigate the user to a separate page regarding each player after selecting the provided link. This, however, also didn't work at this current stage as this displayed the same page but with a message of no players being found instead.

The Relevant Code in the PlayersController.php File

```php
public function index()
{

    $players = Player::orderBy('name', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();

      return view('players.index')->with('players', $players);
    return view('players.index', compact('players', 'categories', 'ages'));
      $players = Player::orderBy('name','desc')->get();
    return view('players.index')->with('players', $players);*/

}

public function age($id)
{

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

      $players = Player::with('player')
          ->where('category_id', $id);

    $players = Player::orderBy('name')->get()->where('age_id', $id);

    return view('players.index', compact('players', 'categories', 'ages'));

}

public function category($test)
{

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

      $players = Player::with('player')
          ->where('category_id', $id);

    $players = Player::orderBy('name')->get()->where('nationality', $test);

    return view('players.index', compact('players', 'categories', 'ages'));

}
```

```php
public function playerNationality($id) {

    $players = Player::orderBy('name', 'asc')->get();

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

      $players = Player::with('player')
          ->where('category_id', $id);

    $players = Player::orderBy('name')->get()
    ->where('category_id', $id);

    return view('players.index', compact('players', 'categories', 'ages'));

}
public function playerAge($id) {

    $players = Player::orderBy('name', 'asc')->get();

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

      $players = Player::with('player')
          ->where('category_id', $id);

    $players = Player::orderBy('name')->get()
    ->where('age_id', $id);

    return view('players.index', compact('players', 'categories', 'ages'));

}
```

As can be seen above, at this stage I had attempted to integrate two more public functions to relate to each filter type. These were called 'playerNationality' and 'playerAge'. Within each of these, I tried to order each player relating to the correct filter by utilising 'where'. Please also note, before this, I had tried to pass something other than 'id' into the public function of 'category', which was 'test', as I thought that applying the same 'id' aspect to two public functions was causing the issue explained before. However, neither of these methods had worked successfully.

The Relevant Code in the Player.php Model File

```php
public function category() {
    return $this->belongsTo(Category::class);
}

public function age() {
    return $this->belongsTo(Age::class);
}
```

The Relevant Code in the Age.php Model File

```php
class Age extends Model
{
    public function players() {
        return $this->hasMany(Player::class);
    }
}
```

The Relevant Code in the Category.php Model File (This Related to the Nationalities)

```php
class Category extends Model
{
    public function players() {
        return $this->hasMany(Player::class);
    }
}
```

As would have been evident above, the 'Player' model contained both aspects for the nationalities and ages regarding the public functions with each of the filter models being assigned to the public function of 'players'. This would allow these functions to execute throughout the application.

The Routes File (web.php) – The Custom Search Section

```
Route::any('/search', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
//    $categories = Category::with('players')->orderBy('title', 'asc')->get();
//    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    if(count($players) > 0){
        return view('players.index')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get());
//        return view('players.index')->with('ages', $ages);
//       return view('players.index')->with('categories', $categories);
    }
    else return view ('players.index')
        ->withMessage('No Details found. Try to search again !')
        ->with('players', $players)
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get());
});
```

As is evident above, this route would allow for collecting terms entered by users into the custom search box on the page, returning either players matching these terms or no players if none existed.

The Routes File (web.php) – The Different Filters Section

```
Route::any('/players/{player}', [
    'uses' => 'PlayersController@category',
    'as' => 'category',
    'uses' => 'PlayersController@age',
    'as' => 'age',
    'uses' => 'PlayersController@playerNationality',
    'as' => 'player',
    'uses' => 'PlayersController@playerAge',
    'as' => 'player',
    'uses' => 'PlayersController@show',
    'as' => 'show',

]);

Route::any('/players/{player}', [
    'uses' => 'PlayersController@playerNationality',
    'as' => 'player'
]);

Route::any('/players/{player}', [
    'uses' => 'PlayersController@playerAge',
    'as' => 'player'
]);

Route::any('/players/{category}', [
    'uses' => 'PlayersController@category',
    'as' => 'category'
]);

Route::any('/players/{age}', [
    'uses' => 'PlayersController@age',
    'as' => 'age'

]);

Route::any('/players/{show}', [
    'uses' => 'PlayersController@show',
    'as' => 'show'

]);
```

As is evident above, I had tried to assign the two new created public functions to both the routes relating to each filter set. I had also assigned these to the route relating to the players. Furthermore,

to attempt to resolve the issue with the show function not working properly when selecting a player name, I therefore created a route for this and also assigned this to the players route.

With regards to each public function relating to the different filter sets, I added these to the other controllers relating to the filter sets, removing the public functions not necessary and also changing the routes. This process can be viewed below:

Assigning each Public Function to their Relevant Controllers and Removing Other Code (CategoriesController.php File)

```
public function index()
{

    $players = Player::orderBy('name', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    return view('players.index', compact('players', 'categories', 'ages', 'positions'));

}
public function category($id)
{

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    $players = Player::orderBy('name')->get()->where('category_id', $id);

    return view('players.index', compact('players', 'categories', 'ages', 'positions'));

}
```

Assigning each Public Function to their Relevant Controllers and Removing Other Code (AgesController.php File)

```
public function index()
{

    $players = Player::orderBy('name', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    return view('players.index', compact('players', 'categories', 'ages'));

}
public function age($id)
{

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    $players = Player::orderBy('name')->get()->where('age_id', $id);

    return view('players.index', compact('players', 'categories', 'ages', 'positions'));

}
```

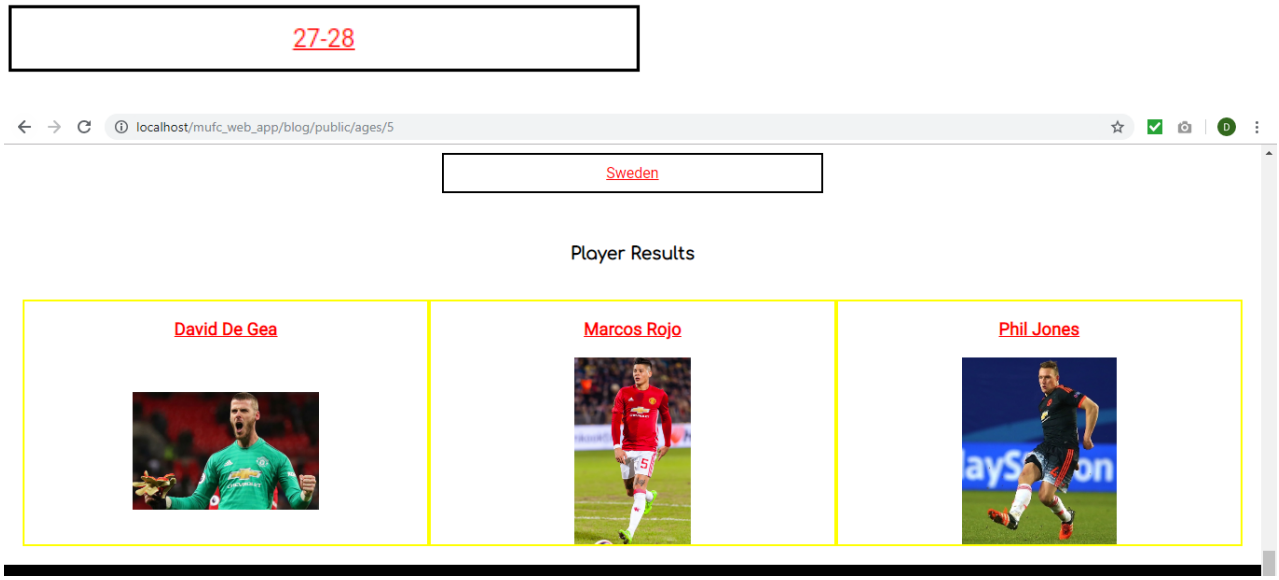
Changing the Routes in the 'web.php' File

```
Route::any('/ages/{age}', [
    'uses' => 'AgesController@age',
    'as' => 'age'

]);

Route::any('/categories/{category}', [
    'uses' => 'CategoriesController@category',
    'as' => 'category'
]);
```

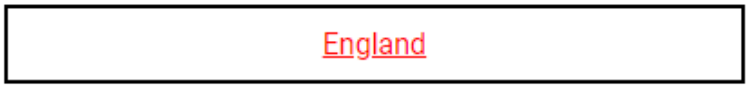
As is evident above with the routes, I removed the individual route for the players similar to the ones displayed above. I also changed the 'URL' to include the aspects relevant to each filter which would mean that the 'ID' would relate to one filter set only. The 'uses' was changed to relate to the relevant controllers as the public functions had now been moved.

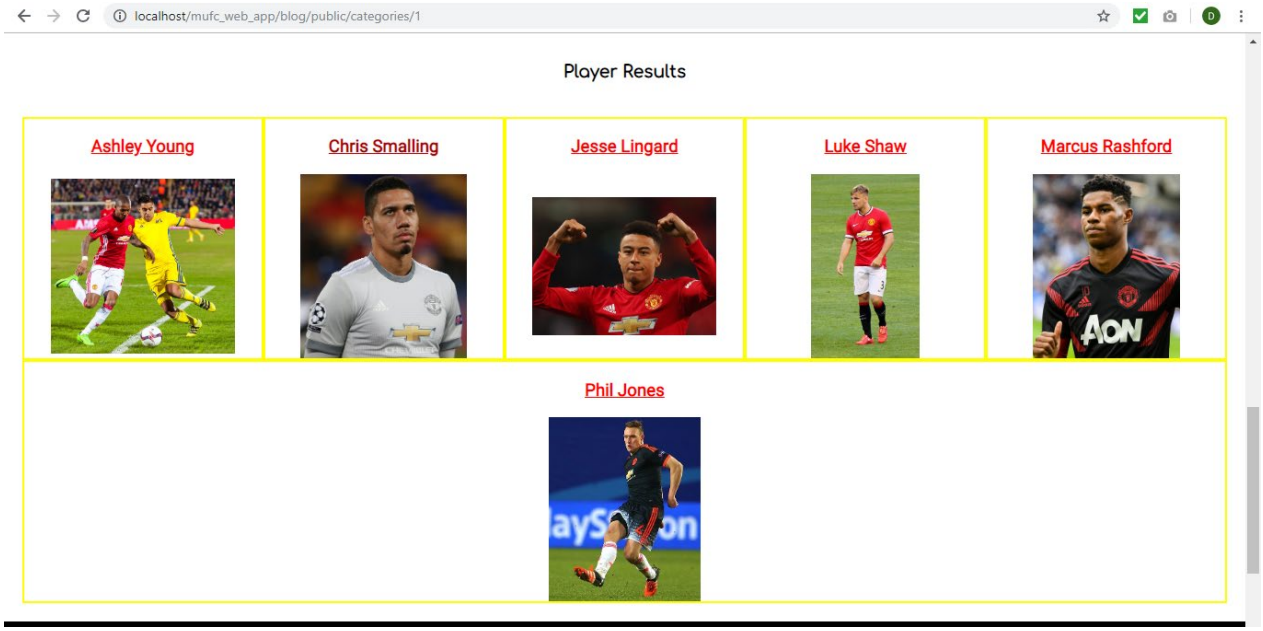
The changes shown above then helped to resolve the issue on the page and also helped to resolve the issue with selecting a player name, displaying the players on separate pages now. This is evident below. Please note that the images below were captured at a later date where the appearance had changed:

Selecting an Age Filter – This Still Worked

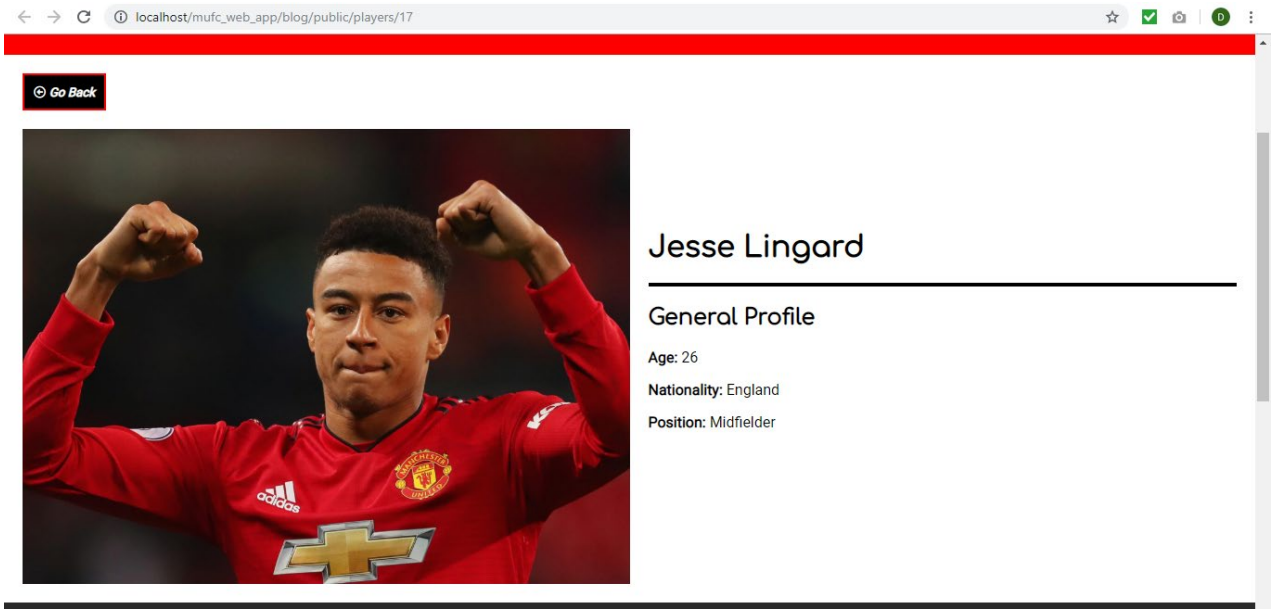


Selecting a Nationality Filter – This now Worked Successfully





Selecting to View an Individual Player – This Now Worked Again



My thought process for applying these changes to resolve the issues were inspired by the fact that I was informed I was using one controller to control everything and that having individual controllers would have been better.

After managing to resolve the previous issue, this now meant I could then make another filter set relating to positions. I then made this other set of filters for the positions relating to goalkeeper, defender, midfielder and forward positions. To begin, I created a new controller and model to relate to the positions:

Creating the PositionsController in the 'Command Line Interface'

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller PositionsController
 --resource
Controller created successfully.
```

Having Created the Model for the Positions Filter Set (Adding 'title' Column)

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePositionsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('positions', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('title');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('positions');
    }
}
```

Executing 'php artisan migrate' to Update the Database with the New Table

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan migrate
Migrating: 2019_04_10_121110_create_positions_table
Migrated:  2019_04_10_121110_create_positions_table
```

After the previous process, I then added each position to the new table as seen below:

Adding the New Values to the Table in the Database

☐	Edit	Copy	Delete	1	Goalkeeper	NULL	NULL
☐	Edit	Copy	Delete	2	Defender	NULL	NULL
☐	Edit	Copy	Delete	3	Midfielder	NULL	NULL
☐	Edit	Copy	Delete	4	Forward	NULL	NULL

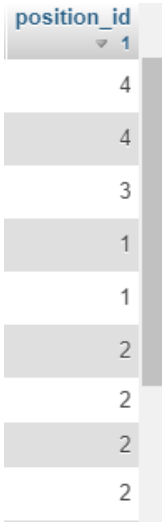
After completing the previous process, I then made changes to the controller relating to the positions filter set as well as the other controllers to include the new positions aspect:

Altering the 'PositionsController.php' File

```php
use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;

class PositionsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {

        $players = Player::orderBy('name', 'asc')->get();
        $categories = Category::with('players')->orderBy('title', 'asc')->get();
        $ages = Age::with('players')->orderBy('title', 'asc')->get();
        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }

    public function position($id)
    {

        $ages = Age::with('players')->orderBy('title', 'asc')->get();

        $categories = Category::with('players')->orderBy('title', 'asc')->get();

        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        $players = Player::orderBy('name')->get()->where('position_id', $id);

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }
```

As is evident above, I integrated similar code as before to return the 'index.blade.php' file with all filter sets through the public function of 'index'. However, with regards to adding the specific public function for this controller, I altered the code highlighted to obtain players by 'position_id' as opposed to others seen in other controllers previously. This would specify players by position when selecting a certain filter. This column was added to the 'players' table in the database as seen on the following page with a few examples of specifying players by position 'ID'.

Adding the 'position_id' Column to the Table in the Database



position_id ▼ 1
4
4
3
1
1
2
2
2
2

Altering the Other Controllers Example ('AgesController.php' File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;

class AgesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {

        $players = Player::orderBy('name', 'asc')->get();
        $categories = Category::with('players')->orderBy('title', 'asc')->get();
        $ages = Age::with('players')->orderBy('title', 'asc')->get();
        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        return view('players.index', compact('players', 'categories', 'ages'));

    }

    public function age($id)
    {

        $ages = Age::with('players')->orderBy('title', 'asc')->get();

        $categories = Category::with('players')->orderBy('title', 'asc')->get();

        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        $players = Player::orderBy('name')->get()->where('age_id', $id);

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }
```

As is evident above, I included the variables for the positions aspect so that the filters would be returned on the page for this section. Please note, I did add 'positions' to the 'compact' element at a later date, although not displayed here. Furthermore, 'use App' was integrated to allow for the positions aspect to work.

Furthermore, within the model, I added similar code as before to allow the public function of 'players' to successfully work within the new 'PositionsController'. Similar code was also added to the model regarding the players to allow the public function of 'position' to work:

Altering the Model File Relating to the 'positions' Aspect

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Age extends Model
{
    public function players() {
        return $this->hasMany(Player::class);
    }
}
```

Altering the Model File Relating to the 'players' Aspect

```php
public function position() {
    return $this->belongsTo(Position::class);
}
```

Following on from this, I then modified the routes in the 'web.php' file to include the position filter set. This process can be seen below:

Utilising 'use App' to Allow for the 'positions' Aspect to be Found

```php
use Illuminate\Support\Facades\Input;
use App\Player;
use App\Category;
use App\Age;
use App\Position;
```

Allowing the Position Filter Set to be Returned with the Other Filter Sets Regarding the Custom Search as well as Including an Individual Route for the Position Filter Set, as Explained Before with Other Routes

```
Route::get('/', 'PagesController@index');

//Route::resource('posts', 'PostsController');

Route::resource('players', 'PlayersController');

Route::get('/', function(){
    return view('welcome');
});

Route::any('/search', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($players) > 0){
        return view('players.index')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get())
        ->with('positions', Position::orderBy('title', 'asc')->get());
    }
    else return view ('players.index')
        ->withMessage('No Details found. Try to search again !')
        ->with('players', $players)
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get());
        ->with('positions', Position::orderBy('title', 'asc')->get());
});

Route::any('/ages/{age}', [
    'uses' => 'AgesController@age',
    'as' => 'age'

]);

Route::any('/categories/{category}', [
    'uses' => 'CategoriesController@category',
    'as' => 'category'
]);

Route::any('/categories/{position}', [
    'uses' => 'PositionsController@position',
    'as' => 'position'
]);
```

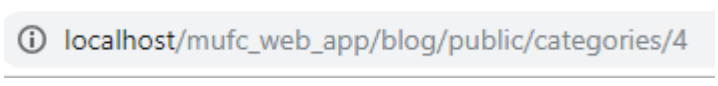
I then realised that I needed to remove one of the semicolons within the 'else return view' section of the route as this caused an issue. After this, I then added a 'foreach' statement to the 'index.blade.php' file as was done with the other filter sets shown before. This would allow for each category to be returned with a link to select in order to filter players by different positions.

Applying the 'foreach' Statement to the 'index.blade.php' File

```
<h4>Position</h4>
<ul>
@foreach ($positions as $position)
    <a href="{{ route('position', $position->id) }}"><li>{{ $position->title }}</li></a>
@endforeach
</ul>
```

Whilst testing the page, I realised I needed to change the route for the position filter set as I had accidentally assigned 'categories' instead of 'positions':

The Mistake Signified by the 'URL' on the Page



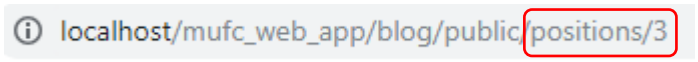
Altering the Routes File – Before and After

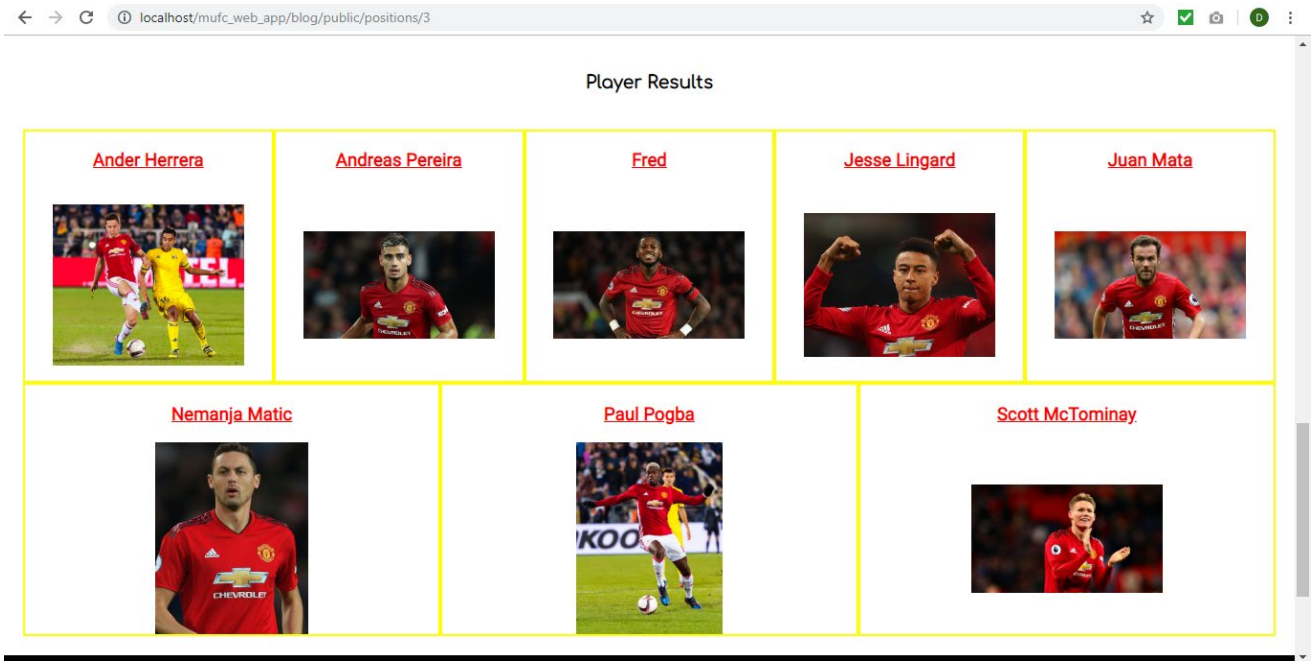
```
Route::any('/categories/{position}', [
    'uses' => 'PositionsController@position',
    'as' => 'position'
]);
```

```
Route::any('/positions/{position}', [
    'uses' => 'PositionsController@position',
    'as' => 'position'
]);
```

After changing the above, this now caused the position filter set to work successfully as seen with an example below:

The 'URL' had now Changed with the Correct Players Appearing on the Page for the Midfielder Position



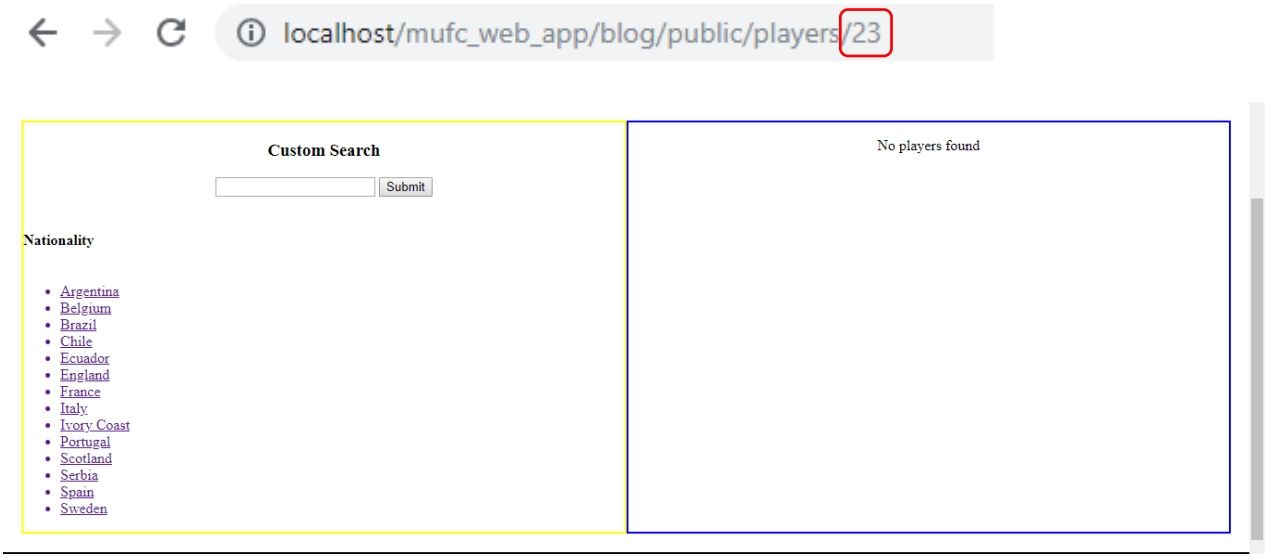


As I had now managed to allow the filters to work properly, this allowed for progression elsewhere.

Resolving Issues with Selecting a Player to View their Information/Statistics

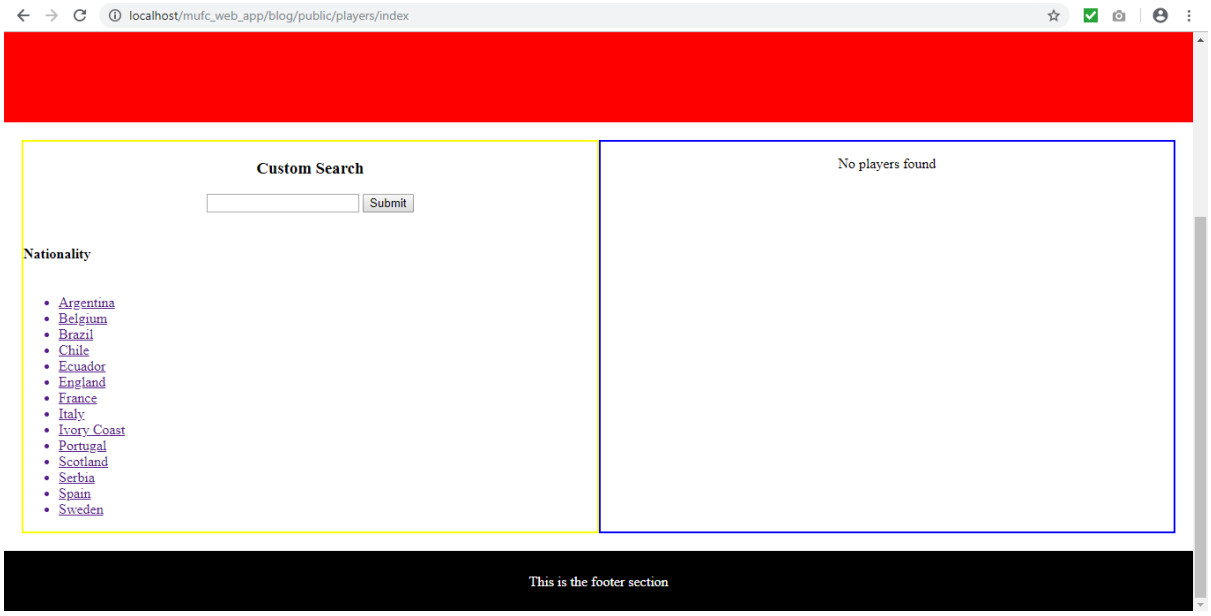
I needed to allow for the user to be navigated to a separate page where they could view a player's season statistics after selecting a link regarding each player's name. The setup of this will be viewable in the 'Initial Process' subsection of the 'Establishing Areas of the Application as well as Creating the Appearance' section. At this current moment, I experienced the following issue where selecting a player would navigate to 'index.blade.php' with 'No players found' rather than a separate page with the required player. Please note this was working before applying the filters aspect, as seen in the previous subsection:

The Issue with Being Navigated to the 'index.blade.php' File with no Player

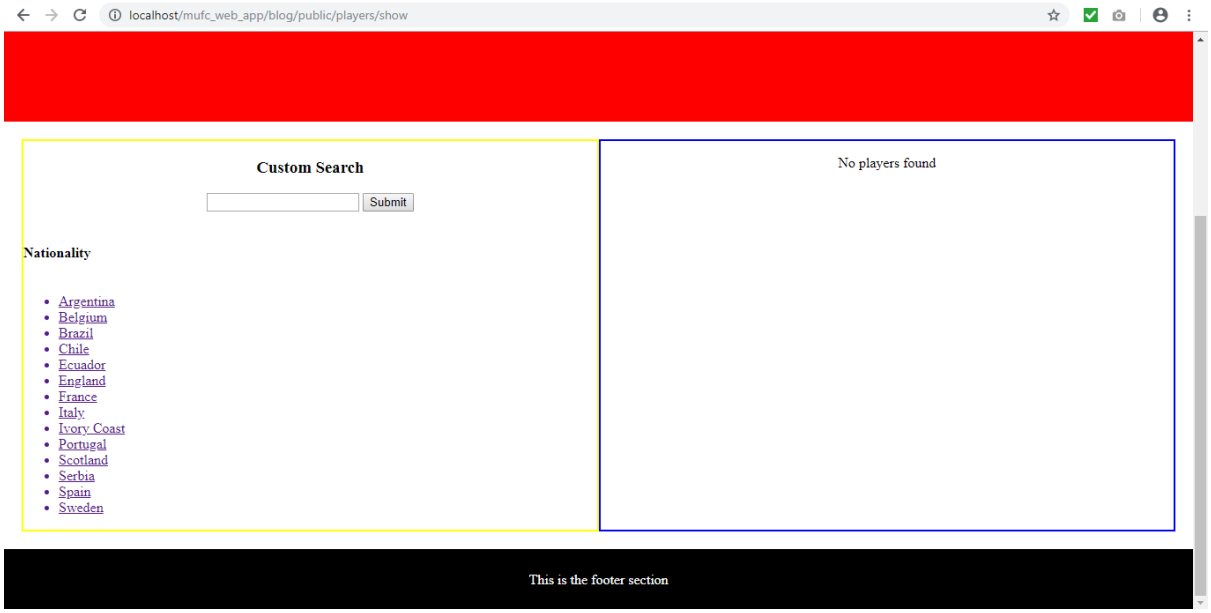


Whilst progressing further with this, this was a severe issue which I found difficult to resolve as I had noticed that typing in both 'index' and 'show' into the 'URL' would return the same result of that shown below. I believed that I should have encountered an error instead of the same page:

Entering 'index' into the 'URL' – This Returned the Same Page



Entering 'show' into the 'URL' – This Returned the Same Page



However, I then discovered by removing the nationality/categories route from the 'index.blade.php' file, that this resolved the issue regarding showing the same outcome as displayed above. However, this didn't resolve the issue regarding selecting a player to view separately.

As will have been explained when resolving the previously shown filter sets issue, the solution to this aspect helped to then allow the 'show' aspect to work properly. A reminder of this solution as well as the resolved issue is evident below:

Assigning each Public Function to their Relevant Controllers and Removing Other Code (CategoriesController.php File) – This Related to the Filter Sets

```
public function index()
{

    $players = Player::orderBy('name', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    return view('players.index', compact('players', 'categories', 'ages', 'positions'));

}
public function category($id)
{

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    $players = Player::orderBy('name')->get()->where('category_id', $id);

    return view('players.index', compact('players', 'categories', 'ages', 'positions'));

}
```

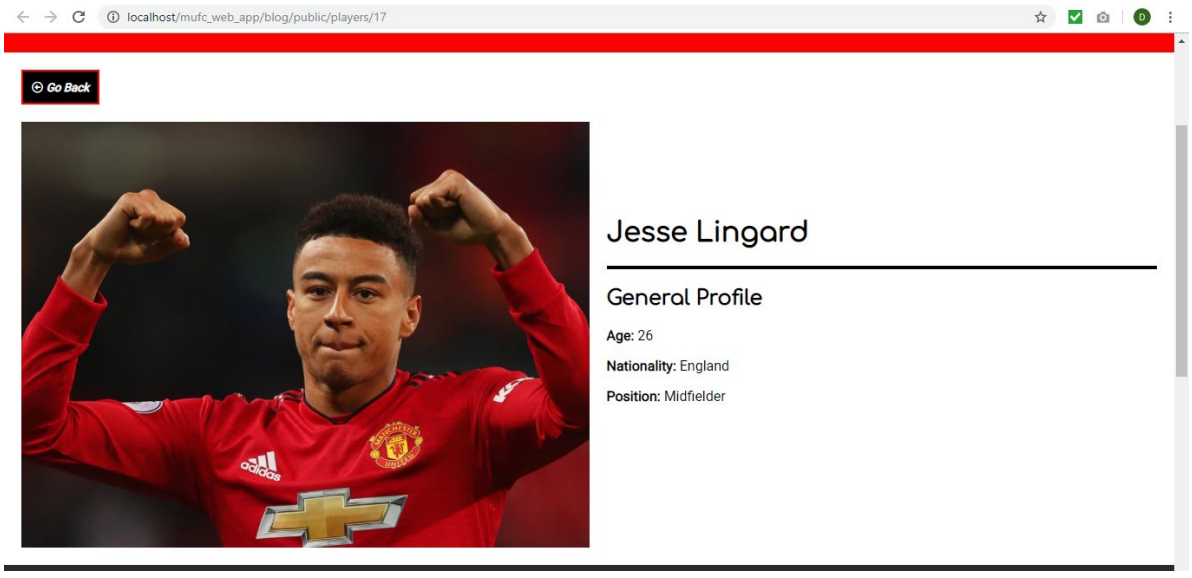
Assigning each Public Function to their Relevant Controllers and Removing Other Code (AgesController.php File) – This Related to the Filter Sets

```php
public function index()
{

    $players = Player::orderBy('name', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    return view('players.index', compact('players', 'categories', 'ages'));

}

public function age($id)
{

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

    $positions = Position::with('players')->orderBy('title', 'asc')->get();

    $players = Player::orderBy('name')->get()->where('age_id', $id);

    return view('players.index', compact('players', 'categories', 'ages', 'positions'));

}
```

Changing the Routes in the 'web.php' File – This Related to the Filter Sets

```php
Route::any('/ages/{age}', [
    'uses' => 'AgesController@age',
    'as' => 'age'

]);

Route::any('/categories/{category}', [
    'uses' => 'CategoriesController@category',
    'as' => 'category'
]);
```

The Player now Appeared on a Separate Page with Correct Information after Selecting the Player's Name



This now signified the end of resolving this issue, allowing for progression elsewhere.

Refining the Appearance and Finalising the Project

After having been able to allow for the filters and custom search to successfully work, despite challenges encountered, I now believed it would be beneficial to enhance the design of the application. This was to match that of the wireframes I had created and to make the website application more appealing.

Altering the Template File

To begin, I changed the template file to represent some of the heading and footer aspects contained within the created wireframes. This can be viewed below:

The 'mufc_app_template1.blade.php' File – Before

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="{{ asset('css/stylesheet.css') }}" rel="stylesheet">
    <title>MUFC Web App - Home</title>
</head>
<body>
    <div class="navigation_bar">
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Player Comparison</a></li>
        </ul>
    </div>
    <div class="application_title_container">
        <h1>Manchester United Website Application</h1>
    </div>
    @yield('content')
    <div style="width:100%;height:auto;padding:10px;background-color:black;color:white;text-align:center;">
        <p>This is the footer section</p>
    </div>
</body>
</html>
```

The 'mufc_app_template1.blade.php' File – After

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="{{ asset('css/stylesheet.css') }}" rel="stylesheet">
    <title>MUFC Web App - Home</title>
</head>
<body>
    <div class="header_container">
        <div class="header_title_container">
            <h1>Manchester United Player Statistics 2017/18</h1>
            <hr>
        </div>
        <div class="header_navigation_container">
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">Individual Players</a></li>
                <li><a href="#">Player Comparison</a></li>
            </ul>
        </div>
    </div>
    @yield('content')
    <div class="footer_container">
        <div class="fc_sub_container1">
            <div class="fc_logo_container">
                <div class="fc_logo_sub_container">
                    <img src="images/man_utd_logo.png" class="footer_logo">
                </div>
            </div>
            <div class="fc_sources_container">
                <div class="fc_sources_sub_container">
                    <p>Produced by Daniel Wilkins with aid of sources viewable <a href="#" target="_blank">here</a></p>
                </div>
            </div>
        </div>
        <div class="fc_hr_container">
            <hr>
        </div>
        <div class="fc_sub_container2">
            <div class="fc_page_links_container">
                <div class="fc_page_links_sub_container">
                    <div class="fc_plsc_heading_container">
                        <h3>Page Links</h3>
                        <hr>
                    </div>
                    <div class="fc_plsc_links_container">
                        <ul>
                            <li><a href="#">Home</a></li>
                            <li><a href="#">Individual Players</a></li>
                            <li><a href="#">Player Comparison</a></li>
                        </ul>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

As will be evident above, I originally had a very basic structure which was to allow for progression with the 'back-end' aspects. Within the changed file, I added more containers to allow for 'Flexbox' to be applied. The 'header_container' aspect was the parent container for the header section which contained two containers relating to both the title and navigation aspects. Regarding the footer section, this was divided into several containers to separate the logo, sources and navigation sections as well as including a container for a 'hr' element. This would also help create a divide but in-between the logo and sources and navigation sections.

The CSS for the Template – Before

```
/* TEMPLATE STYLES START */

/* Navigation Bar Section Start */

.navigation_bar {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 20px;
    background-color: black;
}

.navigation_bar ul {
    list-style-type: none;
}

.navigation_bar li {
    display: inline-block;
    padding: 10px;
}

/* Navigation Bar Section End */

/* Application Heading Section Start */

.application_title_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 100px;
    background-color: red;
    border: 2px solid red;
    text-align: center;
    justify-content: center;
    color: white;
}

/* Application Heading Section End */
```

As is evident here, I originally styled the navigation section to display the navigation links in horizontal format through the use of 'flex-direction: row' for the 'navigation_bar' container. I had also added 'padding' to allow for space to be created surrounding aspects contained within this container. Furthermore, for the 'navigation_bar' aspect, which related to the navigation bar itself, I set 'list-style-type' to 'none' so that the bullet points would be removed, creating a better appearance. I also applied 'padding' to each of the link elements to allow for them to be spaced further apart.

With regards to the 'application_title_container', the key styles I had assigned was setting a width of 100% to allow the container to fill the whole width of the page whilst also adding 'padding' to place space around the title element inside. I also assigned 'text-align: center' and 'justify-content: center' to help place the heading centrally.

Also, I had added inline styles for the footer section to create a basic appearance, also setting the width of the container to be 100%, adding 'padding' and 'text-align: center' for the same reasons stated above.

The CSS for the Template – After

```css
/* Header Section Start */

.header_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    background-color: #FF0000;
    padding: 30px;
}

.header_title_container {
    width: 100%;
    height: auto;
    text-align: center;
    color: #FFFFFF;
}

.header_title_container hr {
    width: 80%;
    height: auto;
    border: 2px solid #FFFFFF;
}

.header_navigation_container {
    display: flex;
    justify-content: center;
    width: 100%;
    height: auto;
}

.header_navigation_container ul {
    display: flex;
    flex-direction: row;
    list-style-type: none;
}

.header_navigation_container li {
    border: 2px solid #000000;
    background-color: #000000;
    padding: 10px;
    margin-left: 10px;
    margin-right: 10px;
}
```

```css
.header_navigation_container a {
    color: #FFFFFF;
}

/* Header Section End */

/* Footer Section Start */

.footer_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    background-color: #000000;
    padding: 10px;
    color: #FFFFFF;
}

.fc_sub_container1 {
    display: flex;
    flex-direction: row;
    align-items: center;
    width: 55%;
    height: auto;
}

.fc_logo_container {
    width: 20%;
    height: auto;
}

.fc_logo_sub_container {
    width: 100%;
    height: auto;
}

.footer_logo {
    width: 100%;
    height: auto;
}

.fc_sources_container {
    width: 80%;
    height: auto;
    padding-left: 20px;
}
```

```css
.fc_sources_sub_container {
    width: 100%;
    height: auto;
}

.fc_sub_container2 {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 35%;
    height: auto;
    padding-right: 50px;
}

.fc_page_links_container {
    width: 100%;
    height: auto;
}

.fc_page_links_sub_container {
    width: 100%;
    height: auto;
}

.fc_plsc_heading_container {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 100%;
    height: auto;
}

.fc_plsc_heading_container hr {
    width: 100%;
    height: auto;
    border: 2px solid #FF0000;
}
```

```css
.fc_hr_container {
    width: 10%;
    height: auto;
}

.fc_hr_container hr {
    width: 1%;
    height: 26%;
    transform: rotate(180deg);
    border: 2px solid #FFFFFF;
}
```

As is evident above, I added 'flex-direction: column' to the parent container 'header_container' to allow for containers inside to be displayed in a column format. This was because the navigation bar/links needed to be placed underneath the title. Regarding the title section of the header, the 'hr' tag was styled to be both white and 80% in width, creating a tidy appearance. For the container called 'header_navigation_container', 'justify-content: center' was assigned to place all the navigation links to the centre of the page. One final aspect to note regarding the header section is that 'margin-left' and 'margin-right' were assigned to each link/list element to help place space between each navigation link section with 'padding' allowing for a box effect to be created through also using a background colour of black.

Regarding the footer section, the parent container 'footer_container' was assigned a 'flex-direction' of 'row' to allow all containers/content to be displayed in horizontal format. Furthermore, the two containers 'fc_sub_container1' and 'fc_sub_container2', relating to the aspects containing the logo and sources and navigation sections, were assigned widths to relate to the space required for each. The styles of 'align-items: center' and 'justify-content: center' helped to position the content inside these containers vertically central. A 'flex-direction' of 'row' was assigned to the first mentioned container to display content horizontally and a 'flex-direction' of 'column' was assigned to the second mentioned container to display content vertically. Within each of these containers, relevant widths were assigned to each section with the logo image styles being placed at 100% in width with a height of auto to allow for adjustment to different changes in its container's size. One final aspect to note is that the container containing the 'hr' tag was set to the width of 10% as this didn't require too much space but enough to separate each section surrounding this. The 'hr' tag itself was assigned styles such as 'transform: rotate(180deg)' to position this vertically rather than horizontally.

Creating the 'Introduction/Home' Page

After having refined the styles for the header and footer sections, I then decided to create the introduction page to welcome users to the website application. To begin, I created a controller called 'HomeController2' to allow for returning the new view for this page. This was undertaken through utilising the 'command line interface', as seen before. After having completed this, I then added code to return the view of the new file created called 'home.blade.php' file as well as assigning a new route. This process can be viewed below:

Adding the Relevant Code to Return the View in the Controller

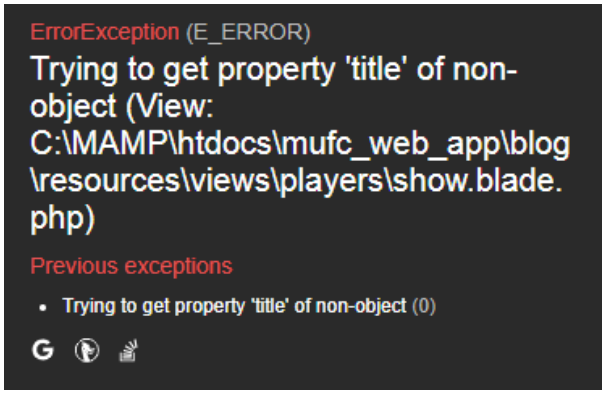
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController2 extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('players.home');
    }
```

Creating the new Route in the 'web.php' File

```php
Route::get('/home', 'HomeController2@index');
```

However, whilst testing to see if the view would be returned, I encountered the following error which can be viewed below:

The Encountered Error



I then believed that because there was already a file called 'home.blade.php' within another folder that changing the name would help to resolve this. Therefore, I did this whilst also updating the other relevant files and this resolved the issue as will be evident below:

Altering the Route to Relate to the new File Name

```php
Route::get('/test', 'HomeController2@index');
```

Altering the Controller to Relate to the new File Name

```php
public function index()
{
    return view('players.test');
}
```

The Current Code for the 'test.blade.php' File

```
@extends('layouts.mufc_app_template1')

@section('content')

<h1>TEST</h1>

@endsection
```

As is evident above, at this stage, I had included the previously created template for this file with the '@section('content')' allowing for changing the content to match that of this page. I had only included a title to test if this would work.

The Outcome on the Page – This now Successfully Displayed

After I had understood that this had successfully worked, I then continued to style the page and add content. This can be viewed below:

Adding the Content and Structuring for the Page in the 'test.blade.php' File

```
@extends('layouts.mufc_app_template1')
<title>MUFC Web App - Home</title>

@section('content')

<div class="page_welcome_container">
    <div class="page_welcome_sub_container">
        <h2>Welcome!</h2>
        <hr>
        <p>The purpose of this website application is to allow you, as Manchester United fans, to view player statistics for
        different players at Manchester United. Within this application, you will be able to view individual players as well as
        being able to compare players next to each other. To undertake these actions, please select the relevant links provided
        below.</p>
    </div>
</div>

<div class="individual_players_container">
    <div class="individual_players_sub_container">
        <div class="ipc_title_container">
            <h3>Individual Players</h3>
            <i class="far fa-futbol"></i>
        </div>
        <div class="ipc_description_container">
            <p>By selecting the link below, you will be able to view a player's profile and statistics through selecting their
            name provided on the page. If you wish to search for or filter players, search options are provided at the top of
            this page, once navigated to, to help undertake this process.</p>
            <a href="http://localhost/mufc_web_app/blog/public/players"><button>Page Link</button></a>
        </div>
    </div>
</div>

<div class="player_comparison_container">
    <div class="player_comparison_sub_container">
        <div class="pcc_description_container">
            <p>By selecting the link below, you will be able to view two selected players next to each other to be able to
            compare statistics for the season. To view certain players, once having navigated to the page, you will be able to
            undertake a custom search or select search filters to achieve this.</p>
            <a href="#"><button>Page Link</button></a>
        </div>
        <div class="pcc_title_container">
            <h3>Player Comparison</h3>
            <i class="far fa-futbol"></i>
        </div>
    </div>
</div>

@endsection
```

As will have been evident above I had divided the page into three sections. These related to the welcome section where the user would be welcomed to the page and website application and the sections relating to what could be achieved within the website application. These sections were the individual players and player comparison sections of the website application. Within both of these sections, 'Font Awesome' was utilised to display images of footballs, relating to the wireframe design whilst also including links to each of the relevant pages. Furthermore, the 'title' aspect was implemented as it was understood that applying this to the template would cause the same title to be displayed on each page. Implementing this into each individual page would make this unique to each page.

The CSS Code for this Page

```css
/* HOME PAGE START */

/* Welcome Section Start */

.page_welcome_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    padding: 50px;
}

.page_welcome_sub_container {
    width: 50%;
    height: auto;
    text-align: center;
    margin: auto;
}

.page_welcome_container hr {
    width: 100%;
    height: auto;
    border: 2px solid #FF0000;
}

/* Welcome Section End */
```

```css
/* Individual Players Section Start */

.individual_players_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 50px;
    background-color: #2A2A2A;
    color: #FFFFFF;
}

.individual_players_sub_container {
    display: flex;
    flex-direction: row;
    width: 80%;
    height: auto;
    margin: auto;
    align-items: center;
}

.ipc_title_container {
    width: 50%;
    height: auto;
    text-align: center;
}

.ipc_title_container i {
    font-size: 250px;
}

.ipc_description_container {
    width: 50%;
    height: auto;
}
```

```css
.ipc_description_container button {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    padding: 10px;
    cursor: pointer;
    transition: 0.5s;
    font-style: italic;
    font-weight: bold;
}

.ipc_description_container button:hover {
    background-color: #FF0000;
    border: 2px solid #000000;
    transition: 0.5s;
}
```

```css
/* Player Comparison Section Start */

.player_comparison_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 50px;
}

.player_comparison_sub_container {
    display: flex;
    flex-direction: row;
    width: 80%;
    height: auto;
    margin: auto;
    align-items: center;
}

.pcc_title_container {
    width: 50%;
    height: auto;
    text-align: center;
}

.pcc_title_container i {
    font-size: 250px;
}

.pcc_description_container {
    width: 50%;
    height: auto;
}

.pcc_description_container button {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    padding: 10px;
    cursor: pointer;
    transition: 0.5s;
    font-style: italic;
    font-weight: bold;
}
```

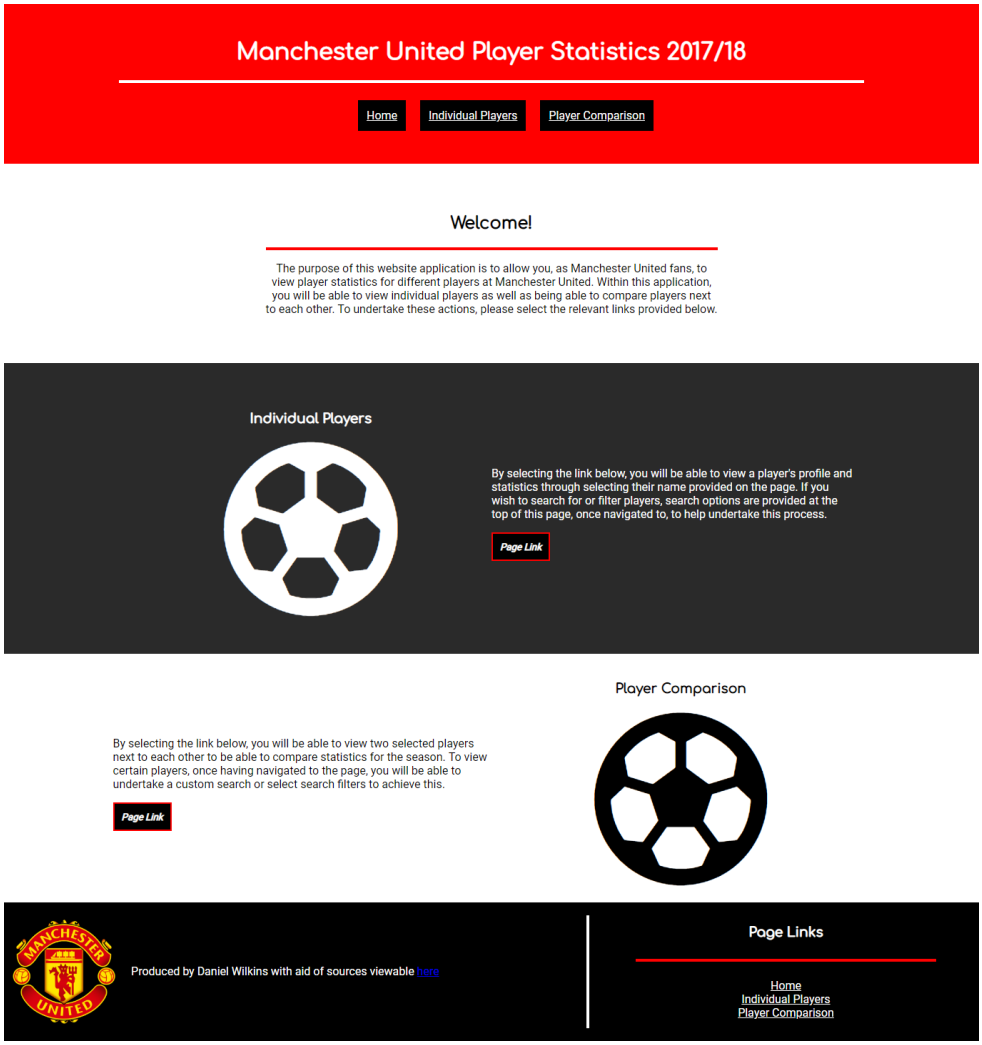
```css
.pcc_description_container button:hover {
    background-color: #FF0000;
    border: 2px solid #000000;
    transition: 0.5s;
}

/* Player Comparison Section End */
```

As will have been noticeable above, the 'page_welcome_container' acted as the parent container for the welcome section. The 'flex-direction' was set to 'column' to display content vertically with 'padding' allowing for space to be placed around the content inside, creating a better appearance. Furthermore, the 'page_welcome_sub_container' was assigned a width of 50% with 'margin: auto' to allow for the text to be placed more centrally rather than being stretched to the whole width of the page.

Regarding the section for the individual players information, the parent container 'individual_players_container' was assigned a 'flex-direction' of 'row' to place content horizontally, matching that of the required design. Likewise to the sub container for the heading section, the sub container for this section had a width of less than 100% with 'margin: auto' assigned to place the content more centrally. Additionally, 'align-items: center' was assigned to allow for the text to be placed vertically central next to the football icon, creating a better appearance. Equal widths were assigned to both the sections containing the title and description to create a balanced appearance with the 'font-size' being set to a large figure to ensure that the football icon wouldn't appear too small on the page. The button aspect was styled so that it would change colour when being hovered over with 'cursor: pointer' being assigned to inform users that this was something to interact with.

Regarding the section for the player comparison information, this was styled in the same way as that explained above for the individual players information. The outcome can be viewed below:

The Outcome on the Web Page



Altering the Appearance of the 'Individual Players' Page

Another aspect I improved visually was the individual players page. This was to match that of the chosen wireframe design. This process can be viewed below:

Altering the 'index.blade.php' File (Individual Players Page HTML Code)

```
<link href="{{ asset('css/stylesheet.css') }}" rel="stylesheet">

@extends('layouts.mufc_app_template1')

<title>MUFC Web App - Individual Players</title>

@section('content')

<div class="page_heading_container">
    <div class="page_heading_sub_container">
        <h2>Individual Players</h2>
        <hr>
    </div>
</div>

<div class="player_statistics_container">
    <div class="psc_filter_container">
        <div class="typed_search_container">
            <h3>Custom Search</h3>
            <form action="http://localhost/mufc_web_app/blog/public/search" method="post">
                {{ csrf_field() }}
                <label for="query"></label>
                <input type="query" id="query" name="query">
                <input type="submit" class="submit_styles">
            </form>
        </div>

        <div class="filters_search_container">
            <div class="fsc_title_container">
                <h3>Player Filters</h3>
            </div>
            <div class="fsc_main_container">
                <div class="age_filter_container">
                <h4>Age</h4>
                    <ul>
                    @foreach ($ages as $age)
                        <a href="{{ route('age', $age->id) }}"><li>{{ $age->title }}</li></a>
                    @endforeach
                    </ul>
                </div>
```

```
                <div class="nationality_filter_container">
                    <h4>Nationality</h4>
                    <ul>
                    @foreach ($categories as $category)
                        <a href="{{ route('category', $category->id) }}"><li>{{ $category->title }}</li></a>
                    @endforeach
                    </ul>
                </div>

                <div class="position_filter_container">
                    <h4>Position</h4>
                    <ul>
                    @foreach ($positions as $position)
                        <a href="{{ route('position', $position->id) }}"><li>{{ $position->title }}</li></a>
                    @endforeach
                    </ul>
                </div>
            </div>
        </div>
    </div>

    <div class="psc_title_container">
        <h3>Player Results</h3>
    </div>

    @if(count($players) > 0)
    @foreach($players->chunk(5) as $chunk)
            <div class="psc_pc_players">
            @foreach($chunk as $player)
                <div class="psc_pc_players2">
                    <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}" target="_blank">{{$player->name}}</a></h3>
                    <img src="{{$player->image}}" class="psc_pc_player_images">
                </div>
            @endforeach
            </div>
    @endforeach
    @else
        <p>No players found</p>
    @endif
</div>

@endsection
```

As can be seen above, I had integrated an individual 'title' section for this page, as also explained with the introduction/home page. The page was divided into two main sections which were the page title section and the main content section. The title section included the page title. Within the main content section, there were two further sections with one relating to the search/filters aspect and one relating to the displaying of players aspect. Within the filters section, there were subsections for both the custom search and filters relating to age, nationality and position. Within the players section, there was a section to display the players with links surrounding their names to be able to

view individual players on the 'show' page. One final aspect to note is that 'foreach' statements were integrated to allow for styling each piece of data incorporated into this page. Please note that the 'chunk' method will be explained later.

Altering the CSS Code

```css
/* INDIVIDUAL PLAYERS PAGE START */

/* Page Heading Section Start */

.page_heading_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    padding: 50px;
}

.page_heading_sub_container {
    width: 50%;
    height: auto;
    text-align: center;
    margin: auto;
}

.page_heading_container hr {
    width: 100%;
    height: auto;
    border: 2px solid #FF0000;
}

/* Page Heading Section End */
```

```css
/* Main Content Section Start */

.player_statistics_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    padding: 20px;
    background: white;
    color: black;
}

/* Main Content Section - Search/Filters Section Start */

.psc_filter_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
}

.psc_filter_container a {
    color: #FF0000;
}

.typed_search_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.typed_search_container input {
    font-family: 'Roboto', sans-serif;
}

.submit_styles {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    cursor: pointer;
}
```

```css
.filters_search_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    padding: 10px;
}

.fsc_title_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.fsc_main_container {
    width: 100%;
    height: auto;
    display: flex;
    flex-direction: row;
}

.filters_search_container ul {
    list-style-type: none;
    padding: 0;
}

.filters_search_container li {
    padding: 10px;
    border: 2px solid #000000;
    margin-top: 10px;
}

.age_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

.nationality_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}
```

```css
/* Main Content Section - Players Section Start */

.psc_title_container {
    width: 100%;
    height: auto;
    text-align: center;
    margin-bottom: 20px;
}

.psc_pc_players {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
}

.psc_pc_players2 {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    border: 2px solid yellow;
    text-align: center;
}

.psc_pc_players2 a {
    color: #FF0000;
    transition: 0.5s;
}

.psc_pc_players2 a:hover {
    color: #000000;
    transition: 0.5s;
}

.psc_pc_player_images {
    width: 200px;
    height: 200px;
    object-fit: contain;
    margin: auto;
}

/* Main Content Section - Players Section End */
```

The parent container for the heading section was called 'page_heading_section' where 'padding' was applied to allow for space to be placed around the title, creating a better appearance. Furthermore, a width of 50% and 'margin: auto' helped to centre this section regarding the 'page_heading_sub_container'. Regarding the container relating to the main content, 'player_statistics_container', 'flex-direction: column' was assigned to position the containers relating to searching and the players themselves vertically, relating to the required design. Regarding the parent container for both the custom search and filters, 'psc_filter_container', 'flex-direction: column' was also assigned to allow for the filters to be placed underneath the custom search. However, the container relating to each of the filters was assigned a 'flex-direction' of 'row' as these needed to be displayed inline. Regarding the 'psc_pc_players2' container, a 'flex-direction' of 'column' was assigned to relate to the 'chunk' method, allowing for players to not be displayed all in one row. Furthermore, the player images were set to be 200px in width and height with 'object-fit: contain' as assigning a width of 100% with a height of auto would cause some images to display very large on the page. Some other styles related to changing colours and fonts of elements.

The outcome of these changes can be viewed below:

The Outcome on the Web Page

As mentioned previously, I integrated the 'chunk' method which allowed for displaying a certain quantity of players for a row on this page. Before discovering this, this was an area which I experienced difficulties attempting to understand. The process of integrating the 'chunk' method can be viewed below with the challenges encountered.

When first integrating the 'chunk' method, I had the following code:

The Current Code when First Integrating the 'chunk' Method

```
<div class="psc_player_container">
@if(count($players) > 0)
@foreach($players->chunk(3) as $chunk)
    <div class="psc_pc_players">
        <div class="psc_pc_players2">
            @foreach($chunk as $player)
                <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}" target="_blank">{{$player->name}}</a></h3>
                <img src="{{$player->image}}" class="psc_pc_player_images" ">
            Player {{ $loop->iteration }}
            @endforeach
        </div>
    </div>
@endforeach
@else
    <p>No players found</p>
@endif
</div>
```

As is evident above, I had placed the content other than the container within a 'foreach' statement whilst also adding 'Player {{ $loop->iteration }}', as seen from an online resource. This was because I thought that the content for each player would then be displayed a certain quantity of times before being placed on the next row with new players.

However, when viewing the result, the 'chunk' method was working but wasn't displaying 3 players per row. This is evident below:

The Outcome – This Didn't Successfully Work

Following on from the previous aspect, I then modified the code to the following shown below:

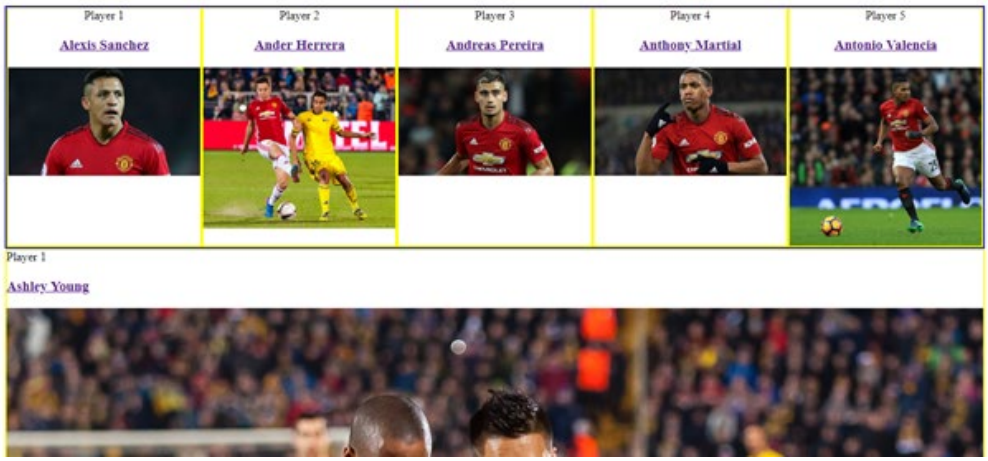
Altering the Code

```
<div class="psc_player_container">
@if(count($players) > 0)
@foreach($players->chunk(5) as $chunk)
        @foreach($chunk as $player)
        <div class="psc_pc_players">
            <div class="psc_pc_players2">
            Player {{ $loop->iteration }}
                <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}" target="_blank">{{$player->name}}</a></h3>
                <img src="{{$player->image}}" class="psc_pc_player_images" ">
            </div>
        </div>
        @endforeach
    </div>
@endforeach
@else
    <p>No players found</p>
@endif
</div>
```

What I had undertaken at this stage involved placing the containers inside of the 'foreach' statement as well. This was because I thought that this would then cause the formatting to correct itself, displaying a certain quantity of players per row. This time, this was meant to output 5 players per row.

From viewing the outcome, I understood this was working to an extent but that after one row of players, this would then display one player per row:

The Outcome – This Worked to an Extent



Following on from this, I then thought that placing the 'Player {{ $loop->iteration }}' outside of the container consisting of the content for each player would help. However, this didn't as this caused 'Player 1', for example, to be placed next to each player:

Modifying the Code

```
<div class="psc_player_container">
@if(count($players) > 0)
@foreach($players->chunk(5) as $chunk)
        @foreach($chunk as $player)
        <div class="psc_pc_players">
                        Player {{ $loop->iteration }}
            <div class="psc_pc_players2">
                <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}" target="_blank">{{$player->name}}</a></h3>
                <img src="{{$player->image}}" class="psc_pc_player_images" ">
            </div>
        </div>
        @endforeach
    </div>
@endforeach
@else
    <p>No players found</p>
@endif
</div>
</div>
```

The Outcome – This Didn't Resolve the Issue

After having analysed the code, I then believed the problem may have been caused by not including a closing 'div'. Therefore, I added this into the code but this then caused all players to be displayed on one row again. Please note, I had also moved one of the closing 'div' tags outside of both 'foreach' statements which may have also caused this issue:

Adding the Missing 'div' into the Code and Moving Another 'div'

```
    <div class="psc_player_container">
    @if(count($players) > 0)
    @foreach($players->chunk(5) as $chunk)
            @foreach($chunk as $player)
            <div class="psc_pc_players">
                <div class="psc_pc_players2">
                    <h3><a href="/mufc_web_app/blog/public/players/{{$player-
                    <img src="{{$player->image}}" class="psc_pc_player_images
                </div>
            </div>
            @endforeach
    @endforeach
    </div>
    @else
        <p>No players found</p>
    @endif
</div>
</div>

@endsection
```

The Outcome – This Caused All Players to be Displayed in One Row Again

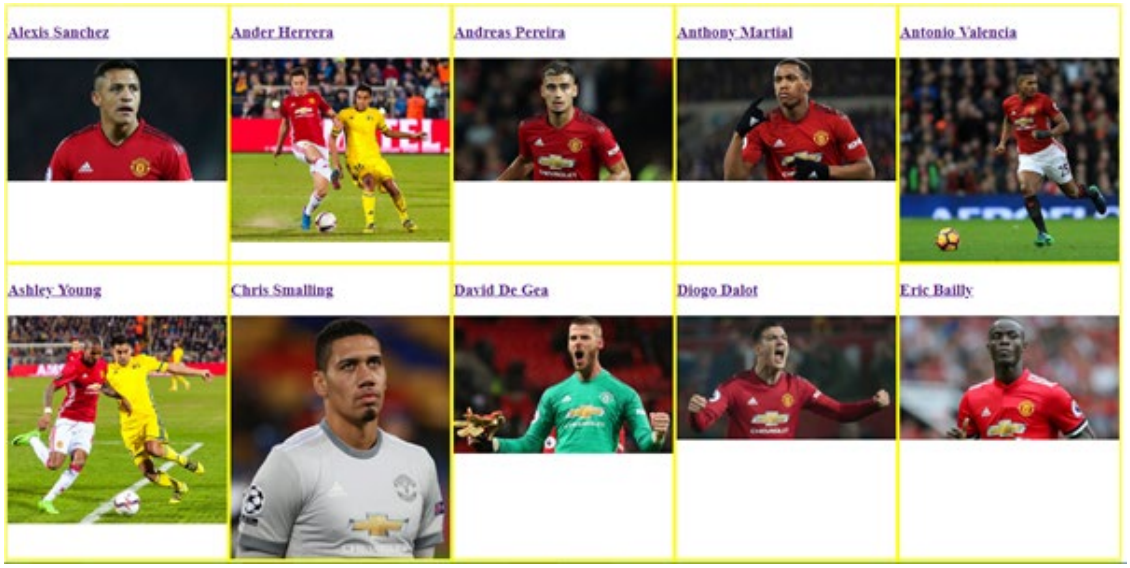
After viewing the documentation from 'Laravel', I then managed to resolve this issue by removing the parent container called 'psc_player_container' and changing the positioning of the containers to be in the correct place:

Removing the Parent Container and Repositioning Other Containers

```
    @if(count($players) > 0)
    @foreach($players->chunk(5) as $chunk)
            <div class="psc_pc_players">
            @foreach($chunk as $player)
                <div class="psc_pc_players2">
                    <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}" target="_blank">{{$player->name}}</a></h3>
                    <img src="{{$player->image}}" class="psc_pc_player_images">
                </div>
            @endforeach
            </div>
    @endforeach
    @else
        <p>No players found</p>
    @endif
</div>
```

As can be seen above, the 'chunk' method was placed around the container relating to the individual players and not the row itself with the 'psc_pc_players' container placed outside, acting as the row. The outcome of this can be viewed below:

The Outcome of these Changes – This Resolved the Issue (Examples)



This now signified the completion of this page with the structure and style changes shown before.

Altering the Appearance of the 'show' Page (Actual Individual Players with their Statistics)

Another aspect I changed related to the page displayed after selecting a certain player on the 'Individual Players' page. From the initial stages, the current code for this page can be viewed below:

The Code for the 'show.blade.php' File

```
@extends('layouts.mufc_app_template1')
@section('content')
    <a href="/mufc_web_app/blog/public/players">Go Back</a>
    <h1>{{$player->title}}</h1>
    <small>Written on {{$player->created_at}}</small>
    <div>
        {{$player->nationality}}
<img src="{{$player->image}}" style="width: 50%; height: auto;">
    </div>

@endsection

```

As is evident above, I had a very basic structure. This included a 'Go Back' button to return users back to the 'Individual Players' page and limited information regarding each player. This was because I had been wanting to test certain aspects before refining and adding more content. This aspect was helped through a tutorial viewed on 'YouTube'.

Whilst beginning to change the page to relate to the wireframe chosen, I wanted to be able to allow users to filter player statistics by different matchdays as well as for the full season. To begin, I first of all attempted to do this with the full season statistics. I firstly thought about creating a table to consist of the full season statistics for each player. I believed I could utilise foreign keys to link the statistics to the players in the 'players' table in the database.

To begin, I created a migration and model for the full season statistics aspect to allow for entering of and using data regarding the created table. I also created a controller as I believed this would need to be used to control the filtering aspect:

Creating the Model and Migration in the 'Command Line Interface'

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:model FullSeasonStatistic -m
Model created successfully.
Created Migration: 2019_04_15_133652_create_full_season_statistics_table
```

Creating the Controller in the 'Command Line Interface'

```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller FullSeasonStatistic
sController --resource
Controller created successfully.
```

After successfully creating the previous two aspects, I then added a route for this newly created element to allow for the filtering aspect to work later, taking inspiration from the filters on the 'Individual Players' page:

Adding the Route for this Aspect

```
Route::any('/fullseasonstatistics/{fullseasonstatistic}', [
    'uses' => 'FullSeasonStatisticController@fullseasonstatistic',
    'as' => 'fullseasonstatistic'
]);
```

Furthermore, I added similar aspects as before with other controllers and models to allow this to relate to the players and other filters as I thought this was needed:

Adding Similar Aspects as Before to the Controller and Model Files for the Full Season Statistics

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;
use App\FullSeasonStatistic;

class FullSeasonStatisticsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $ages = Age::with('players')->orderBy('title', 'asc')->get();

        $categories = Category::with('players')->orderBy('title', 'asc')->get();

        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        $players = Player::orderBy('name')->get()->where('position_id', $id);

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));
    }
```

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class FullSeasonStatistic extends Model
{
    public function players() {
        return $this->hasMany(Player::class);
    }
}
```

Adding Similar Aspects as Before to the Model File for the Players

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Player extends Model
{
/*    //Table Name
    protected $table = 'players';
    //Primary Key
    public $primaryKey = 'id';
    //Timestamps
    public $timestamps = true;*/

    public function category() {
        return $this->belongsTo(Category::class);
    }

    public function age() {
        return $this->belongsTo(Age::class);
    }

    public function position() {
        return $this->belongsTo(Position::class);
    }

    public function fullseasonsstatistic() {
        return $this->belongsTo(FullSeasonStatistic::class);
    }
}
```

Following on from this, I then added a new field to the migration file for the purpose of testing, executing 'php artisan migrate' to update the database:

Adding Another Field to the Migration File and the Outcome After Executing 'php artisan migrate'

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateFullSeasonStatisticsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('full_season_statistics', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('Goals');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('full_season_statistics');
    }
}
```

	id	Goals	created_at	updated_at
Edit Copy Delete	1	25	NULL	NULL

After completing this, I then altered the 'PlayersController.php' file to allow for displaying of the 'title' element on the page with the selected player. I also changed the 'show.blade.php' file to attempt to include this. This was to allow myself to understand if the data was being successfully pulled into the page. This process can be viewed below:

Altering the 'PlayersController.php' File

```php
use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;
use App\FullSeasonStatistic;
```

```php
public function show($id)
{
    $player = Player::find($id);
    return view('players.show')->with('player', $player);
    $fullseasonstatistics = FullSeasonStatistic::with('player')->orderBy('title', 'asc')->get();
}
```

Altering the 'show.blade.php' File

```php
<div class="poc_gic_information_container">
    <h1>{{$player->name}}</h1>
    <h2>General Profile</h2>
    <p><b>Age:</b> {{$player->age}}</p>
    <p><b>Nationality:</b> {{$player->nationality}}</p>
    <p><b>Position:</b> {{$player->position}}</p>
    <p><b>Position:</b> {{$player->fullseasonstatistic}}</p>
</div>
```

However, whilst testing this, I understood that this hadn't worked as will be seen below:

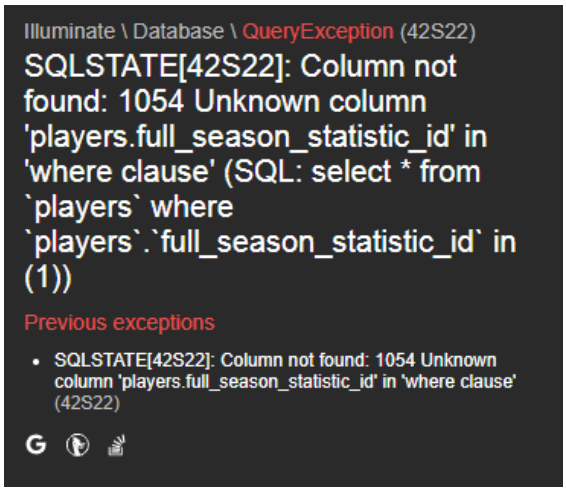
The Outcome – This Didn't Successfully Work

After having added in some missing elements that I had forgotten to include, I experienced an issue when adding the full season statistics aspect to the public function of 'index' in the 'PlayersController.php' file. I thought that adding in some of the missing elements would have helped to resolve the issue:

Adding the Full Season Statistics Aspect to the 'PlayersController.php' File

```
*/
public function index()
{

    $players = Player::orderBy('name', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $positions = Position::with('players')->orderBy('title', 'asc')->get();
    $fullseasonstatistics = FullSeasonStatistic::with('players')->get();

    return view('players.index', compact('players', 'categories', 'ages', 'positions', 'fullseasonstatistics'));
}
```

The Issue Experienced



From viewing this error, I then believed that I could have undertaken the same process as before with the other filters. This was to create a column within the 'players' table to relate to an 'ID'. This would then allow myself to potentially link to the other table with the full season statistics, collecting the data for a particular 'ID'.

Therefore, to begin, I added the column to the 'players' table, assigning different 'IDs' to each player as seen with the example shown below:

Adding the Column and Assigning 'IDs'

full_season_statistic_id
1
2
3
4
5
6
7
8
9
10

Whilst doing this, I also resolved an issue with the position of the player selected not displaying by adding in the column called 'position' and assigning values to each player. This is evident below:

Adding the 'position' Column to the 'players' Table and Assigning Values (Examples)

name	age	nationality	position
Anthony Martial	23	France	Forward
Marcus Rashford	21	England	Forward
Paul Pogba	26	France	Midfielder
David De Gea	28	Spain	Goalkeeper
Sergio Romero	32	Argentina	Goalkeeper
Victor Lindelof	24	Sweden	Defender
Eric Bailly	24	Ivory Coast	Defender

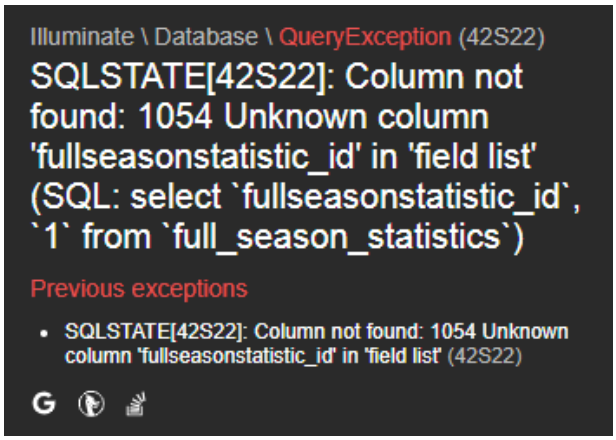
The Outcome – This now Worked (Examples)

Returning the previous problem, I then added code to the public function of 'fullseasonstatistic' in the 'FullSeasonStatisticsController.php' file. This was to collect the selected player by their 'ID' relating to the column previously shown above:

Adding Code to the Function in the Controller

```
public function fullseasonstatistic($id)
{
    $fullseasonstatistics = FullSeasonStatistic::all('fullseasonstatistic_id', $id)->get();
}
```

However, I then encountered the following error:

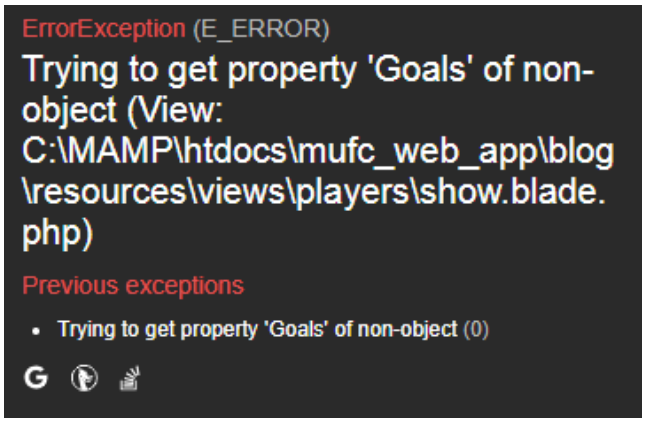


Following on from this, I then believed that if I were to return the view with both the 'player' and 'fullseasonstatistic' aspects, that this would help to display the relevant statistics for the selected player. However, this then returned another error:

Attempting to Return the View with Both Aspects

```
public function show($id)
{
    $player = Player::find($id);
    $fullseasonstatistic = FullSeasonStatistic::find($id);
    return view('players.show', compact('player', 'fullseasonstatistic'));
}
```

The Outcome – I Experienced an Error



As I was then struggling with this aspect, I then undertook research. From this, I understood that I could utilise the 'belongsToMany' aspect regarding foreign keys. Therefore, I attempted to integrate this into the public function of 'show', as is evident below:

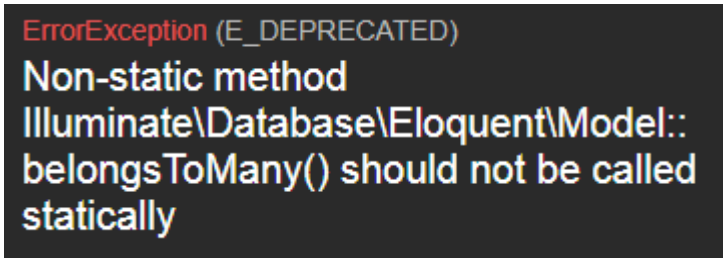
Modifying the Code to Include the 'belongsToMany' Element

```
public function show($id)
{
    $fullseasonstatistic = FullSeasonStatistic::belongsToMany(Player::class, "full_season_statistic_id");
    $player = Player::find($id);
    return view('players.show')->with('player', $player);
}
```

As is seen above, I tried to assign a player to their 'ID' for the full season statistics aspect. This was because I thought this would help to place the player on the page with only their statistics unique to them.

However, when testing this, I understood that the 'belongsToMany' aspect wouldn't be allowed to be used in this circumstance, as seen below:

The Error Experienced:



After returning to this problem later on, I then believed it would be easier to display only full season statistics from the previous season for each player by adding this manually to the table in the database. Therefore, I removed the idea of including filters. This was due to both inexperience and time constraints in this area. Examples of adding these statistics, used from the official 'Premier League' website, can be viewed below:

Adding the Statistics to the Table Manually (Examples)

goals	assists	passes	shots	tackles	fouls	offsides	minutes_played	yellow_cards	red_cards	saves
9	5	731	49	15	17	8	1581	1	0	0
7	5	576	61	14	21	9	1807	3	0	0
6	10	1721	76	33	44	3	2151	5	1	0
0	0	942	0	0	0	0	3330	0	0	115
0	0	20	0	0	0	0	90	0	0	3

Following on from adding the relevant statistics to the 'players' table, I then inserted this data into the 'show.blade.php' file, listing the statistics under appropriate headings such as 'Attack'. This can be viewed below:

Inserting the Data under Different Headings in the 'show.blade.php' File (Examples Highlighted)

```
<div class="player_statistics_container">
    <div class="psc_title_container">
        <h2>2017/18 Season Statistics</h2>
    </div>
    <div class="psc_categories_container">
        <div class="psc_cc_attack">
            <h3>Attack</h3>
            <p><b>Goals:</b> {{$player->goals}}</p>
            <p><b>Assists:</b> {{$player->assists}}</p>
        </div>
        <div class="psc_cc_defence">
            <h3>Defence</h3>
            <p><b>Tackles:</b> {{$player->tackles}}</p>
            <p><b>Saves:</b> {{$player->saves}}</p>
        </div>
        <div class="psc_cc_general">
            <h3>General Play</h3>
            <p><b>Minutes Played:</b> {{$player->minutes_played}}</p>
            <p><b>Passes:</b> {{$player->passes}}</p>
            <p><b>Shots:</b> {{$player->shots}}</p>
            <p><b>Offsides:</b> {{$player->offsides}}</p>
        </div>
        <div class="psc_cc_discipline">
            <h3>Discipline</h3>
            <p><b>Fouls:</b> {{$player->fouls}}</p>
            <p><b>Yellow Cards:</b> {{$player->yellow_cards}}</p>
            <p><b>Red Cards:</b> {{$player->red_cards}}</p>
        </div>
    </div>
</div>
```

The CSS Code

```
.player_statistics_container {
    display: flex;
    flex-direction: column;
    padding: 20px;
    width: 100%;
    height: auto;
    border: 2px solid #000000;
}

.psc_title_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.psc_categories_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    text-align: center;
}

.psc_cc_attack {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}

.psc_cc_defence {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}

.psc_cc_general {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}
```

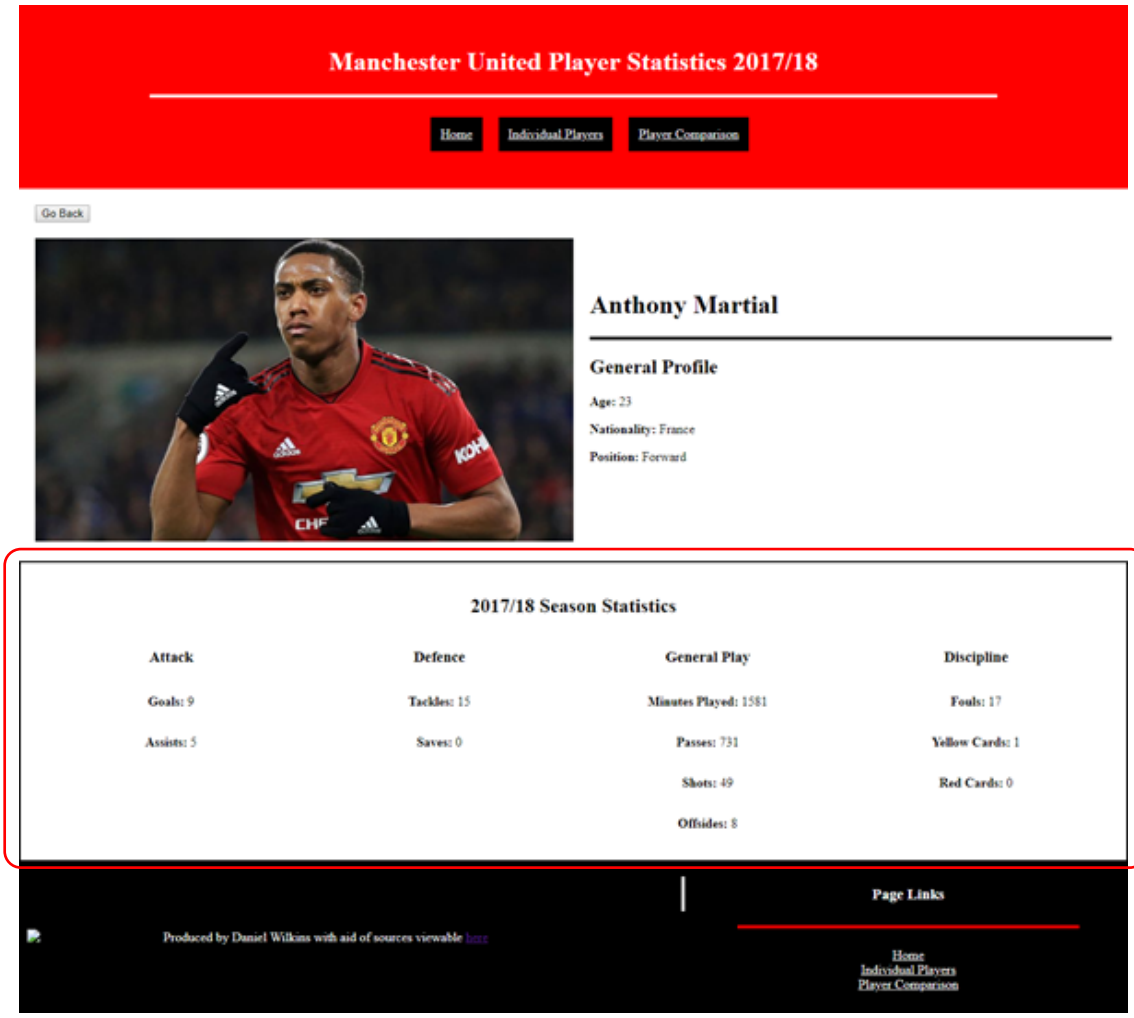
```
.psc_cc_discipline {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}
```

As can be seen above, at this stage, I had created a section relating to the statistics only. The parent container was 'player_statistics_container' whereby 'flex-direction: column' was applied to display

content vertically inside. Furthermore, there were two containers inside this container relating to both the title and statistics themselves. For the statistics container called 'psc_categories_container', the 'flex-direction' was set to 'row' to display the different categories with their statistics in a horizontal format. One final aspect to note is that there were individual containers for the different types of statistics such as 'Attack' and 'Defence' which had equal widths assigned to create a balanced appearance as well as 'flex-direction: column' to position the content vertically without being affected by other set 'flex-direction' styles.

The outcome of this code at the stage can be viewed below. Please note that at this stage, some of the other styles hadn't been changed and that I had already structured the player overview section:

The Outcome – The Statistics Successfully Appeared on the Page for the Correct Player (Example)



After progressing further at a later stage, I then refined the structure and appearance of this page to the following below:

The Changed 'show.blade.php' File

```
@extends('layouts.mufc_app_template1')

<title>MUFC Web App - Individual Players ({{$player->name}})</title>

@section('content')

<div class="player_overview_container">
    <div class="poc_general_information_container">
        <div class="poc_gic_image_container">
            <a href="/mufc_web_app/blog/public/players"><button><i class="far fa-arrow-alt-circle-left"></i> Go Back</button></a>
            <img src="{{$player->image}}" class="poc_gic_images">
        </div>
        <div class="poc_gic_information_container">
            <h1>{{$player->name}}</h1>
            <hr>
            <h2>General Profile</h2>
            <p><b>Age:</b> {{$player->age}}</p>
            <p><b>Nationality:</b> {{$player->nationality}}</p>
            <p><b>Position:</b> {{$player->position}}</p>
        </div>
    </div>
</div>
```

```
<div class="player_statistics_container2">
    <div class="psc2_title_container">
        <h2>2017/18 Season Statistics</h2>
    </div>
    <div class="psc2_categories_container">
        <div class="psc2_cc_attack">
            <h3>Attack</h3>
            <hr>
            <p><b>Goals:</b> {{$player->goals}}</p>
            <p><b>Assists:</b> {{$player->assists}}</p>
        </div>
        <div class="psc2_cc_defence">
            <h3>Defence</h3>
            <hr>
            <p><b>Tackles:</b> {{$player->tackles}}</p>
            <p><b>Saves:</b> {{$player->saves}}</p>
        </div>
        <div class="psc2_cc_general">
            <h3>General Play</h3>
            <hr>
            <p><b>Minutes Played:</b> {{$player->minutes_played}}</p>
            <p><b>Passes:</b> {{$player->passes}}</p>
            <p><b>Shots:</b> {{$player->shots}}</p>
            <p><b>Offsides:</b> {{$player->offsides}}</p>
        </div>
        <div class="psc2_cc_discipline">
            <h3>Discipline</h3>
            <hr>
            <p><b>Fouls:</b> {{$player->fouls}}</p>
            <p><b>Yellow Cards:</b> {{$player->yellow_cards}}</p>
            <p><b>Red Cards:</b> {{$player->red_cards}}</p>
        </div>
    </div>
</div>

@endsection
```

As can be seen above, the aspect relating to the actual player statistics remained the same other than a change of 'class' name for different elements. As the overview section wasn't originally explained, there were two containers relating to showing the player's image and a 'Go Back' button and showing the context/overview of the selected player. This was contained within a parent container called 'player_overview_container' to separate this from the statistics section below this. Again, the template was included with '@section('content')' allowing for entering content unique to this page. One final aspect to note is that with the 'title' aspect of the page, I included '{{$player->name}}' to display the name of the selected player in the tab when viewing the page.

The Changed CSS Code

```css
/* INDIVIDUAL PLAYERS ACTUAL PAGE START */

/* Player Overview Section Start */

.player_overview_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 20px;
}

.poc_general_information_container {
    display: flex;
    flex-direction: row;
    align-items: center;
    width: 100%;
    height: auto;
}

.poc_gic_image_container {
    width: 50%;
    height: auto;
}

.poc_gic_image_container button {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    padding: 10px;
    cursor: pointer;
    transition: 0.5s;
    font-style: italic;
    font-weight: bold;
}
```

```css
.poc_gic_image_container button:hover {
    background-color: #FF0000;
    border: 2px solid #000000;
    transition: 0.5s;
}

.poc_gic_images {
    width: 100%;
    height: auto;
    margin-top: 20px;
}

.poc_gic_information_container {
    width: 50%;
    height: auto;
    padding-left: 20px;
}

.poc_gic_information_container hr {
    width: 100%;
    height: auto;
    border: 2px solid #000000;
}

/* Player Overview Section End */
```

```css
/* Player Statistics Section Start */

.player_statistics_container2 {
    display: flex;
    flex-direction: column;
    padding: 20px;
    width: 100%;
    height: auto;
    background-color: #2A2A2A;
    color: #FFFFFF;
}

.psc2_title_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.psc2_categories_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    text-align: center;
}

.psc2_categories_container hr {
    width: 90%;
    height: auto;
    border: 2px solid #FFFFFF;
}

.psc2_cc_attack {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}
```

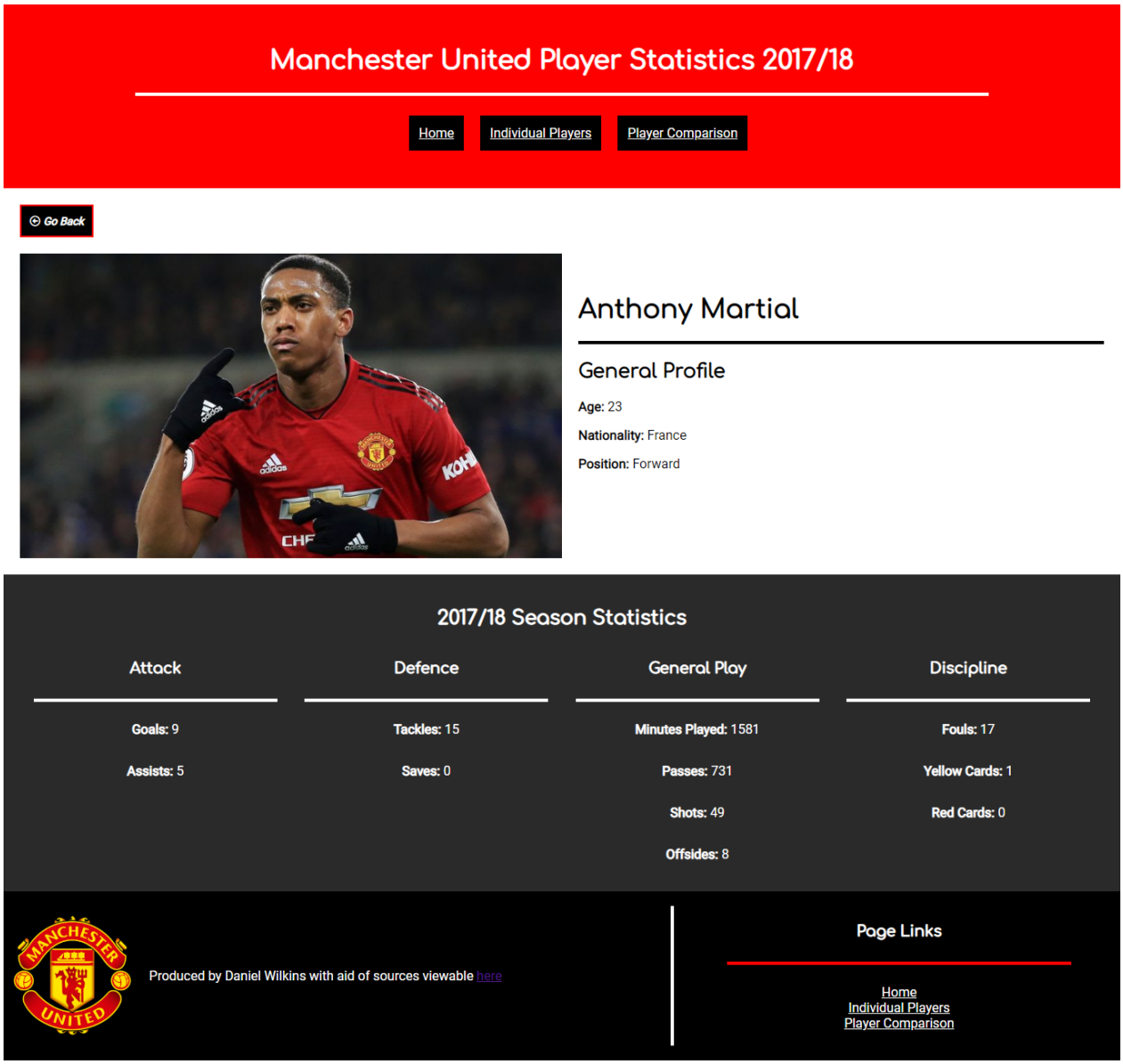
```css
.psc2_cc_defence {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}

.psc2_cc_general {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}

.psc2_cc_discipline {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}

/* Player Statistics Section Start */

/* INDIVIDUAL PLAYERS ACTUAL PAGE END */
```

Again, most of the styles remained the same for statistics section of the page other than changing some colours of aspects and adding styling for a 'hr' tag. Regarding the overview section, a 'flex-direction' of 'row' was assigned to the parent container called 'player_overview_container' and the 'poc_general_information_container' container to display containers inside in a horizontal format, placing the information next to the player's image. Equal widths were assigned to the sections for the player image and information to provide a balanced appearance and 'align-items: center' was applied to the 'poc_general_information_container' to position the text to the centre of the image, creating a better appearance. Furthermore, the button element was styled to change colours when

hovering over, indicating interaction as well as including a 'Font Awesome' icon for this in the 'HTML' code. One final aspect to note is that 'padding' and 'margin' was added, where appropriate, to provide more space between elements, creating a better appearance on the page.

The outcome of this can be seen below:

The Outcome on the Page (Example)



This now signified the end of this page, allowing for progression elsewhere.

Creating the 'Player Comparison' Page

As one of the aspects displayed in the wireframes created was a player comparison page, I wanted to therefore create this. This would attempt to allow users to compare different player's statistics after specifying players to compare.

To begin this task, I created a new 'blade' file as well as a 'controller' and 'route' to be able to display the page on the website application. This process can be viewed below:

Creating the new File and Implementing the Template as well as Some Text to Test

```
player_comparison.blade.php
@extends('layouts.mufc_app_template1')

@section('content')

TEST 123

@endsection
```

Creating the Controller and Editing to Return the Correct View

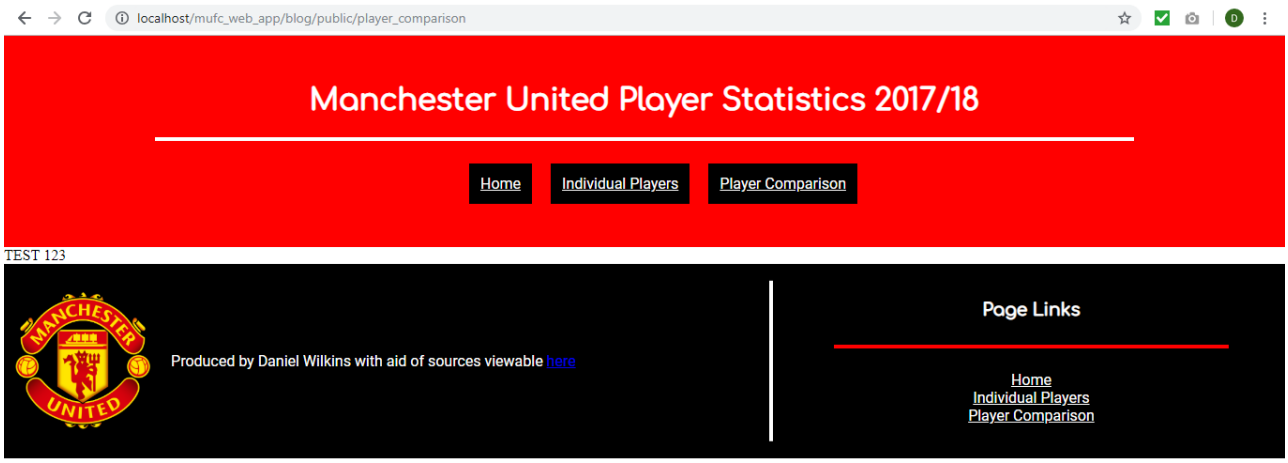
```
C:\MAMP\htdocs\mufc_web_app\blog>php artisan make:controller PlayerComparisonsCo
ntroller --resource
Controller created successfully.
```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PlayerComparisonsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('players.player_comparison');
    }
```

Creating the Route for the new Page in the 'web.php' File

```
Route::get('/player_comparison', 'PlayerComparisonsController@index');
```

The Outcome – This Successfully Returned the View

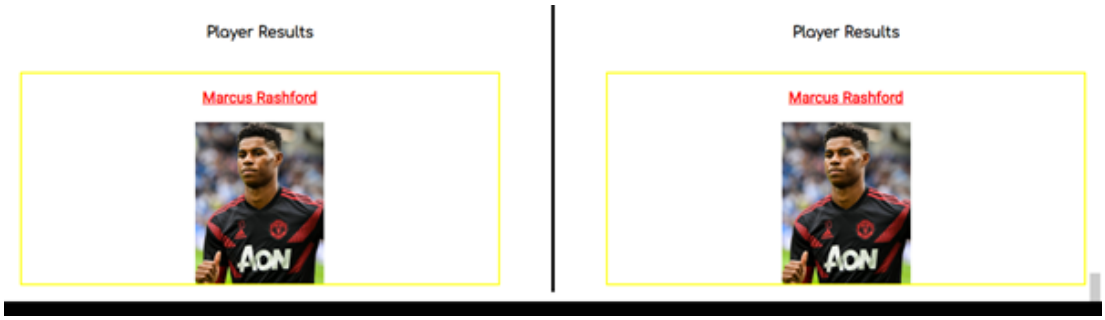


The appearance of this page will be shown and explained at the end of this section. To summarise, at this point, I had managed to integrate the same aspects as that shown for the 'Individual Players' page with filters and a custom search duplicated. This was to allow for specifying players on either side of the page to then compare. I then continued, attempting to allow the custom search to only apply to one aspect of the page. This was because, at this current time, the searched player was appearing on both sides and I only wanted the player to appear on the side that had been used. This is evident below:

Duplicating the Routes for the 'Player Comparisons' Page to allow the Search to Function Correctly

```
/* Player Comparison Page Search and Filters */

Route::any('/player_comparison', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($players) > 0){
        return view('players.player_comparison')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get())
        ->with('positions', Position::orderBy('title', 'asc')->get());
    }
    else return view ('players.player_comparison')
        ->withMessage('No Details found. Try to search again !')
        ->with('players', $players)
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get())
        ->with('positions', Position::orderBy('title', 'asc')->get());
});

Route::any('/player_comparison/ages/{age}', [
    'uses' => 'AgesController@age',
    'as' => 'age'

]);

Route::any('/player_comparison/categories/{category}', [
    'uses' => 'CategoriesController@category',
    'as' => 'category'
]);

Route::any('/player_comparison/positions/{position}', [
    'uses' => 'PositionsController@position',
    'as' => 'position'
]);
```

The Outcome – The Search Worked but Displayed the Same Player for each Side of the Page



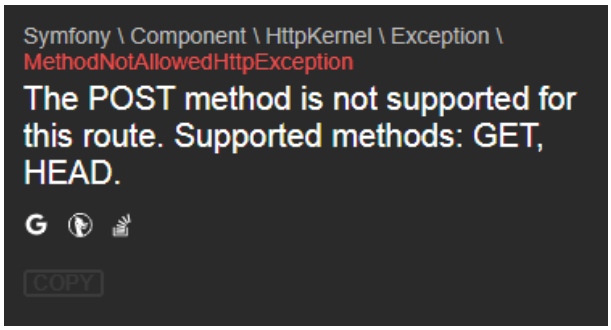
To attempt to resolve this issue, I attempted to integrate an 'ID' as I thought this may work when utilising a '#' in the 'URL' to link to this particular part of the page. However, this didn't work:

Integrating an 'ID' for One Custom Search – Relating to the View and Route

```
<form action="http://localhost/mufc_web_app/blog/public/player_comparison" method="post" id="1">
```

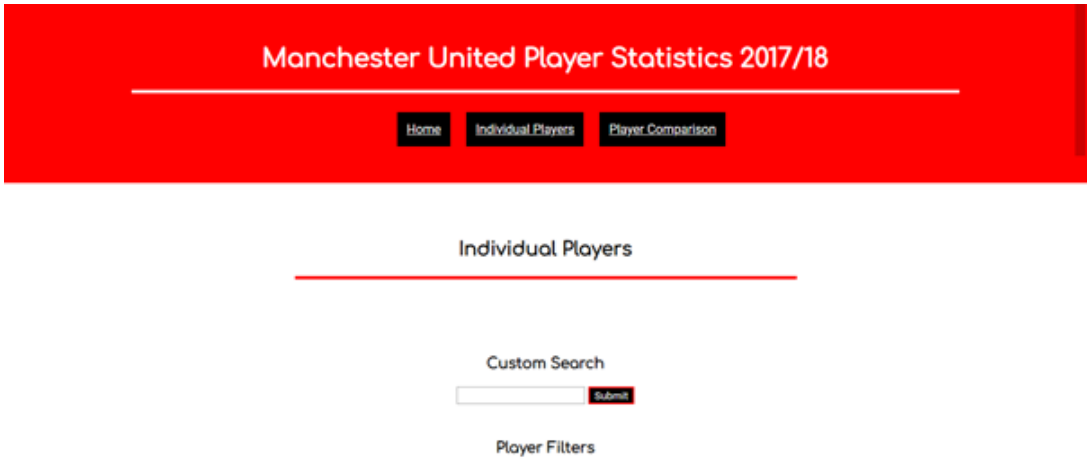
```
Route::any('/player_comparison#1',
```

The Outcome – This Caused an Error to Appear



As, at this current time, I couldn't manage to resolve this issue, I therefore attempted to resolve the issue when selecting the filters on the page. This was because this was causing navigation to 'Individual Players' page with the selected filters instead of remaining on the same page as is evident below:

Selecting the Filter would Navigate Myself to the 'Individual Players' Page



To try to resolve this issue, I added another 'return view' statement to the function relating to the player's position in the 'PositionsController.php' file to test if this would work before applying this to other filter aspects. This was because I thought this would help to return the 'Player Comparison' page also. However, this was unsuccessful as the same issue occurred as before:

Adding Another 'return view' Statement

```
public function position($id)
{
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $positions = Position::with('players')->orderBy('title', 'asc')->get();
    $players = Player::orderBy('name')->get()->where('position_id', $id);
    return view('players.index', compact('players', 'categories', 'ages', 'positions'));
    return view('players.player_comparison', compact('players', 'categories', 'ages', 'positions'));
}
```

Following on from the previously failed attempt, I then thought that duplicating the public function would help, changing the name as I knew that I couldn't declare the same function twice. Therefore, I then undertook this process for only the aspect relating to the positions before applying this to other elements to test if this would work. I also changed the 'return view' to return the correct page:

Duplicating the Same Function as well as Changing other Aspects to Relate to this – The 'PlayerComparisonsController.php' File

```
public function index()
{
    $players = Player::orderBy('name', 'asc')->get();
    $categories = Category::with('players')->orderBy('title', 'asc')->get();
    $ages = Age::with('players')->orderBy('title', 'asc')->get();
    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();

    return view('players.player_comparison', compact('players', 'categories', 'ages', 'positions2'));
}
```

Duplicating the Same Function as well as Changing other Aspects to Relate to this – The 'PositionsController.php' File

```
public function position2($id)
{

    $ages = Age::with('players')->orderBy('title', 'asc')->get();

    $categories = Category::with('players')->orderBy('title', 'asc')->get();

    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();

    $players = Player::orderBy('name')->get()->where('position_id', $id);

    return view('players.player_comparison', compact('players', 'categories', 'ages', 'positions2'));

}
```

Duplicating the Same Function as well as Changing other Aspects to Relate to this – The 'player_comparison.blade.php' File

```
<div class="pccs1_position_filter_container">
    <h4>Position</h4>
    <ul>
    @foreach ($positions2 as $position2)
        <a href="{{ route('position2', $position2->id) }}"><li>{{ $position2->title }}</li></a>
    @endforeach
    </ul>
</div>
```

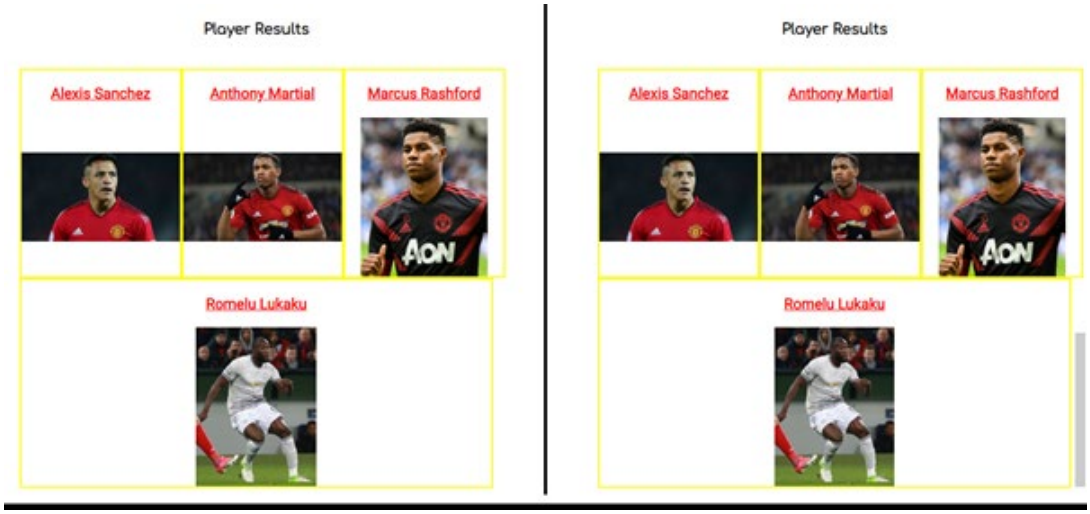
```
<div class="pccs2_position_filter_container">
    <h4>Position</h4>
    <ul>
    @foreach ($positions2 as $position2)
        <a href="{{ route('position2', $position2->id) }}"><li>{{ $position2->title }}</li></a>
    @endforeach
    </ul>
</div>
```

Duplicating the Same Function as well as Changing other Aspects to Relate to this – The 'web.php' File (Routes)

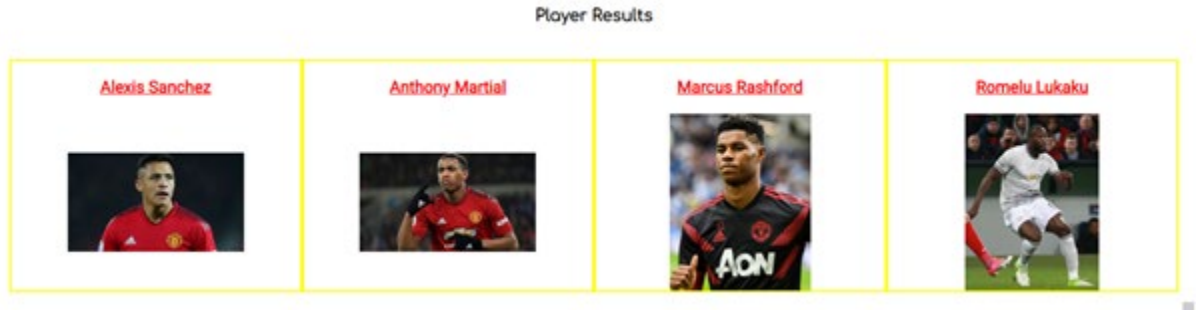
```
Route::any('/player_comparison', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($players) > 0){
        return view('players.player_comparison')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get())
        ->with('positions2', Position::orderBy('title', 'asc')->get());
    }
    else return view ('players.player_comparison')
        ->withMessage('No Details found. Try to search again !')
        ->with('players', $players)
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get())
        ->with('positions2', Position::orderBy('title', 'asc')->get());
});

Route::any('/player_comparison/ages/{age}', [
    'uses' => 'AgesController@age',
    'as' => 'age'

]);

Route::any('/player_comparison/categories/{category}', [
    'uses' => 'CategoriesController@category',
    'as' => 'category'
]);

Route::any('/positions2/{position2}', [
    'uses' => 'PositionsController@position2',
    'as' => 'position2'
]);
```

After testing this to see if this had successfully worked, this had as will be evident below. This also didn't affect the 'Individual Players' page position filter set either:

The Outcome – This Worked



The Outcome – This Didn't Affect the Other Filter Set



As this had now worked, I then undertook the same process again for the ages and nationalities filter sets:

The 'CategoriesController.php' File

```
public function category2($id)
{
    $categories2 = Category::with('players')->orderBy('title', 'asc')->get();
    $ages2 = Age::with('players')->orderBy('title', 'asc')->get();
    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();
    $players = Player::orderBy('name')->get()->where('category_id', $id);
    return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));
}
```

The 'AgesController.php' File

```
public function age2($id)
{
    $ages2 = Age::with('players')->orderBy('title', 'asc')->get();
    $categories2 = Category::with('players')->orderBy('title', 'asc')->get();
    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();
    $players = Player::orderBy('name')->get()->where('age_id', $id);
    return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));
}
```

The 'PlayerComparisonsController.php' File

```
public function index()
{
    $players = Player::orderBy('name', 'asc')->get();
    $categories2 = Category::with('players')->orderBy('title', 'asc')->get();
    $ages2 = Age::with('players')->orderBy('title', 'asc')->get();
    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();
    return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));
}
```

The 'web.php' File (Routes)

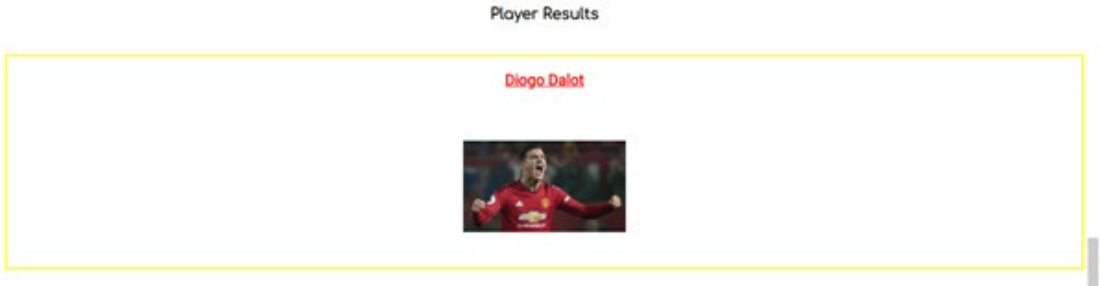
```
Route::any('/ages2/{age2}', [
    'uses' => 'AgesController@age2',
    'as' => 'age2'
]);

Route::any('/categories2/{category2}', [
    'uses' => 'CategoriesController@category2',
    'as' => 'category2'
]);
```

The 'player_comparison.blade.php' File

```
<div class="pccs2_fsc_main_container">
    <div class="pccs2_age_filter_container">
    <h4>Age</h4>
        <ul>
        @foreach ($ages2 as $age2)
            <a href="{{ route('age2', $age2->id) }}"><li>{{ $age2->title }}</li></a>
        @endforeach
        </ul>
    </div>

    <div class="pccs2_nationality_filter_container">
        <h4>Nationality</h4>
        <ul>
        @foreach ($categories2 as $category2)
            <a href="{{ route('category2', $category2->id) }}"><li>{{ $category2->title }}</li></a>
        @endforeach
        </ul>
    </div>
```

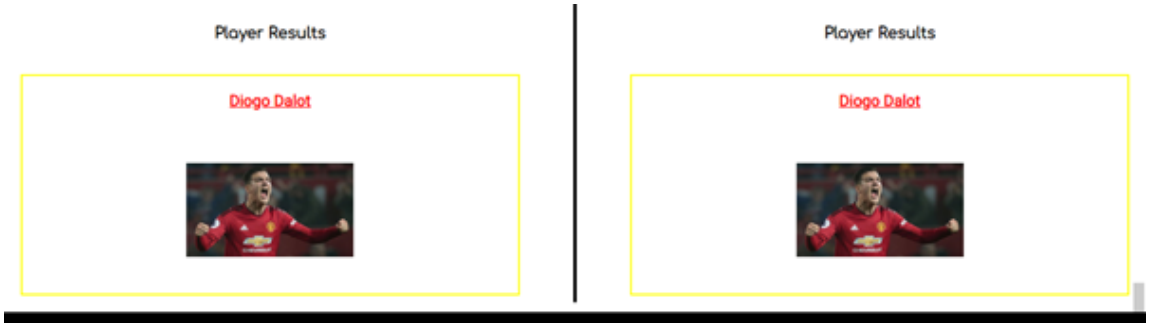
```
<div class="pccs1_fsc_main_container">
    <div class="pccs1_age_filter_container">
    <h4>Age</h4>
        <ul>
        @foreach ($ages2 as $age2)
            <a href="{{ route('age2', $age2->id) }}"><li>{{ $age2->title }}</li></a>
        @endforeach
        </ul>
    </div>

    <div class="pccs1_nationality_filter_container">
        <h4>Nationality</h4>
        <ul>
        @foreach ($categories2 as $category2)
            <a href="{{ route('category2', $category2->id) }}"><li>{{ $category2->title }}</li></a>
        @endforeach
        </ul>
    </div>
```

The Outcome After Selecting a Filter – The 'Individual Players' Page Wasn't Affected – Age Filter



The Outcome After Selecting a Filter – The 'Individual Players' Page Wasn't Affected – Nationality Filter

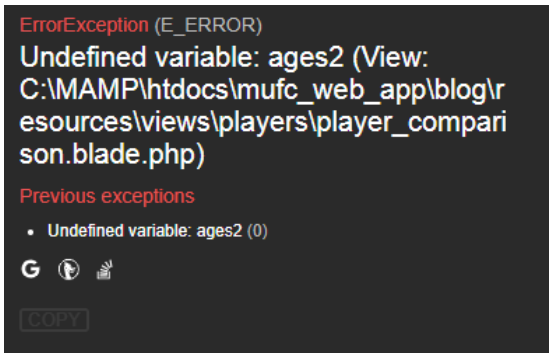
The Outcome After Selecting a Filter – The 'Player Comparison' Page Was Working – Age Filter



The Outcome After Selecting a Filter – The 'Player Comparison' Page Was Working – Nationality Filter

As I had now allowed for the filters to work, I then had to resolve a couple of issues which can be viewed below:

Encountered Issue 1 – A Variable was Undefined



Encountered Issue 2 – When Selecting the 'Goalkeeper' Position option, this Navigated Myself to the 'Player Comparison' Page from the 'Individual Players' Page

Resolving the first issue also helped to resolve the second issue shown above. To resolve these issues, I added in the missing variables, as seen below:

Adding the Missing Variables

```
public function position2($id)
{
    $ages2 = Age::with('players')->orderBy('title', 'asc')->get();
    $categories2 = Category::with('players')->orderBy('title', 'asc')->get();
    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();
    $players = Player::orderBy('name')->get()->where('position_id', $id);
    return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));
}
```

This then fixed the problems I was experiencing:

The Players now Appeared on the 'Player Comparison' Page

The User Remained on the 'Individual Players' Page when Selecting the 'Goalkeeper' Position

As I had now managed to allow for filters and a custom search to work on the 'Player Comparison' page, I now needed to allow for applying each search to one aspect only on the page. However, despite researching how to complete this, I was unable to integrate anything which could work. Therefore, I decided to leave this and progress with the appearance of the page, adding in statistics for each player in a certain format. This way, users would still be able to see player statistics next to each other.

The improved appearance can be viewed below:

The Updated 'player_comparison.blade.php' File

```
@extends('layouts.mufc_app_template1')

<title>MUFC Web App - Player Comparison</title>

@section('content')

<div class="page_heading_container">
    <div class="page_heading_sub_container">
        <h2>Player Comparison</h2>
        <hr>
    </div>
</div>

<div class="player_comparison_container">
    <div class="pcc_section1">
            <div class="pccs1_filter_container">
                <div class="pccs1_typed_search_container">
                    <h3>Custom Search</h3>
                    <form action="http://localhost/mufc_web_app/blog/public/player_comparison" method="post">
                        {{ csrf_field() }}
                        <label for="query"></label>
                        <input type="query" id="query" name="query">
                        <input type="submit" class="pccs1_submit_styles">
                    </form>
                </div>
                <div class="pccs1_filters_search_container">
                    <div class="pccs1_fsc_title_container">
                        <h3>Player Filters</h3>
                    </div>
                    <div class="pccs1_fsc_main_container">
                        <div class="pccs1_age_filter_container">
                        <h4>Age</h4>
                            <ul>
                            @foreach ($ages2 as $age2)
                                <a href="{{ route('age2', $age2->id) }}"><li>{{ $age2->title }}</li></a>
                            @endforeach
                            </ul>
                        </div>
```

```
                        <div class="pccs1_nationality_filter_container">
                            <h4>Nationality</h4>
                            <ul>
                            @foreach ($categories2 as $category2)
                                <a href="{{ route('category2', $category2->id) }}"><li>{{ $category2->title }}</li></a>
                            @endforeach
                            </ul>
                        </div>

                        <div class="pccs1_position_filter_container">
                            <h4>Position</h4>
                            <ul>
                            @foreach ($positions2 as $position2)
                                <a href="{{ route('position2', $position2->id) }}"><li>{{ $position2->title }}</li></a>
                            @endforeach
                            </ul>
                        </div>
                    </div>
                </div>
            </div>
            <div class="pcc_mrcs1_title_container">
                <h3>Player Results</h3>
            </div>

            @if(count($players) > 0)
            @foreach($players->chunk(3) as $chunk)
                    <div class="pcc_mrcs1_players">
                    @foreach($chunk as $player)
                        <div class="pcc_mrcs1_players2">
                            <h3>{{$player->name}}</a></h3>
                            <img src="{{$player->image}}" class="pcc_mrcs1_player_images">
                            <div class="pcc_mrsc1_statistic_containers">
                                <p><b>Minutes Played:</b> {{$player->minutes_played}}</p>
                            </div>
                            <div class="pcc_mrsc1_statistic_containers">
                                <p><b>Goals:</b> {{$player->goals}}</p>
                            </div>
                            <div class="pcc_mrsc1_statistic_containers">
                                <p><b>Assists:</b> {{$player->assists}}</p>
                            </div>
                            <div class="pcc_mrsc1_statistic_containers">
                                <p><b>Tackles:</b> {{$player->tackles}}</p>
                            </div>
```

```
                        <div class="pcc_mrsc1_statistic_containers">
                            <p><b>Saves:</b> {{$player->saves}}</p>
                        </div>
                        <div class="pcc_mrsc1_statistic_containers">
                            <p><b>Passes:</b> {{$player->passes}}</p>
                        </div>
                        <div class="pcc_mrsc1_statistic_containers">
                            <p><b>Shots:</b> {{$player->shots}}</p>
                        </div>
                        <div class="pcc_mrsc1_statistic_containers">
                            <p><b>Offsides:</b> {{$player->offsides}}</p>
                        </div>
                        <div class="pcc_mrsc1_statistic_containers">
                            <p><b>Fouls:</b> {{$player->fouls}}</p>
                        </div>
                        <div class="pcc_mrsc1_statistic_containers">
                            <p><b>Yellow Cards:</b> {{$player->yellow_cards}}</p>
                        </div>
                        <div class="pcc_mrsc1_statistic_containers">
                            <p><b>Red Cards:</b> {{$player->red_cards}}</p>
                        </div>
                    </div>
                @endforeach
                </div>
        @endforeach
        @else
            <p>No players found</p>
        @endif
    </div>

    <div class="pcc_hr_section">
        <hr>
    </div>
```

```
<div class="pcc_section2">
        <div class="pccs2_filter_container">
            <div class="pccs2_typed_search_container">
                <h3>Custom Search</h3>
                <form action="http://localhost/mufc_web_app/blog/public/player_comparison" method="post">
                    {{ csrf_field() }}
                    <label for="query"></label>
                    <input type="query" id="query" name="query">
                    <input type="submit" class="pccs2_submit_styles">
                </form>
            </div>
            <div class="pccs2_filters_search_container">
                <div class="pccs2_fsc_title_container">
                    <h3>Player Filters</h3>
                </div>
                <div class="pccs2_fsc_main_container">
                    <div class="pccs2_age_filter_container">
                    <h4>Age</h4>
                        <ul>
                        @foreach ($ages2 as $age2)
                            <a href="{{ route('age2', $age2->id) }}"><li>{{ $age2->title }}</li></a>
                        @endforeach
                        </ul>
                    </div>

                    <div class="pccs2_nationality_filter_container">
                        <h4>Nationality</h4>
                        <ul>
                        @foreach ($categories2 as $category2)
                            <a href="{{ route('category2', $category2->id) }}"><li>{{ $category2->title }}</li></a>
                        @endforeach
                        </ul>
                    </div>

                    <div class="pccs2_position_filter_container">
                        <h4>Position</h4>
                        <ul>
                        @foreach ($positions2 as $position2)
                            <a href="{{ route('position2', $position2->id) }}"><li>{{ $position2->title }}</li></a>
                        @endforeach
                        </ul>
                    </div>
                </div>
            </div>
```

```
            </div>
            <div class="pcc_mrcs2_title_container">
                <h3>Player Results</h3>
            </div>

            @if(count($players) > 0)
            @foreach($players->chunk(3) as $chunk)
                    <div class="pcc_mrcs2_players">
                    @foreach($chunk as $player)
                        <div class="pcc_mrcs2_players2">
                            <h3>{{$player->name}}</a></h3>
                            <img src="{{$player->image}}" class="pcc_mrcs2_player_images">
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Minutes Played:</b> {{$player->minutes_played}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Goals:</b> {{$player->goals}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Assists:</b> {{$player->assists}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Tackles:</b> {{$player->tackles}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Saves:</b> {{$player->saves}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Passes:</b> {{$player->passes}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Shots:</b> {{$player->shots}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Offsides:</b> {{$player->offsides}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Fouls:</b> {{$player->fouls}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Yellow Cards:</b> {{$player->yellow_cards}}</p>
                            </div>
                            <div class="pcc_mrsc2_statistic_containers">
                                <p><b>Red Cards:</b> {{$player->red_cards}}</p>
                            </div>
                            <div
                        </div>
                    @endforeach
                    </div>
            @endforeach
            @else
                <p>No players found</p>
            @endif
        </div>

</div>

@endsection
```

As is evident above, this was code duplicated from the 'Individual Players' page where there would be filter/search and player sections for each side of the page. However, containers were added to separate each type of statistic as well as integrating the data into the page to allow users to compare different players. For this page, there were three main sections. These were the page heading aspect, the section to allow users to search for players on the left and the section to allow users to search for players on the right.

The CSS Code for this Page

```css
/* PLAYER COMPARISON PAGE START */

.player_comparison_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 20px;
    background: white;
    color: black;
}

/* Player Comparison Section 1 Start */

.pcc_section1 {
    display: flex;
    flex-direction: column;
    width: 45%;
    height: auto;
}

/* Player Comparison Section 1 - Search/Filters Section Start */

.pccs1_filter_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
}

.pccs1_filter_container a {
    color: #FF0000;
}

.pccs1_typed_search_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.pccs1_typed_search_container input {
    font-family: 'Roboto', sans-serif;
}
```

```css
.pccs1_submit_styles {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    cursor: pointer;
}

.pccs1_filters_search_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    padding: 10px;
}

.pccs1_fsc_title_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.pccs1_fsc_main_container {
    width: 100%;
    height: auto;
    display: flex;
    flex-direction: row;
}

.pccs1_filters_search_container ul {
    list-style-type: none;
    padding: 0;
}

.pccs1_filters_search_container li {
    padding: 10px;
    border: 2px solid #000000;
    margin-top: 10px;
}
```

```css
.pccs1_age_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

.pccs1_nationality_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

.pccs1_position_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

/* Player Comparison Section 1 - Search/Filters Section End */

/* Player Comparison Section 1 - Main Results Section Start */

.pcc_mrcs1_title_container {
    width: 100%;
    height: auto;
    text-align: center;
    margin-bottom: 20px;
}

.pcc_mrcs1_players {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
}
```

```css
.pcc_mrcs1_players2 {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    border: 2px solid yellow;
    text-align: center;
}

.pcc_mrcs1_players2 a {
    color: #FF0000;
    transition: 0.5s;
}

.pcc_mrcs1_players2 a:hover {
    color: #000000;
    transition: 0.5s;
}

.pcc_mrcs1_player_images {
    width: 200px;
    height: 200px;
    object-fit: contain;
    margin: auto;
}

.pcc_mrsc1_statistic_containers {
    width: 100%;
    height: auto;
    border: 2px solid #000000;
    margin-top: 10px;
}

/* Player Comparison Section 1 - Main Results Section End */

/* Player Comparison Section 1 End */
```

```css
/* Player Comparison Section 2 Start */

.pcc_section2 {
    display: flex;
    flex-direction: column;
    width: 45%;
    height: auto;
}

/* Player Comparison Section 2 - Search/Filters Section Start */

.pccs2_filter_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
}

.pccs2_filter_container a {
    color: #FF0000;
}

.pccs2_typed_search_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.pccs2_typed_search_container input {
    font-family: 'Roboto', sans-serif;
}

.pccs2_submit_styles {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    cursor: pointer;
}
```

```css
.pccs2_filters_search_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    padding: 10px;
}

.pccs2_fsc_title_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.pccs2_fsc_main_container {
    width: 100%;
    height: auto;
    display: flex;
    flex-direction: row;
}

.pccs2_filters_search_container ul {
    list-style-type: none;
    padding: 0;
}

.pccs2_filters_search_container li {
    padding: 10px;
    border: 2px solid #000000;
    margin-top: 10px;
}

.pccs2_age_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

.pccs2_nationality_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}
```

```css
.pccs2_position_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

/* Player Comparison Section 2 - Search/Filters Section End */

/* Player Comparison Section 2 - Main Results Section Start */

.pcc_mrcs2_title_container {
    width: 100%;
    height: auto;
    text-align: center;
    margin-bottom: 20px;
}

.pcc_mrcs2_players {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
}

.pcc_mrcs2_players2 {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    border: 2px solid yellow;
    text-align: center;
}

.pcc_mrcs2_players2 a {
    color: #FF0000;
    transition: 0.5s;
}

.pcc_mrcs2_players2 a:hover {
    color: #000000;
    transition: 0.5s;
}
```

```css
.pcc_mrcs2_players2 a:hover {
    color: #000000;
    transition: 0.5s;
}

.pcc_mrcs2_player_images {
    width: 200px;
    height: 200px;
    object-fit: contain;
    margin: auto;
}

.pcc_mrsc2_statistic_containers {
    width: 100%;
    height: auto;
    border: 2px solid #000000;
    margin-top: 10px;
}

/* Player Comparison Section 2 - Main Results Section End */

/* hr Section Start */

.pcc_hr_section {
    width: 10%;
    height: auto;
}

.pcc_hr_section hr {
    width: 1%;
    height: 100%;
    transform: rotate(180deg);
    border: 2px solid #000000;
}

/* hr Section End */

/* Player Comparison Section 2 - Main Results Section End */

/* PLAYER COMPARISON PAGE END */
```

As will have been noticeable above, the page heading section was already styled within another page as has been explained before. Regarding the main section of the page, the parent container 'player_comparison_container' allowed for displaying all containers inside in a row format through 'flex-direction: row'. Regarding the parent containers for each section on the page called 'pcc_section1' and 'pcc_section2', these were assigned 'flex-direction: column' to display the players underneath the search sections. Furthermore, 'flex-direction: column' was also assigned to the parent containers for the search sections to allow for the filters to be displayed underneath the custom search aspects. One final aspect to note regarding the search sections is that 'flex-direction:

row' was assigned to the containers called 'pccs1_fsc_main_container' 'pccs2_fsc_main_container' display the filters in a row format.

Regarding the players sections, these were assigned similar styles as before from the 'Individual Players' page. However, within the 'blade' file, the 'chunk' method was set to display 3 players per row as 5 would cause an unattractive appearance.

One final key aspect to note is the fact that I also added styles for the 'hr' section, utilising aspects such as 'height: 100%' and 'transform: rotate(180deg)' to display the line vertically. This helped to create a divide in the middle of the page between the two other sections.

The outcome of this can be viewed on the following page.

The Outcome – This Successfully Structured and Styled the Page

This now signified the end of the task for this page.

Making the Pages Responsive

As I had created some mobile wireframes, I therefore thought it would be beneficial to make the website application fit onto mobile device screen sizes. The process of this for each page can be viewed below. The main aspect to note is that I changed the 'flex-direction' to display most elements, if not all, in a vertical format to match that of mobile devices.

The Header and Footer Sections of Each Page

I changed the 'CSS' code to the following shown below. The outcome will be viewable on the different displayed pages within this section:

The Changed Code

```
/* Header Section Start */

.header_navigation_container {
    display: flex;
    justify-content: center;
    width: 100%;
    height: auto;
}

.header_navigation_container ul {
    display: flex;
    flex-direction: column;
    padding: 0;
}

.header_navigation_container li {
    margin-top: 10px;
    margin-left: 0px;
    margin-right: 0px;
}

/* Header Section End */
```

```
/* Footer Section Start */

.footer_container {
    display: flex;
    flex-direction: column;
}

.fc_sub_container1 {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    text-align: center;
}

.fc_logo_container {
    width: 30%;
    height: auto;
}

.fc_sources_container {
    width: 100%;
    height: auto;
    padding-left: 20px;
    padding-right: 20px;
}

.fc_sub_container2 {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    padding-right: 0px;
}
```

```
.fc_hr_container {
    width: 100%;
    height: auto;
}

.fc_hr_container hr {
    width: 100%;
    height: auto;
    transform: rotate(0deg);
}

.fc_plsc_heading_container hr {
    width: 50%;
    height: auto;
}

/* Footer Section End */
```

The 'Home/Introduction' Page

I changed the 'CSS' code to the following shown below with the outcome:

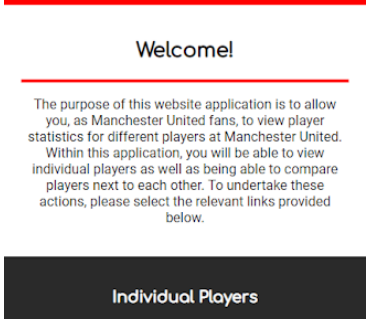
The Changed Code

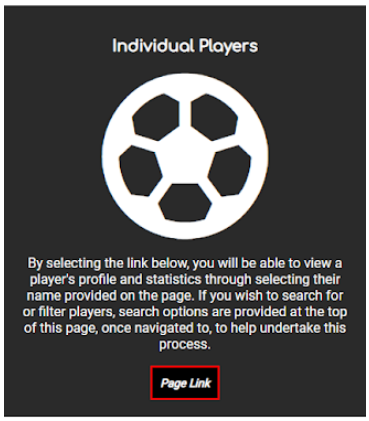
```css
/* HOME PAGE START */

/* Welcome Section Start */

.page_welcome_container {
    padding: 20px;
}

.page_welcome_sub_container {
    width: 100%;
    height: auto;
}

/* Welcome Section End */

/* Individual Players Section Start */

.individual_players_container {
    display: flex;
    flex-direction: column;
    padding: 20px;
}

.individual_players_sub_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
}

.ipc_title_container {
    width: 100%;
    height: auto;
}

.ipc_title_container i {
    font-size: 200px;
}
```

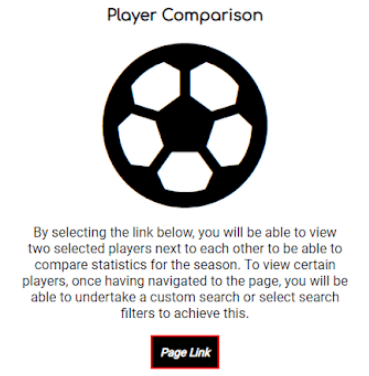
```css
.ipc_description_container {
    width: 100%;
    height: auto;
    text-align: center;
}

/* Individual Players Section End */

/* Player Comparison Section Start */

.player_comparison_container {
    display: flex;
    flex-direction: column;
    padding: 20px;
}

.player_comparison_sub_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
}

.pcc_title_container {
    width: 100%;
    height: auto;
    order: 1;
}

.pcc_title_container i {
    font-size: 200px;
}

.pcc_description_container {
    width: 100%;
    height: auto;
    text-align: center;
    order: 2;
}

/* Player Comparison Section End */

/* HOME PAGE END */
```

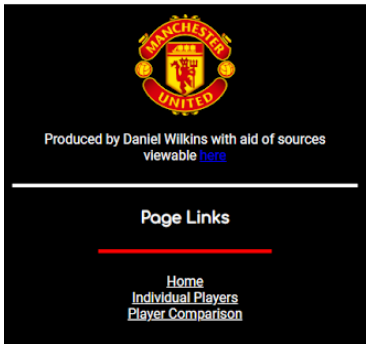
The Outcome


Manchester United
Player Statistics
2017/18
Home
Individual Players
Player Comparison


Welcome!
The purpose of this website application is to allow you, as Manchester United fans, to view player statistics for different players at Manchester United. Within this application, you will be able to view individual players as well as being able to compare players next to each other. To undertake these actions, please select the relevant links provided below.


Individual Players
By selecting the link below, you will be able to view a player's profile and statistics through selecting their name provided on the page. If you wish to search for or filter players, search options are provided at the top of this page, once navigated to, to help undertake this process.
Page Link


Player Comparison
By selecting the link below, you will be able to view two selected players next to each other to be able to compare statistics for the season. To view certain players, once having navigated to the page, you will be able to undertake a custom search or select search filters to achieve this.
Page Link


Produced by Daniel Wilkins with aid of sources viewable here
Page Links
Home
Individual Players
Player Comparison

The 'Individual Players' Page

I changed the 'CSS' code to the following shown below with the outcome:

The Changed Code

```
/* INDIVIDUAL PLAYERS PAGE START */

/* Page Heading Section Start */

.page_heading_container {
    padding: 20px;
}

.page_heading_sub_container {
    width: 100%;
    height: auto;
}

/* Page Heading Section End */

/* Main Content Section Start */

/* Main Content Section - Players Section Start */

.psc_player_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    text-align: center;
    border: 2px solid blue;
}

.psc_pc_players {
    display: flex;
    flex-direction: column;
}

.psc_pc_players2 {
    padding: 30px;
}

.psc_pc_player_images {
    width: 100%;
    height: auto;
}

/* Main Content Section - Players Section Start */

/* Main Content Section End */
```

The Outcome


Manchester United Player Statistics 2017/18
Home
Individual Players
Player Comparison
Individual Players
Custom Search
Submit
Player Filters
Age
18-20
21-22
23-24
25-26
27-28
29-30
31-32
33-34
35-36
Nationality
Argentina
Belgium
Brazil
Chile
Ecuador
England
France
Italy
Ivory Coast
Portugal
Scotland
Serbia
Spain
Sweden
Position
Defender
Forward
Goalkeeper
Midfielder
Player Results
Diogo Dalot
Produced by Daniel Wilkins with aid of sources viewable here
Page Links
Home
Individual Players
Player Comparison

The 'show' Page (Actual Individual Players with their Statistics)

I changed the 'CSS' code to the following shown below with the outcome:

The Changed Code

```
/* INDIVIDUAL PLAYERS ACTUAL PAGE START */

.player_overview_container {
    display: flex;
    flex-direction: column;
    padding: 20px;
}

.poc_general_information_container {
    display: flex;
    flex-direction: column;
}

.poc_gic_image_container {
    width: 100%;
    height: auto;
}

.poc_gic_information_container {
    width: 100%;
    height: auto;
    padding-left: 0px;
}

.psc2_categories_container {
    display: flex;
    flex-direction: column;
}

.psc2_cc_attack {
    width: 100%;
    height: auto;
}

.psc2_cc_defence {
    width: 100%;
    height: auto;
}

.psc2_cc_general {
    width: 100%;
    height: auto;
}
```

```
.psc2_cc_discipline {
    width: 100%;
    height: auto;
}

/* INDIVIDUAL PLAYERS ACTUAL PAGE END */
```

The Outcome


Manchester United Player Statistics 2017/18
Home
Individual Players
Player Comparison
Go Back
Alexis Sanchez
General Profile
Age: 30
Nationality: Chile
Position: Forward
2017/18 Season Statistics
Attack
Goals: 2
Assists: 3
Defence
Tackles: 21
Saves: 0
General Play
Minutes Played: 1046
Passes: 582
Shots: 19
Offsides: 10
Discipline
Fouls: 10
Yellow Cards: 1
Red Cards: 0
Produced by Daniel Wilkins with aid of sources viewable here
Page Links
Home
Individual Players
Player Comparison

The 'Player Comparison' Page

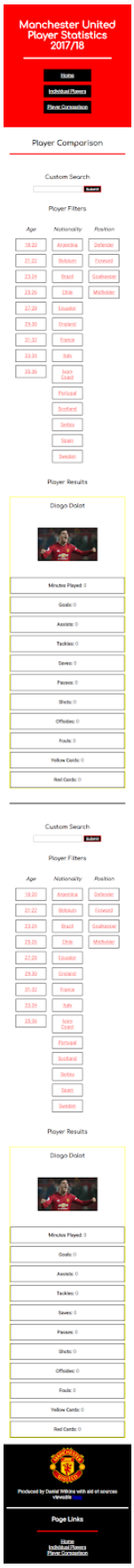
I changed the 'CSS' code to the following shown below with the outcome:

The Changed Code

```css
/* ACTUAL PLAYER COMPARISON PAGE START */

.player_comparison_container {
    display: flex;
    flex-direction: column;
}

/* Player Comparison Section 1 Start */

.pcc_section1 {
    width: 100%;
    height: auto;
}

/* Player Comparison Section 1 - Search/Filters Section Start */

.pccs1_fsc_main_container {
    width: 100%;
    height: auto;
    display: flex;
    flex-direction: row;
}

/* Player Comparison Section 1 - Search/Filters Section End */

/* Player Comparison Section 1 - Main Results Section Start */

.pcc_mrcs1_players {
    display: flex;
    flex-direction: column;
}

/* Player Comparison Section 1 - Main Results Section End */

/* Player Comparison Section 1 End */
```

```css
/* Player Comparison Section 2 Start */

.pcc_section2 {
    width: 100%;
    height: auto;
}

/* Player Comparison Section 2 - Search/Filters Section Start */

.pccs2_filter_container {
    margin-top: 40px;
}

/* Player Comparison Section 2 - Search/Filters Section End */

/* Player Comparison Section 2 - Main Results Section Start */

.pcc_mrcs2_title_container {
    width: 100%;
    height: auto;
    text-align: center;
    margin-bottom: 20px;
}

.pcc_mrcs2_players {
    display: flex;
    flex-direction: column;
}

/* Player Comparison Section 2 - Main Results Section End */

/* hr Section Start */

.pcc_hr_section {
    width: 100%;
    height: auto;
    margin-top: 40px;
}

.pcc_hr_section hr {
    width: 100%;
    height: auto;
    transform: rotate(0deg);
}

/* hr Section End */
```

The Outcome


Manchester United Player Statistics 2017/18
Home
Individual Players
Player Comparison
Player Comparison
Custom Search
Submit
Player Filters
Age
Nationality
Position
Player Results
Diogo Dalot
Minutes Played: 0
Goals: 0
Assists: 0
Tackles: 0
Saves: 0
Passes: 0
Shots: 0
Offsides: 0
Fouls: 0
Yellow Cards: 0
Red Cards: 0
Custom Search
Submit
Player Filters
Age
Nationality
Position
Player Results
Diogo Dalot
Minutes Played: 0
Goals: 0
Assists: 0
Tackles: 0
Saves: 0
Passes: 0
Shots: 0
Offsides: 0
Fouls: 0
Yellow Cards: 0
Red Cards: 0
Page Links
Home
Individual Players
Player Comparison

Examples of Additional Elements Added

Resolving the Issue with the Logo Aspect not Displaying on Certain Pages

Whilst refining the appearance of the web pages, I had noticed that the Manchester United logo hadn't transferred across to each page as will be evident below:

The Issue with the Logo not Appearing (Example)

I then decided to change the 'URL' contained within the template file to the following as this would help navigate to the image on any page:

Changing the 'URL'

```
<a href="http://localhost/mufc_web_app/blog/public/test"><img src="http://localhost/mufc_web_app/blog/public/
images/man_utd_logo.png" class="footer_logo"></a>
```

The Outcome – This Resolved the Issue (Example)

Adding 'Google Fonts'

I also added 'Google Fonts' to match that chosen before starting the development process, integrating this for different elements such as 'h1' and 'p' elements. This process can be viewed below:

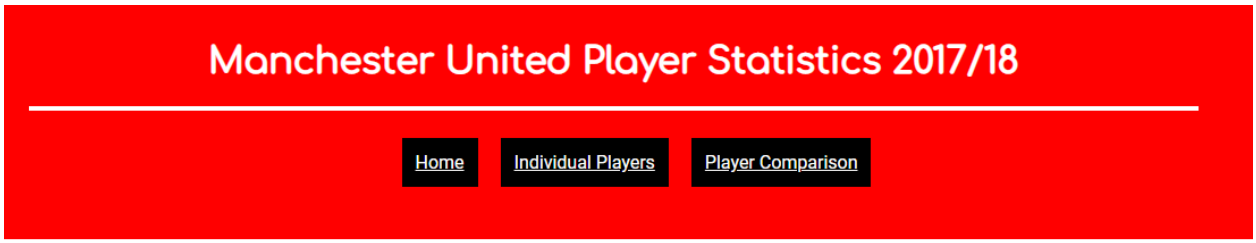
Adding the Link to the Stylesheet in the Template File

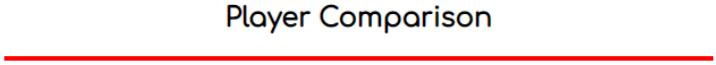
```
<link href="https://fonts.googleapis.com/css?family=Comfortaa|Roboto" rel="stylesheet">
```

Specifying the Fonts in the CSS File

```
/* GENERAL STYLES START */

h1 {
    font-family: 'Comfortaa', cursive;
}

h2 {
    font-family: 'Comfortaa', cursive;
}

h3 {
    font-family: 'Comfortaa', cursive;
}

h4 {
    font-family: 'Comfortaa', cursive;
    font-style: italic;
}

p {
    font-family: 'Roboto', sans-serif;
}

button {
    font-family: 'Roboto', sans-serif;
}

a {
    font-family: 'Roboto', sans-serif;
}

/* GENERAL STYLES END */
```

Examples of the Outcome





Deploying the Manchester United Website Application and the Final Code

Deploying the Manchester United Website Application

As I had been developing the application on my old laptop using 'localhost', I therefore needed to upload this to my own server to show the final outcome for submission. As I was inexperienced in how to do this, I undertook research, following a tutorial online.

The first stage involved exporting the database into a 'SQL' file format to then import onto a created database on my server. This would allow for pulling the data into the application whilst on my server.

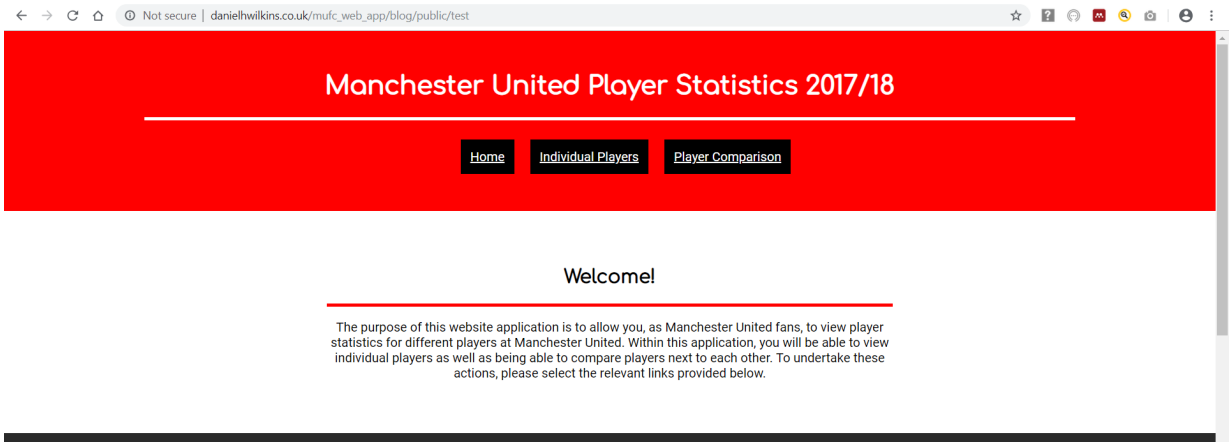
After completing the previous step, I then went to upload the files onto my server. However, I first needed to transfer these to my new laptop where I had 'FileZilla' installed, an FTP client. From viewing the tutorial online, I understood that I needed to place the 'public' folder's contents separate from the actual application, placing the files into the 'public_html' folder. However, I currently had files within this folder which meant that I kept the files within the same project folder rather than separating these as I didn't want to cause my website to not function as a result. I then also changed the '.env' file to match that of my server's established credentials so that the application could link to the correct database, allowing for using the imported data.

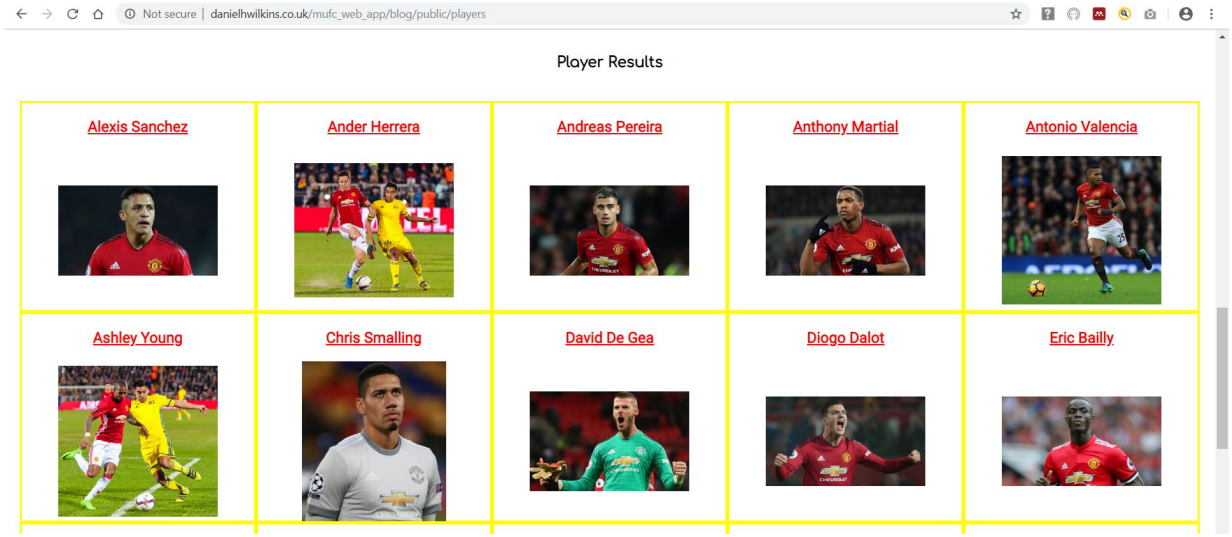
However, whilst attempting to test if the website application now appeared on my server, I encountered a HTTP 500 error with the different pages not displaying. As a result of this, I then attempted the method shown in the tutorial by placing the contents of the 'public' folder separate from the other files and changing some files paths within the 'index.php' to suit these changes. However, I also encountered an issue but this time, this was regarding the fact that the pages could not be found.

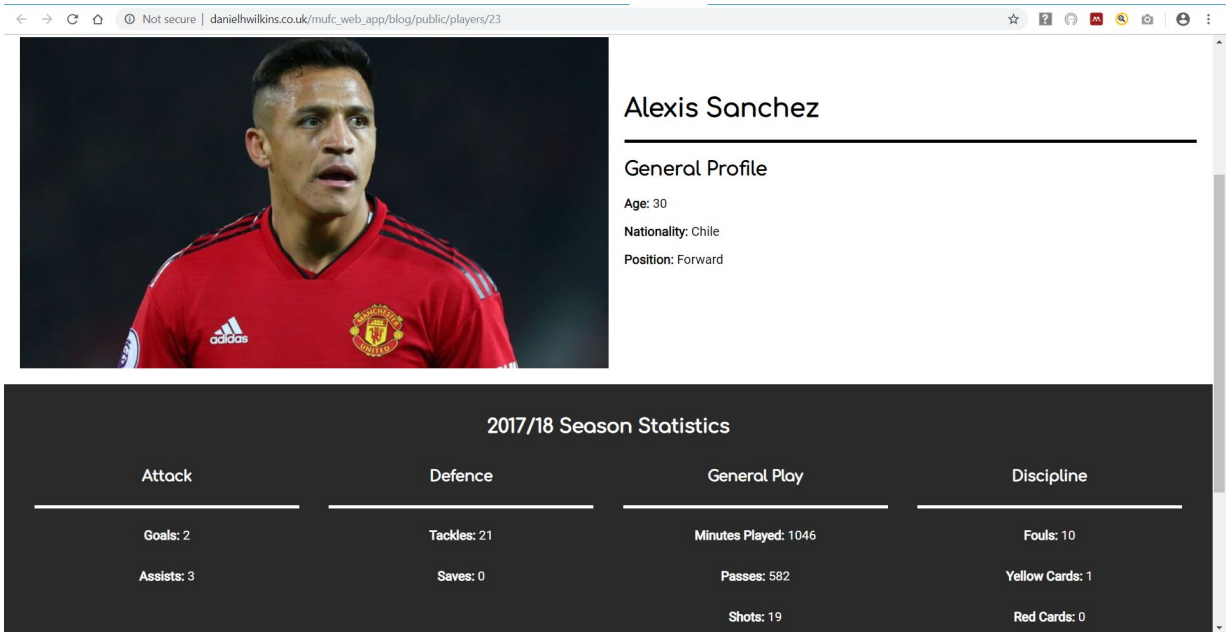
After the previous problems, I then continued experimenting through areas discovered through research. One area I attempted was changing the application 'URL' within the '.env' file but this didn't have an effect. I changed this as I thought this needed to relate to the 'URL' of my server. Despite attempting many different aspects, I was unsuccessful. Consequently, I returned the files to their original position within the 'public_html' folder on the server. This was because I knew that this was the aspect I was getting closer with regarding displaying the website application on the server. I also understood that this was an error relating to a potential 'PHP' problem which meant I examined the relevant files but couldn't find anything causing the issue. From research, I then inspected the error log, changing an aspect from an issue identified in this. However, again, this didn't work.

After experiencing severe difficulties with this aspect, I then discovered a solution online regarding the 'PHP' version being utilised by my server. From this, I then discovered that the version was one not compatible with 'Laravel'. Therefore, I changed this from version 7.0 to 7.2 and this resolved the issue, with the application now displaying as seen on the following page.

The Application now Displayed on the Server with Relevant Data (Examples)







After having managed to allow for the website application to display on the server, I then altered some files containing links to elements such as pages and images to relate to the 'URL' of my server. This was because some pages couldn't be navigated to when selecting links due to the fact that these related to the 'localhost' 'URL' on my old laptop.

The Final Code

As I had now managed to upload the project to my own server, this meant that some code was modified. Furthermore, the code had been modified to include comments where appropriate, organising and tidying the code as best as I could. The final code for each most relevant aspect can be seen below. Please note that some comments were rearranged due to disappearance off the page when transferred across to a different version of 'Sublime Text'. Please also note that, at a later stage, I integrated the references aspect into the footer section of each page. The actual final outcome can be viewed via my personal website.

Most Relevant Models

'Player.php' File

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Player extends Model
{

    public function category() {
        return $this->belongsTo(Category::class); // This was created to allow for the nationality filter set to function correctly
    }

    public function age() {
        return $this->belongsTo(Age::class); // This was created to allow for the age filter set to function correctly
    }

    public function position() {
        return $this->belongsTo(Position::class); // This was created to allow for the position filter set to function correctly
    }

    public function fullseasonsstatistic() {
        return $this->belongsTo(FullSeasonStatistic::class); // This was created to attempt to allow for the user to specify player statistics by full season compared to matchday
    }
}
```

'Category.php' File (Relating to Nationalities)

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    public function players() {
        return $this->hasMany(Player::class); // This model was created to allow for the nationality filter set to function correctly
    }
}
```

'Position.php' File

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Position extends Model
{
    public function players() {
        return $this->hasMany(Player::class); // This model was created to allow for the position filter set to function correctly
    }
}
```

'Age.php' File

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Age extends Model
{
    public function players() {
        return $this->hasMany(Player::class); // This model was created to allow for the age filter set to function correctly
    }
}
```

Most Relevant Migrations

'Players' Migration File

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration; // This file was created to update the database with a players table to contain information regarding different players

class CreatePlayersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('players', function (Blueprint $table) { // These aspects related to the fields to be placed into the relevant table in the database after updating the database
            $table->bigIncrements('id');
            $table->timestamps();
            $table->string('name');
            $table->integer('age');
            $table->string('nationality');
            $table->string('image');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('players');
    }
}
```

'Categories' Migration File (Relating to Nationalities)

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration; // This file was created to update the database with a categories table to contain information regarding
                                              // different nationalities to then be linked with the players on the application

class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) { // These aspects related to the fields to be placed into the relevant table in
                                                                   //the database after updating the database
            $table->bigIncrements('id');
            $table->string('title');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

'Ages' Migration File

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration; // This file was created to update the database with an ages
                                              //table to contain information regarding different ages to then be linked with the players on the application

class CreateAgesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('ages', function (Blueprint $table) { // These aspects related to the fields to be placed into the
                                                             //relevant table in the database after updating the database
            $table->bigIncrements('id');
            $table->string('title');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('ages');
    }
}
```

'Positions' Migration File

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration; // This file was created to update the database with a positions table to contain
                                              //information regarding different positions to then be linked with the players on the application

class CreatePositionsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('positions', function (Blueprint $table) { // These aspects related to the fields to be placed
                                                                  //into the relevant table in the database after updating the database
            $table->bigIncrements('id');
            $table->string('title');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('positions');
    }
}
```

Most Relevant Template File

'mufc_app_template1.blade.php' File

```
<!DOCTYPE html> <!-- This file was created to act as a template for the header and footer sections of each relevant page -->
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8"> <!-- Character Set -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="{{ asset('css/stylesheet.css') }}" rel="stylesheet"> <!-- Link to the Relevant Stylesheet -->
    <link href="https://fonts.googleapis.com/css?family=Comfortaa|Roboto" rel="stylesheet"> <!-- Google Fonts Integration -->
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css" integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/" crossorigin="
    anonymous"> <!-- Font Awesome Icons Integration -->
</head>
<body>
```

```
<!-- HEADER SECTION START -->

    <div class="header_container">

        <!-- Title Section Start -->

        <div class="header_title_container">
            <h1>Manchester United Player Statistics 2017/18</h1>
            <hr>
        </div>

        <!-- Title Section End -->

        <!-- Navigation Section Start -->

        <div class="header_navigation_container">
            <ul>
                <li><a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/test">Home</a></li>
                <li><a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/players">Individual Players</a></li>
                <li><a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/player_comparison">Player Comparison</a></li>
            </ul>
        </div>

        <!-- Navigation Section End -->

    </div>

<!-- HEADER SECTION END -->
```

```
<!-- INDIVIDUAL PAGE CONTENT SECTION START -->

    @yield('content')

<!-- INDIVIDUAL PAGE CONTENT SECTION END -->

<!-- FOOTER SECTION START -->

    <div class="footer_container">

        <!-- Footer Section 1 Start -->

        <div class="fc_sub_container1">

            <!-- Logo Section Start -->

            <div class="fc_logo_container">
                <div class="fc_logo_sub_container">
                    <a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/test"><img src="http://danielhwilkins.co.uk/mufc_web_app/blog/public/images/man_utd_logo.png" class="
                    footer_logo"></a>
                </div>
            </div>

            <!-- Logo Section End -->

            <!-- Sources Section Start -->

            <div class="fc_sources_container">
                <div class="fc_sources_sub_container">
                    <p>Produced by Daniel Wilkins with aid of sources viewable <a href="#" target="_blank">here</a></p>
                </div>
            </div>

            <!-- Sources Section End -->

        </div>

        <!-- Footer Section 1 End -->

        <!-- Footer hr Section Start -->

        <div class="fc_hr_container">
            <hr>
        </div>

        <!-- Footer hr Section End -->
```

```
        <!-- Footer Section 2 Start -->

        <div class="fc_sub_container2">
            <div class="fc_page_links_container">
                <div class="fc_page_links_sub_container">

                    <!-- Title Section Start -->

                    <div class="fc_plsc_heading_container">
                        <h3>Page Links</h3>
                        <hr>
                    </div>

                    <!-- Title Section End -->

                    <!-- Navigation Links Section Start -->

                    <div class="fc_plsc_links_container">
                        <ul>
                            <li><a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/test">Home</a></li>
                            <li><a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/players">Individual Players</a></li>
                            <li><a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/player_comparison">Player Comparison</a></li>
                        </ul>
                    </div>

                    <!-- Navigation Links Section End -->

                </div>
            </div>
        </div>

        <!-- Footer Section 2 End -->

    </div>

<!-- FOOTER SECTION END -->

</body>
</html>
```

Main Pages of the Website Application

'Home/Introduction' Page File

```
<!-- This file was created to act as the 'Home/Introduction' page -->

<!-- INTEGRATING THE TEMPLATE SECTION START -->

@extends('layouts.mufc_app_template1')

<!-- INTEGRATING THE TEMPLATE SECTION END -->

<!-- CHANGING THE TAB TITLE SECTION START -->

<title>MUFC Web App - Home</title>

<!-- CHANGING THE TAB TITLE SECTION END -->

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION START -->

@section('content')

<!-- APPLICATION WELCOME SECTION START -->

<div class="page_welcome_container">
    <div class="page_welcome_sub_container">
        <h2>Welcome!</h2>
        <hr>
        <p>The purpose of this website application is to allow you, as Manchester United fans, to view player statistics for different players at Manchester United. Within this
        application, you will be able to view individual players as well as being able to compare players next to each other. To undertake these actions, please select the relevant links
        provided below.</p>
    </div>
</div>

<!-- APPLICATION WELCOME SECTION END -->
```

```
<!-- 'INDIVIDUAL PLAYERS' PAGE INFORMATION SECTION START -->

<div class="individual_players_container">
    <div class="individual_players_sub_container">

        <!-- Title Section Start -->

        <div class="ipc_title_container">
            <h3>Individual Players</h3>
            <i class="far fa-futbol"></i>
        </div>

        <!-- Title Section End -->

        <!-- Description Section Start -->

        <div class="ipc_description_container">
            <p>By selecting the link below, you will be able to view a player's profile and statistics through selecting their name provided on the page. If you wish to search for or
            filter players, search options are provided at the top of this page, once navigated to, to help undertake this process.</p>
            <a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/players"><button>Page Link</button></a>
        </div>

        <!-- Description Section End -->

    </div>
</div>

<!-- 'INDIVIDUAL PLAYERS' PAGE INFORMATION SECTION END -->
```

```
<!-- 'PLAYER COMPARISON' PAGE INFORMATION SECTION START -->

<div class="player_comparison_container">
    <div class="player_comparison_sub_container">

        <!-- Description Section Start -->

        <div class="pcc_description_container">
            <p>By selecting the link below, you will be able to view two selected players next to each other to be able to compare statistics for the season. To view certain players, once
            having navigated to the page, you will be able to undertake a custom search or select search filters to achieve this.</p>
            <a href="http://danielhwilkins.co.uk/mufc_web_app/blog/public/player_comparison"><button>Page Link</button></a>
        </div>

        <!-- Description Section End -->

        <!-- Title Section Start -->

        <div class="pcc_title_container">
            <h3>Player Comparison</h3>
            <i class="far fa-futbol"></i>
        </div>

        <!-- Title Section End -->

    </div>
</div>

<!-- 'PLAYER COMPARISON' PAGE INFORMATION SECTION END -->

@endsection

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION END -->
```

'Individual Players' Page File

```
<!-- This file was created to act as the 'Individual Players' page -->

<!-- INTEGRATING THE TEMPLATE SECTION START -->

@extends('layouts.mufc_app_template1')

<!-- INTEGRATING THE TEMPLATE SECTION END -->

<!-- CHANGING THE TAB TITLE SECTION START -->

<title>MUFC Web App - Individual Players</title>

<!-- CHANGING THE TAB TITLE SECTION END -->

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION START -->

@section('content')

<!-- PAGE HEADING SECTION START -->

<div class="page_heading_container">
    <div class="page_heading_sub_container">
        <h2>Individual Players</h2>
        <hr>
    </div>
</div>

<!-- PAGE HEADING SECTION END -->

<!-- MAIN CONTENT SECTION START -->

<div class="player_statistics_container">

    <!-- Search/Filters Section Start -->

    <!-- Custom Search Section Start -->

    <div class="psc_filter_container">
        <div class="typed_search_container">
            <h3>Custom Search</h3>
            <form action="http://danielhwilkins.co.uk/mufc_web_app/blog/public/search" method="post">
                {{ csrf_field() }}
                <label for="query"></label>
                <input type="query" id="query" name="query">
                <input type="submit" class="submit_styles">
            </form>
        </div>
```

```
        <!-- Custom Search Section End -->

        <!-- Filters Section Start -->

        <div class="filters_search_container">
            <div class="fsc_title_container">
                <h3>Player Filters</h3>
            </div>
            <div class="fsc_main_container">
                <div class="age_filter_container">
                <h4>Age</h4>
                    <ul>
                    @foreach ($ages as $age)
                        <a href="{{ route('age', $age->id) }}"><li>{{ $age->title }}</li></a>
                    @endforeach
                    </ul>
                </div>

                <div class="nationality_filter_container">
                    <h4>Nationality</h4>
                    <ul>
                    @foreach ($categories as $category)
                        <a href="{{ route('category', $category->id) }}"><li>{{ $category->title }}</li></a>
                    @endforeach
                    </ul>
                </div>

                <div class="position_filter_container">
                    <h4>Position</h4>
                    <ul>
                    @foreach ($positions as $position)
                        <a href="{{ route('position', $position->id) }}"><li>{{ $position->title }}</li></a>
                    @endforeach
                    </ul>
                </div>
            </div>
        </div>

        <!-- Filters Section End -->

    </div>

    <!-- Search/Filters Section End -->
```

```
    <!-- Players Section Start -->

    <!-- Title Section Start -->

    <div class="psc_title_container">
        <h3>Player Results</h3>
    </div>

    <!-- Title Section End -->

    <!-- Player Results Section Start -->

    @if(count($players) > 0)
    @foreach($players->chunk(5) as $chunk)
            <div class="psc_pc_players">
            @foreach($chunk as $player)
                <div class="psc_pc_players2">
                    <h3><a href="/mufc_web_app/blog/public/players/{{$player->id}}" target="_blank">{{$player->name}}</a></h3>
                    <img src="{{$player->image}}" class="psc_pc_player_images">
                </div>
            @endforeach
            </div>
    @endforeach
    @else
        <p>No players found</p>
    @endif

    <!-- Player Results Section End -->

    <!-- Players Section End -->

</div>

<!-- MAIN CONTENT SECTION END -->

@endsection

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION END -->
```

The 'show' Page File (Actual Individual Players with their Statistics)

```
<!-- This file was created to act as the page to display a selected player from the 'Individual Players' page -->

<!-- INTEGRATING THE TEMPLATE SECTION START -->

@extends('layouts.mufc_app_template1')

<!-- INTEGRATING THE TEMPLATE SECTION END -->

<!-- CHANGING THE TAB TITLE WITH SELECTED PLAYER NAME SECTION START -->

<title>MUFC Web App - Individual Players ({{$player->name}})</title>

<!-- CHANGING THE TAB TITLE WITH SELECTED PLAYER NAME SECTION END -->

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION START -->

@section('content')

<!-- PLAYER OVERVIEW/GENERAL INFORMATION SECTION START -->

<div class="player_overview_container">
    <div class="poc_general_information_container">
        <div class="poc_gic_image_container">
            <a href="/mufc_web_app/blog/public/players"><button><i class="far fa-arrow-alt-circle-left"></i> Go Back</button></a>
            <img src="{{$player->image}}" class="poc_gic_images">
        </div>
        <div class="poc_gic_information_container">
            <h1>{{$player->name}}</h1>
            <hr>
            <h2>General Profile</h2>
            <p><b>Age:</b> {{$player->age}}</p>
            <p><b>Nationality:</b> {{$player->nationality}}</p>
            <p><b>Position:</b> {{$player->position}}</p>
        </div>
    </div>
</div>

<!-- PLAYER OVERVIEW/GENERAL INFORMATION SECTION END -->
```

```
<!-- PLAYER STATISTICS SECTION START -->

<div class="player_statistics_container2">

    <!-- Title Section Start -->

    <div class="psc2_title_container">
        <h2>2017/18 Season Statistics</h2>
    </div>

    <!-- Title Section End -->

    <!-- Statistics by Categories Section Start -->

    <div class="psc2_categories_container">
        <div class="psc2_cc_attack">
            <h3>Attack</h3>
            <hr>
            <p><b>Goals:</b> {{$player->goals}}</p>
            <p><b>Assists:</b> {{$player->assists}}</p>
        </div>
        <div class="psc2_cc_defence">
            <h3>Defence</h3>
            <hr>
            <p><b>Tackles:</b> {{$player->tackles}}</p>
            <p><b>Saves:</b> {{$player->saves}}</p>
        </div>
        <div class="psc2_cc_general">
            <h3>General Play</h3>
            <hr>
            <p><b>Minutes Played:</b> {{$player->minutes_played}}</p>
            <p><b>Passes:</b> {{$player->passes}}</p>
            <p><b>Shots:</b> {{$player->shots}}</p>
            <p><b>Offsides:</b> {{$player->offsides}}</p>
        </div>
        <div class="psc2_cc_discipline">
            <h3>Discipline</h3>
            <hr>
            <p><b>Fouls:</b> {{$player->fouls}}</p>
            <p><b>Yellow Cards:</b> {{$player->yellow_cards}}</p>
            <p><b>Red Cards:</b> {{$player->red_cards}}</p>
        </div>
    </div>

    <!-- Statistics by Categories Section End -->

</div>

<!-- PLAYER STATISTICS SECTION END -->

@endsection

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION END -->
```

'Player Comparison' Page File

```
<!-- This file was created to act as the 'Player Comparison' page -->

<!-- INTEGRATING THE TEMPLATE SECTION START -->

@extends('layouts.mufc_app_template1')

<!-- INTEGRATING THE TEMPLATE SECTION END -->

<!-- CHANGING THE TAB TITLE SECTION START -->

<title>MUFC Web App - Player Comparison</title>

<!-- CHANGING THE TAB TITLE SECTION END -->

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION START -->

@section('content')

<!-- PAGE HEADING SECTION START -->

<div class="page_heading_container">
    <div class="page_heading_sub_container">
        <h2>Player Comparison</h2>
        <hr>
    </div>
</div>

<!-- PAGE HEADING SECTION END -->
```

```
<!-- MAIN CONTENT SECTION START -->

<div class="player_comparison_container">

    <!-- Player Comparison Left Section Start -->

    <div class="pcc_section1">

            <!-- Search/Filters Section Start -->

            <div class="pccs1_filter_container">

                <!-- Custom Search Section Start -->

                <div class="pccs1_typed_search_container">
                    <h3>Custom Search</h3>
                    <form action="http://danielhwilkins.co.uk/mufc_web_app/blog/public/player_comparison" method="post">
                        {{ csrf_field() }}
                        <label for="query"></label>
                        <input type="query" id="query" name="query">
                        <input type="submit" class="pccs1_submit_styles">
                    </form>
                </div>

                <!-- Custom Search Section End -->

                <!-- Filters Section Start -->

                <div class="pccs1_filters_search_container">
                    <div class="pccs1_fsc_title_container">
                        <h3>Player Filters</h3>
                    </div>
                    <div class="pccs1_fsc_main_container">
                        <div class="pccs1_age_filter_container">
                        <h4>Age</h4>
                            <ul>
                            @foreach ($ages2 as $age2)
                                <a href="{{ route('age2', $age2->id) }}"><li>{{ $age2->title }}</li></a>
                            @endforeach
                            </ul>
                        </div>
```

```
                <div class="pccs1_nationality_filter_container">
                    <h4>Nationality</h4>
                    <ul>
                    @foreach ($categories2 as $category2)
                        <a href="{{ route('category2', $category2->id) }}"><li>{{ $category2->title }}</li></a>
                    @endforeach
                    </ul>
                </div>

                <div class="pccs1_position_filter_container">
                    <h4>Position</h4>
                    <ul>
                    @foreach ($positions2 as $position2)
                        <a href="{{ route('position2', $position2->id) }}"><li>{{ $position2->title }}</li></a>
                    @endforeach
                    </ul>
                </div>
            </div>
        </div>

        <!-- Filters Section End -->

    </div>

    <!-- Search/Filters Section End -->

    <!-- Players Section Start -->

    <!-- Title Section Start -->

    <div class="pcc_mrcs1_title_container">
        <h3>Player Results</h3>
    </div>

    <!-- Title Section End -->
```

```
    <!-- Player Results Section Start -->

    @if(count($players) > 0)
    @foreach($players->chunk(3) as $chunk)
            <div class="pcc_mrcs1_players">
            @foreach($chunk as $player)
                <div class="pcc_mrcs1_players2">
                    <h3>{{$player->name}}</a></h3>
                    <img src="{{$player->image}}" class="pcc_mrcs1_player_images">
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Minutes Played:</b> {{$player->minutes_played}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Goals:</b> {{$player->goals}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Assists:</b> {{$player->assists}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Tackles:</b> {{$player->tackles}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Saves:</b> {{$player->saves}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Passes:</b> {{$player->passes}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Shots:</b> {{$player->shots}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Offsides:</b> {{$player->offsides}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Fouls:</b> {{$player->fouls}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Yellow Cards:</b> {{$player->yellow_cards}}</p>
                    </div>
                    <div class="pcc_mrsc1_statistic_containers">
                        <p><b>Red Cards:</b> {{$player->red_cards}}</p>
                    </div>
                </div>
            @endforeach
            </div>
    @endforeach
    @else
        <p>No players found</p>
    @endif

    <!-- Player Results Section End -->
```

```
        <!-- Players Section End -->

    </div>

    <!-- Player Comparison Left Section End -->

    <!-- hr Section Start -->

    <div class="pcc_hr_section">
        <hr>
    </div>

    <!-- hr Section End -->

<!-- Player Comparison Right Section Start -->

<div class="pcc_section2">

        <!-- Search/Filters Section Start -->

        <div class="pccs2_filter_container">

            <!-- Custom Search Section Start -->

            <div class="pccs2_typed_search_container">
                <h3>Custom Search</h3>
                <form action="http://danielhwilkins.co.uk/mufc_web_app/blog/public/player_comparison" method="post">
                    {{ csrf_field() }}
                    <label for="query"></label>
                    <input type="query" id="query" name="query">
                    <input type="submit" class="pccs2_submit_styles">
                </form>
            </div>

            <!-- Custom Search Section End -->

            <!-- Filters Section Start -->
```

```
            <div class="pccs2_filters_search_container">
                <div class="pccs2_fsc_title_container">
                    <h3>Player Filters</h3>
                </div>
                <div class="pccs2_fsc_main_container">
                    <div class="pccs2_age_filter_container">
                    <h4>Age</h4>
                        <ul>
                        @foreach ($ages2 as $age2)
                            <a href="{{ route('age2', $age2->id) }}"><li>{{ $age2->title }}</li></a>
                        @endforeach
                        </ul>
                    </div>

                    <div class="pccs2_nationality_filter_container">
                        <h4>Nationality</h4>
                        <ul>
                        @foreach ($categories2 as $category2)
                            <a href="{{ route('category2', $category2->id) }}"><li>{{ $category2->title }}</li></a>
                        @endforeach
                        </ul>
                    </div>

                    <div class="pccs2_position_filter_container">
                        <h4>Position</h4>
                        <ul>
                        @foreach ($positions2 as $position2)
                            <a href="{{ route('position2', $position2->id) }}"><li>{{ $position2->title }}</li></a>
                        @endforeach
                        </ul>
                    </div>
                </div>
            </div>

            <!-- Filters Section End -->

        </div>

        <!-- Search/Filters Section End -->

        <!-- Players Section Start -->

        <!-- Title Section Start -->

        <div class="pcc_mrcs2_title_container">
            <h3>Player Results</h3>
        </div>

        <!-- Title Section End -->
```

```
        <!-- Player Results Section Start -->

        @if(count($players) > 0)
        @foreach($players->chunk(3) as $chunk)
                <div class="pcc_mrcs2_players">
                @foreach($chunk as $player)
                    <div class="pcc_mrcs2_players2">
                        <h3>{{$player->name}}</a></h3>
                        <img src="{{$player->image}}" class="pcc_mrcs2_player_images">
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Minutes Played:</b> {{$player->minutes_played}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Goals:</b> {{$player->goals}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Assists:</b> {{$player->assists}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Tackles:</b> {{$player->tackles}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Saves:</b> {{$player->saves}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Passes:</b> {{$player->passes}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Shots:</b> {{$player->shots}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Offsides:</b> {{$player->offsides}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Fouls:</b> {{$player->fouls}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Yellow Cards:</b> {{$player->yellow_cards}}</p>
                        </div>
                        <div class="pcc_mrsc2_statistic_containers">
                            <p><b>Red Cards:</b> {{$player->red_cards}}</p>
                        </div>
                    </div>
                @endforeach
                </div>
        @endforeach
        @else
            <p>No players found</p>
        @endif

        <!-- Player Results Section End -->
```

```
        <!-- Players Section End -->

    </div>

    <!-- Player Comparison Right Section End -->

</div>

<!-- MAIN CONTENT SECTION END -->

@endsection

<!-- IMPLEMENTING THE INDIVIDUAL PAGE CONTENT SECTION END -->
```

Most Relevant Controllers

'AgesController.php' File (Most Relevant Parts of the File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position; // These aspects allow for using elements from elsewhere in the application

class AgesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index() // This will return the 'Individual Players' page with all filter sets
    {

        $players = Player::orderBy('name', 'asc')->get();
        $categories = Category::with('players')->orderBy('title', 'asc')->get();
        $ages = Age::with('players')->orderBy('title', 'asc')->get();
        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        return view('players.index', compact('players', 'categories', 'ages'));

    }

    public function age($id) // This allows the user to specify players by 'age_id' (Age Group) whilst also returning the 'Individual Players' page with all filter sets
    {

        $ages = Age::with('players')->orderBy('title', 'asc')->get();

        $categories = Category::with('players')->orderBy('title', 'asc')->get();

        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        $players = Player::orderBy('name')->get()->where('age_id', $id);

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }
```

```php
    public function age2($id) // This allows the user to specify players by 'age_id' (Age Group) whilst also returning the 'Player Comparison' page with all filter sets
    {

        $ages2 = Age::with('players')->orderBy('title', 'asc')->get();

        $categories2 = Category::with('players')->orderBy('title', 'asc')->get();

        $positions2 = Position::with('players')->orderBy('title', 'asc')->get();

        $players = Player::orderBy('name')->get()->where('age_id', $id);

        return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));

    }
```

'CategoriesController.php' File (Most Relevant Parts of the File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;
use App\FullSeasonStatistic; // These aspects allow for using elements from elsewhere in the application

class CategoriesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index() // This will return the 'Individual Players' page with all filter sets
    {

        $players = Player::orderBy('name', 'asc')->get();
        $categories = Category::with('players')->orderBy('title', 'asc')->get();
        $ages = Age::with('players')->orderBy('title', 'asc')->get();
        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }

    public function category($id) // This allows the user to specify players by 'category_id' (Nationality Group) whilst also returning the 'Individual Players' page with all filter sets
    {

        $categories = Category::with('players')->orderBy('title', 'asc')->get();

        $ages = Age::with('players')->orderBy('title', 'asc')->get();

        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        $players = Player::orderBy('name')->get()->where('category_id', $id);

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }
```

```php
public function category2($id) // This allows the user to specify players by 'category_id' (Nationality Group) whilst also returning the 'Player Comparison' page with all filter sets
{

    $categories2 = Category::with('players')->orderBy('title', 'asc')->get();

    $ages2 = Age::with('players')->orderBy('title', 'asc')->get();

    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();

    $players = Player::orderBy('name')->get()->where('category_id', $id);

    return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));

}
```

'HomeController2.php' File (Most Relevant Parts of the File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController2 extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index() // This will return the 'Home/Introduction' page for the application
    {
        return view('players.test');
    }
```

'PagesController.php' File (Most Relevant Parts of the File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    public function index(){
        return view('pages.index'); // This will return the 'Individual Players' page for the application
    }
}
```

'PlayerComparisonsController.php' File (Most Relevant Parts of the File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;
use App\FullSeasonStatistic; // These aspects allow for using elements from elsewhere in the application

class PlayerComparisonsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index() // This will return the 'Player Comparison' page for the application with all filter sets
    {
        $players = Player::orderBy('name', 'asc')->get();
        $categories2 = Category::with('players')->orderBy('title', 'asc')->get();
        $ages2 = Age::with('players')->orderBy('title', 'asc')->get();
        $positions2 = Position::with('players')->orderBy('title', 'asc')->get();

        return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));
    }
```

'PlayersController.php' File (Most Relevant Parts of the File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;
use App\FullSeasonStatistic; // These aspects allow for using elements from elsewhere in the application

class PlayersController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index() // This will return the 'Individual Players' page for the application with all filter sets
    {

        $players = Player::orderBy('name', 'asc')->get();
        $categories = Category::with('players')->orderBy('title', 'asc')->get();
        $ages = Age::with('players')->orderBy('title', 'asc')->get();
        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }
```

```php
public function show($id) // This will return the selected player in a separate page
{
    $player = Player::find($id);
    return view('players.show')->with('player', $player);
}
```

'PositionsController.php' File (Most Relevant Parts of the File)

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Player;
use App\Age;
use App\Category;
use App\Position;
use App\FullSeasonStatistic; // These aspects allow for using elements from elsewhere in the application

class PositionsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index() // This will return the 'Individual Players' page with all filter sets
    {

        $players = Player::orderBy('name', 'asc')->get();
        $categories = Category::with('players')->orderBy('title', 'asc')->get();
        $ages = Age::with('players')->orderBy('title', 'asc')->get();
        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }

    public function position($id) // This allows the user to specify players by 'position_id' (Position Group) whilst also returning the 'Individual Players' page with all filter sets
    {

        $ages = Age::with('players')->orderBy('title', 'asc')->get();

        $categories = Category::with('players')->orderBy('title', 'asc')->get();

        $positions = Position::with('players')->orderBy('title', 'asc')->get();

        $players = Player::orderBy('name')->get()->where('position_id', $id);

        return view('players.index', compact('players', 'categories', 'ages', 'positions'));

    }
```

```php
public function position2($id) // This allows the user to specify players by 'position_id' (Position Group) whilst also returning the 'Player Comparison' page with all filter sets
{

    $ages2 = Age::with('players')->orderBy('title', 'asc')->get();

    $categories2 = Category::with('players')->orderBy('title', 'asc')->get();

    $positions2 = Position::with('players')->orderBy('title', 'asc')->get();

    $players = Player::orderBy('name')->get()->where('position_id', $id);

    return view('players.player_comparison', compact('players', 'categories2', 'ages2', 'positions2'));

}
```

Most Relevant Routes

'web.php' File

```php
<?php

use Illuminate\Support\Facades\Input;
use App\Player;
use App\Category;
use App\Age;
use App\Position; // These aspects allow for using elements from elsewhere in the application
/*
```

```php
/* ROUTES ALLOWING FOR SHOWING PAGES SECTION START */

/* 'Individual Players' Page Route Start */

Route::get('/', 'PagesController@index');

/* 'Individual Players' Page Route End */

//Route::resource('posts', 'PostsController'); This was established for experimenting during the initial stages of the project

Route::resource('players', 'PlayersController');

Route::get('/', function(){
    return view('welcome');
});

/* 'Home/Introduction' Page Route Start */

Route::get('/test', 'HomeController2@index');

/* 'Home/Introduction' Page Route End */

/* 'Player Comparison' Page Route Start */

Route::get('/player_comparison', 'PlayerComparisonsController@index');

/* 'Player Comparison' Page Route End */

/* ROUTES ALLOWING FOR SHOWING PAGES SECTION END */


/* ------------------------------------------------------------------------------------ */
```

```php
/* ROUTES FOR PLAYER FILTER SETS SECTION START */

/* 'Player Comparison' Page Search and Filters Start */

/* Custom Search Start */

Route::any('/player_comparison', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($players) > 0){
        return view('players.player_comparison')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories2', Category::orderBy('title', 'asc')->get())
        ->with('ages2', Age::orderBy('title', 'asc')->get())
        ->with('positions2', Position::orderBy('title', 'asc')->get());
    }
    else return view ('players.player_comparison')
        ->withMessage('No Details found. Try to search again !')
        ->with('players', $players)
        ->with('categories2', Category::orderBy('title', 'asc')->get())
        ->with('ages2', Age::orderBy('title', 'asc')->get())
        ->with('positions2', Position::orderBy('title', 'asc')->get());
});

/* Custom Search End */

/* Filters Start */

Route::any('/ages2/{age2}', [
    'uses' => 'AgesController@age2',
    'as' => 'age2'

]);

Route::any('/categories2/{category2}', [
    'uses' => 'CategoriesController@category2',
    'as' => 'category2'
]);

Route::any('/positions2/{position2}', [
    'uses' => 'PositionsController@position2',
    'as' => 'position2'
]);

/* Filters End */

/* 'Player Comparison' Page Search and Filters End */
```

```
/* 'Individual Players' Page Search and Filters Start */

/* Custom Search Start */

Route::any('/search', function(Input $input){
$q = Input::get ( 'query' );
    $players = Player::where('name','LIKE','%'.$q.'%')->get();
    if(count($players) > 0){
        return view('players.index')
        ->with('players', $players)->withQuery ( $q )
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get())
        ->with('positions', Position::orderBy('title', 'asc')->get());
    }
    else return view ('players.index')
        ->withMessage('No Details found. Try to search again !')
        ->with('players', $players)
        ->with('categories', Category::orderBy('title', 'asc')->get())
        ->with('ages', Age::orderBy('title', 'asc')->get())
        ->with('positions', Position::orderBy('title', 'asc')->get());
});

/* Custom Search End */

/* Filters Start */

Route::any('/ages/{age}', [
    'uses' => 'AgesController@age',
    'as' => 'age'

]);

Route::any('/categories/{category}', [
    'uses' => 'CategoriesController@category',
    'as' => 'category'
]);

Route::any('/positions/{position}', [
    'uses' => 'PositionsController@position',
    'as' => 'position'
]);

Route::any('/fullseasonstatistics/{fullseasonstatistic}', [
    'uses' => 'FullSeasonStatisticController@fullseasonstatistic',
    'as' => 'fullseasonstatistic'
]);

/* Filters End */

/* 'Individual Players' Page Search and Filters End */

/* ROUTES FOR PLAYER FILTER SETS SECTION END */
```

The Website Application Styling File

'stylesheet.css' File – Desktop Code

```
/* This is the stylesheet for the Manchester United Website Application */

/* GENERAL STYLES START */

    /* For the Page Start */

    *, *:before, *:after {
        box-sizing: border-box;
    }

    body {
        padding: 0;
        margin: 0;
    }

    /* For the Page End */

    /* For the Text/Headings Start */

    h1 {
        font-family: 'Comfortaa', cursive;
    }

    h2 {
        font-family: 'Comfortaa', cursive;
    }

    h3 {
        font-family: 'Comfortaa', cursive;
    }

    h4 {
        font-family: 'Comfortaa', cursive;
        font-style: italic;
    }

    p {
        font-family: 'Roboto', sans-serif;
    }

    button {
        font-family: 'Roboto', sans-serif;
    }

    a {
        font-family: 'Roboto', sans-serif;
    }

    /* For the Text/Headings End */

/* GENERAL STYLES END */
```

```
/* ------------------------------------------------

/* TEMPLATE STYLES START */

    /* Header Section Start */

    .header_container {
        display: flex;
        flex-direction: column;
        width: 100%;
        height: auto;
        background-color: #FF0000;
        padding: 30px;
    }

    .header_title_container {
        width: 100%;
        height: auto;
        text-align: center;
        color: #FFFFFF;
    }

    .header_title_container hr {
        width: 80%;
        height: auto;
        border: 2px solid #FFFFFF;
    }

    .header_navigation_container {
        display: flex;
        justify-content: center;
        width: 100%;
        height: auto;
    }

    .header_navigation_container ul {
        display: flex;
        flex-direction: row;
        list-style-type: none;
    }

    .header_navigation_container li {
        border: 2px solid #000000;
        background-color: #000000;
        padding: 10px;
        margin-left: 10px;
        margin-right: 10px;
        text-align: center;
    }
```

```
    .header_navigation_container a {
        color: #FFFFFF;
    }

    /* Header Section End */

    /* Footer Section Start */

    .footer_container {
        display: flex;
        flex-direction: row;
        width: 100%;
        height: auto;
        background-color: #000000;
        padding: 10px;
        color: #FFFFFF;
    }

    .fc_sub_container1 {
        display: flex;
        flex-direction: row;
        align-items: center;
        width: 55%;
        height: auto;
    }

    .fc_logo_container {
        width: 20%;
        height: auto;
    }

    .fc_logo_sub_container {
        width: 100%;
        height: auto;
    }

    .footer_logo {
        width: 100%;
        height: auto;
    }

    .fc_sources_container {
        width: 80%;
        height: auto;
        padding-left: 20px;
    }

    .fc_sources_sub_container {
        width: 100%;
        height: auto;
    }
```

```
    .fc_sub_container2 {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        width: 35%;
        height: auto;
        padding-right: 50px;
    }

    .fc_page_links_container {
        width: 100%;
        height: auto;
    }

    .fc_page_links_sub_container {
        width: 100%;
        height: auto;
    }

    .fc_plsc_heading_container {
        display: flex;
        flex-direction: column;
        align-items: center;
        width: 100%;
        height: auto;
    }

    .fc_plsc_heading_container hr {
        width: 100%;
        height: auto;
        border: 2px solid #FF0000;
    }

    .fc_plsc_links_container {
        width: 100%;
        height: auto;
    }

    .fc_plsc_links_container ul {
        display: flex;
        flex-direction: column;
        align-items: center;
        list-style-type: none;
        padding: 0;
    }
```

```
    .fc_plsc_links_container a {
        color: #FFFFFF;
    }

    .fc_hr_container {
        width: 10%;
        height: auto;
    }

    .fc_hr_container hr {
        width: 1%;
        height: 26%;
        transform: rotate(180deg);
        border: 2px solid #FFFFFF;
    }

    /* Footer Section End */

/* TEMPLATE STYLES END */

/* ------------------------------------------------------------

/* 'HOME/INTRODUCTION' PAGE START */

    /* Application Welcome Section Start */

    .page_welcome_container {
        display: flex;
        flex-direction: column;
        width: 100%;
        height: auto;
        padding: 50px;
    }

    .page_welcome_sub_container {
        width: 50%;
        height: auto;
        text-align: center;
        margin: auto;
    }

    .page_welcome_container hr {
        width: 100%;
        height: auto;
        border: 2px solid #FF0000;
    }

    /* Application Welcome Section End */

    /* 'Individual Players' Page Information Section Start */
```

```css
.individual_players_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 50px;
    background-color: #2A2A2A;
    color: #FFFFFF;
}

.individual_players_sub_container {
    display: flex;
    flex-direction: row;
    width: 80%;
    height: auto;
    margin: auto;
    align-items: center;
}

.ipc_title_container {
    width: 50%;
    height: auto;
    text-align: center;
}

.ipc_title_container i {
    font-size: 250px;
}

.ipc_description_container {
    width: 50%;
    height: auto;
}

.ipc_description_container button {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    padding: 10px;
    cursor: pointer;
    transition: 0.5s;
    font-style: italic;
    font-weight: bold;
}

.ipc_description_container button:hover {
    background-color: #FF0000;
    border: 2px solid #000000;
    transition: 0.5s;
}
```

```css
/* 'Individual Players' Page Information Section End */

/* 'Player Comparison' Page Information Section Start */

.player_comparison_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 50px;
}

.player_comparison_sub_container {
    display: flex;
    flex-direction: row;
    width: 80%;
    height: auto;
    margin: auto;
    align-items: center;
}

.pcc_title_container {
    width: 50%;
    height: auto;
    text-align: center;
}

.pcc_title_container i {
    font-size: 250px;
}

.pcc_description_container {
    width: 50%;
    height: auto;
}

.pcc_description_container button {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    padding: 10px;
    cursor: pointer;
    transition: 0.5s;
    font-style: italic;
    font-weight: bold;
}
```

```css
.pcc_description_container button:hover {
    background-color: #FF0000;
    border: 2px solid #000000;
    transition: 0.5s;
}

/* 'Player Comparison' Page Information Section End */

/* 'HOME/INTRODUCTION' PAGE END */

/* ---------------------------------------------------------------- */

/* 'INDIVIDUAL PLAYERS' PAGE START */

/* Page Heading Section Start */

.page_heading_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    padding: 50px;
}

.page_heading_sub_container {
    width: 50%;
    height: auto;
    text-align: center;
    margin: auto;
}

.page_heading_container hr {
    width: 100%;
    height: auto;
    border: 2px solid #FF0000;
}

/* Page Heading Section End */

/* Main Content Section Start */
```

```css
.player_statistics_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    padding: 20px;
    background: white;
    color: black;
}

/* Main Content Section - Search/Filters Section Start */

.psc_filter_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
}

.psc_filter_container a {
    color: #FF0000;
}

.typed_search_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.typed_search_container input {
    font-family: 'Roboto', sans-serif;
}

.submit_styles {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    cursor: pointer;
}

.filters_search_container {
    display: flex;
    flex-direction: column;
    width: 100%;
    padding: 10px;
}
```

```css
.fsc_title_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.fsc_main_container {
    width: 100%;
    height: auto;
    display: flex;
    flex-direction: row;
}

.filters_search_container ul {
    list-style-type: none;
    padding: 0;
}

.filters_search_container li {
    padding: 10px;
    border: 2px solid #000000;
    margin-top: 10px;
}

.age_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

.nationality_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

.position_filter_container {
    width: 33.33%;
    height: auto;
    text-align: center;
    padding: 10px;
}

/* Main Content Section - Search/Filters Section End */

/* Main Content Section - Players Section Start */
```

```css
.psc_title_container {
    width: 100%;
    height: auto;
    text-align: center;
    margin-bottom: 20px;
}

.psc_pc_players {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
}

.psc_pc_players2 {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: auto;
    border: 2px solid yellow;
    text-align: center;
}

.psc_pc_players2 a {
    color: #FF0000;
    transition: 0.5s;
}

.psc_pc_players2 a:hover {
    color: #000000;
    transition: 0.5s;
}

.psc_pc_player_images {
    width: 200px;
    height: 200px;
    object-fit: contain;
    margin: auto;
}

/* Main Content Section - Players Section End */

/* Main Content Section End */

/* 'INDIVIDUAL PLAYERS' PAGE END */

/* ---------------------------------------------------------- */

/* 'SHOW' PAGE (ACTUAL INDIVIDUAL PLAYERS) START */
```

```css
/* Player Overview Section Start */

.player_overview_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    padding: 20px;
}

.poc_general_information_container {
    display: flex;
    flex-direction: row;
    align-items: center;
    width: 100%;
    height: auto;
}

.poc_gic_image_container {
    width: 50%;
    height: auto;
}

.poc_gic_image_container button {
    background-color: #000000;
    color: #FFFFFF;
    border: 2px solid #FF0000;
    padding: 10px;
    cursor: pointer;
    transition: 0.5s;
    font-style: italic;
    font-weight: bold;
}

.poc_gic_image_container button:hover {
    background-color: #FF0000;
    border: 2px solid #000000;
    transition: 0.5s;
}

.poc_gic_images {
    width: 100%;
    height: auto;
    margin-top: 20px;
}

.poc_gic_information_container {
    width: 50%;
    height: auto;
    padding-left: 20px;
}
```

```css
.poc_gic_information_container hr {
    width: 100%;
    height: auto;
    border: 2px solid #000000;
}

/* Player Overview Section End */

/* Player Statistics Section Start */

.player_statistics_container2 {
    display: flex;
    flex-direction: column;
    padding: 20px;
    width: 100%;
    height: auto;
    background-color: #2A2A2A;
    color: #FFFFFF;
}

.psc2_title_container {
    width: 100%;
    height: auto;
    text-align: center;
}

.psc2_categories_container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: auto;
    text-align: center;
}

.psc2_categories_container hr {
    width: 90%;
    height: auto;
    border: 2px solid #FFFFFF;
}

.psc2_cc_attack {
    display: flex;
    flex-direction: column;
    width: 25%;
    height: auto;
}
```

```
    .psc2_cc_defence {
        display: flex;
        flex-direction: column;
        width: 25%;
        height: auto;
    }

    .psc2_cc_general {
        display: flex;
        flex-direction: column;
        width: 25%;
        height: auto;
    }

    .psc2_cc_discipline {
        display: flex;
        flex-direction: column;
        width: 25%;
        height: auto;
    }

    /* Player Statistics Section End */

/* 'SHOW' PAGE (ACTUAL INDIVIDUAL PLAYERS) END */

/* ------------------------------------------------------

/* 'PLAYER COMPARISON' PAGE START */

    /* Main Content Section Start */

    .player_comparison_container {
        display: flex;
        flex-direction: row;
        width: 100%;
        height: auto;
        padding: 20px;
        background: white;
        color: black;
    }
```

```
    /* Player Comparison Section 1 Start */

    .pcc_section1 {
        display: flex;
        flex-direction: column;
        width: 45%;
        height: auto;
    }

        /* Player Comparison Section 1 - Search/Filters Section Start */

        .pccs1_filter_container {
            display: flex;
            flex-direction: column;
            width: 100%;
            height: auto;
        }

        .pccs1_filter_container a {
            color: #FF0000;
        }

        .pccs1_typed_search_container {
            width: 100%;
            height: auto;
            text-align: center;
        }

        .pccs1_typed_search_container input {
            font-family: 'Roboto', sans-serif;
        }

        .pccs1_submit_styles {
            background-color: #000000;
            color: #FFFFFF;
            border: 2px solid #FF0000;
            cursor: pointer;
        }

        .pccs1_filters_search_container {
            display: flex;
            flex-direction: column;
            width: 100%;
            padding: 10px;
        }
```

```
        .pccs1_fsc_title_container {
            width: 100%;
            height: auto;
            text-align: center;
        }

        .pccs1_fsc_main_container {
            width: 100%;
            height: auto;
            display: flex;
            flex-direction: row;
        }

        .pccs1_filters_search_container ul {
            list-style-type: none;
            padding: 0;
        }

        .pccs1_filters_search_container li {
            padding: 10px;
            border: 2px solid #000000;
            margin-top: 10px;
        }

        .pccs1_age_filter_container {
            width: 33.33%;
            height: auto;
            text-align: center;
            padding: 10px;
        }

        .pccs1_nationality_filter_container {
            width: 33.33%;
            height: auto;
            text-align: center;
            padding: 10px;
        }

        .pccs1_position_filter_container {
            width: 33.33%;
            height: auto;
            text-align: center;
            padding: 10px;
        }

        /* Player Comparison Section 1 - Search/Filters Section End */
```

```
        /* Player Comparison Section 1 - Main Results Section Start */

        .pcc_mrcs1_title_container {
            width: 100%;
            height: auto;
            text-align: center;
            margin-bottom: 20px;
        }

        .pcc_mrcs1_players {
            display: flex;
            flex-direction: row;
            width: 100%;
            height: auto;
        }

        .pcc_mrcs1_players2 {
            display: flex;
            flex-direction: column;
            width: 100%;
            height: auto;
            border: 2px solid yellow;
            text-align: center;
        }

        .pcc_mrcs1_players2 a {
            color: #FF0000;
            transition: 0.5s;
        }

        .pcc_mrcs1_players2 a:hover {
            color: #000000;
            transition: 0.5s;
        }

        .pcc_mrcs1_player_images {
            width: 200px;
            height: 200px;
            object-fit: contain;
            margin: auto;
        }

        .pcc_mrsc1_statistic_containers {
            width: 100%;
            height: auto;
            border: 2px solid #000000;
            margin-top: 10px;
        }

        /* Player Comparison Section 1 - Main Results Section End */
```

```
/* Player Comparison Section 1 End */

/* Player Comparison Section 2 Start */

.pcc_section2 {
    display: flex;
    flex-direction: column;
    width: 45%;
    height: auto;
}

    /* Player Comparison Section 2 - Search/Filters Section Start */

    .pccs2_filter_container {
        display: flex;
        flex-direction: column;
        width: 100%;
        height: auto;
    }

    .pccs2_filter_container a {
        color: #FF0000;
    }

    .pccs2_typed_search_container {
        width: 100%;
        height: auto;
        text-align: center;
    }

    .pccs2_typed_search_container input {
        font-family: 'Roboto', sans-serif;
    }

    .pccs2_submit_styles {
        background-color: #000000;
        color: #FFFFFF;
        border: 2px solid #FF0000;
        cursor: pointer;
    }

    .pccs2_filters_search_container {
        display: flex;
        flex-direction: column;
        width: 100%;
        padding: 10px;
    }
```

```
    .pccs2_fsc_title_container {
        width: 100%;
        height: auto;
        text-align: center;
    }

    .pccs2_fsc_main_container {
        width: 100%;
        height: auto;
        display: flex;
        flex-direction: row;
    }

    .pccs2_filters_search_container ul {
        list-style-type: none;
        padding: 0;
    }

    .pccs2_filters_search_container li {
        padding: 10px;
        border: 2px solid #000000;
        margin-top: 10px;
    }

    .pccs2_age_filter_container {
        width: 33.33%;
        height: auto;
        text-align: center;
        padding: 10px;
    }

    .pccs2_nationality_filter_container {
        width: 33.33%;
        height: auto;
        text-align: center;
        padding: 10px;
    }

    .pccs2_position_filter_container {
        width: 33.33%;
        height: auto;
        text-align: center;
        padding: 10px;
    }

    /* Player Comparison Section 2 - Search/Filters Section End */
```

```
    /* Player Comparison Section 2 - Main Results Section Start */

    .pcc_mrcs2_title_container {
        width: 100%;
        height: auto;
        text-align: center;
        margin-bottom: 20px;
    }

    .pcc_mrcs2_players {
        display: flex;
        flex-direction: row;
        width: 100%;
        height: auto;
    }

    .pcc_mrcs2_players2 {
        display: flex;
        flex-direction: column;
        width: 100%;
        height: auto;
        border: 2px solid yellow;
        text-align: center;
    }

    .pcc_mrcs2_players2 a {
        color: #FF0000;
        transition: 0.5s;
    }

    .pcc_mrcs2_players2 a:hover {
        color: #000000;
        transition: 0.5s;
    }

    .pcc_mrcs2_player_images {
        width: 200px;
        height: 200px;
        object-fit: contain;
        margin: auto;
    }

    .pcc_mrsc2_statistic_containers {
        width: 100%;
        height: auto;
        border: 2px solid #000000;
        margin-top: 10px;
    }

    /* Player Comparison Section 2 - Main Results Section End */
```

```
    /* hr Section Start */

    .pcc_hr_section {
        width: 10%;
        height: auto;
    }

    .pcc_hr_section hr {
        width: 1%;
        height: 100%;
        transform: rotate(180deg);
        border: 2px solid #000000;
    }

    /* hr Section End */

/* Main Content Section End */

/* 'PLAYER COMPARISON' PAGE END */

/* ------------------------------------------------------------------------ */
```

'stylesheet.css' File – Mobile Code

```
/* MOBILE DEVICE MEDIA QUERY START */

@media screen and (max-width: 960px) and (min-width: 0px) {

    /* TEMPLATE STYLES START */

        /* Header Section Start */

        .header_navigation_container {
            display: flex;
            justify-content: center;
            width: 100%;
            height: auto;
        }

        .header_navigation_container ul {
            display: flex;
            flex-direction: column;
            padding: 0;
        }

        .header_navigation_container li {
            margin-top: 10px;
            margin-left: 0px;
            margin-right: 0px;
        }

        /* Header Section End */

        /* Footer Section Start */

        .footer_container {
            display: flex;
            flex-direction: column;
        }

        .fc_sub_container1 {
            display: flex;
            flex-direction: column;
            width: 100%;
            height: auto;
            text-align: center;
        }

        .fc_logo_container {
            width: 30%;
            height: auto;
        }
```

```
        .fc_sources_container {
            width: 100%;
            height: auto;
            padding-left: 20px;
            padding-right: 20px;
        }

        .fc_sub_container2 {
            display: flex;
            flex-direction: column;
            width: 100%;
            height: auto;
            padding-right: 0px;
        }

        .fc_hr_container {
            width: 100%;
            height: auto;
        }

        .fc_hr_container hr {
            width: 100%;
            height: auto;
            transform: rotate(0deg);
        }

        .fc_plsc_heading_container hr {
            width: 50%;
            height: auto;
        }

        /* Footer Section End */

    /* TEMPLATE STYLES END */

    /* ---------------------------------------------

    /* 'HOME/INTRODUCTION' PAGE START */
```

```
        /* Application Welcome Section Start */

        .page_welcome_container {
            padding: 20px;
        }

        .page_welcome_sub_container {
            width: 100%;
            height: auto;
        }

        /* Application Welcome Section End */

        /* 'Individual Players' Page Information Section Start */

        .individual_players_container {
            display: flex;
            flex-direction: column;
            padding: 20px;
        }

        .individual_players_sub_container {
            display: flex;
            flex-direction: column;
            width: 100%;
            height: auto;
        }

        .ipc_title_container {
            width: 100%;
            height: auto;
        }

        .ipc_title_container i {
            font-size: 200px;
        }

        .ipc_description_container {
            width: 100%;
            height: auto;
            text-align: center;
        }

        /* 'Individual Players' Page Information Section End */
```

```
        /* 'Player Comparison' Page Information Section Start */

        .player_comparison_container {
            display: flex;
            flex-direction: column;
            padding: 20px;
        }

        .player_comparison_sub_container {
            display: flex;
            flex-direction: column;
            width: 100%;
            height: auto;
        }

        .pcc_title_container {
            width: 100%;
            height: auto;
            order: 1;
        }

        .pcc_title_container i {
            font-size: 200px;
        }

        .pcc_description_container {
            width: 100%;
            height: auto;
            text-align: center;
            order: 2;
        }

        /* 'Player Comparison' Page Information Section End */

    /* 'HOME/INTRODUCTION' PAGE END */

    /* ---------------------------------------------
```

```
/* 'INDIVIDUAL PLAYERS' PAGE START */

    /* Page Heading Section Start */

    .page_heading_container {
        padding: 20px;
    }

    .page_heading_sub_container {
        width: 100%;
        height: auto;
    }

    /* Page Heading Section End */

    /* Main Content Section Start */

        /* Main Content Section - Players Section Start */

        .psc_player_container {
            display: flex;
            flex-direction: column;
            width: 100%;
            height: auto;
            text-align: center;
            border: 2px solid blue;
        }

        .psc_pc_players {
            display: flex;
            flex-direction: column;
        }

        .psc_pc_players2 {
            padding: 30px;
        }

        .psc_pc_player_images {
            width: 100%;
            height: auto;
        }

        /* Main Content Section - Players Section End */

    /* Main Content Section End */

/* 'INDIVIDUAL PLAYERS' PAGE END */

/* ------------------------------------------------------------------------------ */
```

```
/* 'SHOW' PAGE (ACTUAL INDIVIDUAL PLAYERS) START */

    /* Player Overview Section Start */

    .player_overview_container {
        display: flex;
        flex-direction: column;
        padding: 20px;
    }

    .poc_general_information_container {
        display: flex;
        flex-direction: column;
    }

    .poc_gic_image_container {
        width: 100%;
        height: auto;
    }

    .poc_gic_information_container {
        width: 100%;
        height: auto;
        padding-left: 0px;
    }

    /* Player Overview Section End */

    /* Player Statistics Section Start */

    .psc2_categories_container {
        display: flex;
        flex-direction: column;
    }

    .psc2_cc_attack {
        width: 100%;
        height: auto;
    }

    .psc2_cc_defence {
        width: 100%;
        height: auto;
    }

    .psc2_cc_general {
        width: 100%;
        height: auto;
    }

    .psc2_cc_discipline {
        width: 100%;
        height: auto;
    }

    /* Player Statistics Section End */

/* 'SHOW' PAGE (ACTUAL INDIVIDUAL PLAYERS) END */

/* ------------------------------------------------------------------------------ */

/* 'PLAYER COMPARISON' PAGE START */

    /* Main Content Section Start */

    .player_comparison_container {
        display: flex;
        flex-direction: column;
    }

        /* Player Comparison Section 1 Start */

        .pcc_section1 {
            width: 100%;
            height: auto;
        }

            /* Player Comparison Section 1 - Search/Filters Section Start */

            .pccs1_fsc_main_container {
                width: 100%;
                height: auto;
                display: flex;
                flex-direction: row;
            }

            /* Player Comparison Section 1 - Search/Filters Section End */

            /* Player Comparison Section 1 - Main Results Section Start */
```

```
            /* Player Comparison Section 1 - Main Results Section Start */

            .pcc_mrcs1_players {
                display: flex;
                flex-direction: column;
            }

            /* Player Comparison Section 1 - Main Results Section End */

        /* Player Comparison Section 1 End */

        /* Player Comparison Section 2 Start */

        .pcc_section2 {
            width: 100%;
            height: auto;
        }

            /* Player Comparison Section 2 - Search/Filters Section Start */

            .pccs2_filter_container {
                margin-top: 40px;
            }

            /* Player Comparison Section 2 - Search/Filters Section End */

            /* Player Comparison Section 2 - Main Results Section Start */

            .pcc_mrcs2_title_container {
                width: 100%;
                height: auto;
                text-align: center;
                margin-bottom: 20px;
            }

            .pcc_mrcs2_players {
                display: flex;
                flex-direction: column;
            }

            /* Player Comparison Section 2 - Main Results Section End */
```

```
            /* hr Section Start */

            .pcc_hr_section {
                width: 100%;
                height: auto;
                margin-top: 40px;
            }

            .pcc_hr_section hr {
                width: 100%;
                height: auto;
                transform: rotate(0deg);
            }

            /* hr Section End */

        /* Main Content Section End */

    /* 'PLAYER COMPARISON' PAGE END */

}

/* MOBILE DEVICE MEDIA QUERY END */
```

Conclusion

Throughout this project, I was able to challenge myself with new technologies as well as enhancing both my 'front-end' and 'back-end' website development skillsets. Despite not having achieved everything that I wanted to, I was still pleased with the outcome. This was because I was able to adapt quickly to learning 'Laravel', something which I had never used before as well as improving my skills with databases and data. Overall, I believe this project was a success.

Reference List/Bibliography/Acknowledgements for the Development Process of this Project

API Platform (n.d.) Getting Started with API Platform: Hypermedia and GraphQL API, Admin and Progressive Web App. Available at: *https://api-platform.com/docs/distribution/*. [Accessed 2 March 2019], [online].

Bootstrap (n.d.) Getting Started. Available at: *https://getbootstrap.com/docs/3.3/getting-started/*. [Accessed 11 March 2019], [online].

Buckler, C. (2009) How to Install MySQL. *Programming.* March 24. Available at: *https://www.sitepoint.com/how-to-install-mysql/*. [Accessed 26 February 2019], [online].

Codecademy (2019) Learn to Code – for Free | Codecademy. Available at: *https://www.codecademy.com/*. [Accessed 23 April 2019], [online].

Codecademy (n.d.) Creating a React App. *Codecademy Articles.* Available at: *https://www.codecademy.com/articles/how-to-create-a-react-app*. [Accessed 2 March 2019], [online].

Coleman, A. (n.d.) Creating a New Route & View In Your Laravel Application. Available at: *https://selftaughtcoders.com/from-idea-to-launch/lesson-13/creating-a-new-route-view-in-your-laravel-application/*. [Accessed 10 April 2019], [online].

Composer (n.d.) Download Composer. Available at: *https://getcomposer.org/download/*. [Accessed 22 April 2019], [online].

Create React App (2019) Adding Images, Fonts, and Files. Available at: *https://facebook.github.io/create-react-app/docs/adding-images-fonts-and-files*. [Accessed 22 February 2019], [online].

Create React App (2019) Create React App Set up a modern web app by running one command.. Available at: *https://facebook.github.io/create-react-app/*. [Accessed 29 April 2019], [online].

Dale, J. (2018) Man Utd's Diogo Dalot says playing for Jose Mourinho 'a dream come true'. *Sky Sports.* Available at: *https://www.skysports.com/football/news/11667/11573127/man-utds-diogo-dalot-says-playing-for-jose-mourinho-a-dream-come-true*. [Accessed 23 April 2019], [online].

Django (2005) How to install Django on Windows. Available at: *https://docs.djangoproject.com/en/2.1/howto/windows/*. [Accessed 2 March 2019], [online].

Django (2005) How to install Django. Available at: *https://docs.djangoproject.com/en/2.1/topics/install/#database-installation*. [Accessed 2 March 2019], [online].

Django (2005) Quick install guide. Available at: *https://docs.djangoproject.com/en/2.1/intro/install/*. [Accessed 2 March 2019], [online].

Docker (2019) Get started with Docker for Windows. Available at: *https://docs.docker.com/docker-for-windows/*. [Accessed 2 March 2019], [online].

Docker (2019) Install Docker Desktop for Windows. Available at: *https://docs.docker.com/docker-for-windows/install/*. [Accessed 2 March 2019], [online].

Edet, A. (2018) Deploying Laravel 5 applications on shared hosting without the use of SSH. March 3. Available at: https://dev.to/asapabedi/deploying-laravel-5-applications-on-shared-hosting-without-the-use-of-ssh--16a6. [Accessed 24 April 2019], [online].

Edwards, D. (2018) What the hell is going on with Fred? Manchester United's £50m summer signing struggling for game time. *Goal.com.* Available at: https://www.goal.com/en-us/news/what-the-hell-is-going-on-with-fred-manchester-united-s-50m/1h0dtvqrgc05310eaoh22yuf6j. [Accessed 23 April 2019], [online].

El Universo (2019) Antonio Valencia busca un equipo para firmar por dos años. *El Universo.* Available at: https://www.eluniverso.com/deportes/2019/03/15/nota/7233275/valencia-busca-equipo-firmar-dos-anos. [Accessed 23 April 2019], [online].

Font Awesome (n.d.) React. Available at: *https://fontawesome.com/how-to-use/on-the-web/using-with/react*. [Accessed 29 April 2019], [online].

Google (n.d.) how to add data to table php myadmin – Google Search. Available at: *https://www.google.com/search?q=how+to+add+data+to+table+php+myadmin&oq=how+to+add+data+to+table+php+myadmin&aqs=chrome..69i57j0.6997j0j8&sourceid=chrome&ie=UTF-8*. [Accessed 16 March 2019], [online].

Google (n.d.) how to add new field into phpmyadmin table – Google Search. Available at: *https://www.google.com/search?q=how+to+add+new+field+into+php+myadmin+table&oq=how+to+add+new+field+into+php+myadmin+table&aqs=chrome..69i57.10456j0j8&sourceid=chrome&ie=UTF-8*. [Accessed 16 March 2019], [online].

Google Fonts (n.d.) Google Fonts. Available at: *https://fonts.google.com/*. [Accessed 24 April 2019], [online].

HostPapa (2017) How to Change your PHP Version in cPanel. *Shared Hosting.* March 29. Available at: https://www.hostpapa.co.uk/knowledgebase/how-to-change-php-version-cpanel/. [Accessed 24 April 2019], [online].

jQuery (2019) jQuery CDN – Latest Stable Versions. Available at: *http://code.jquery.com/*. [Accessed 11 March 2019], [online].

Just Laravel (2017) *How to implement search functionality in laravel.* Available at: *https://www.youtube.com/watch?v=RJqHwsr3Jhs*. [Accessed 11 March 2019], [online].

Kelleher, M. (2018) Eric Bailly set to return for Manchester United against Huddersfield. *Sky Sports.* Available at: https://www.skysports.com/football/news/11667/11253116/eric-bailly-set-to-return-for-manchester-united-against-huddersfield. [Accessed 23 April 2019], [online].

Kerai, H. (2018) Andreas Pereira relying on Manchester United boss Jose Mourinho to make him a winner. *Sky Sports.* Available at: https://www.skysports.com/football/news/11667/11475554/andreas-pereira-relying-on-manchester-united-boss-jose-mourinho-to-make-him-a-winner. [Accessed 23 April 2019], [online].

Korop, P. (2017) Laravel Blade foreach "trick": splitting results into chunks. October 18. Available at: https://laraveldaily.com/laravel-blade-foreach-trick-splitting-results-chunks/. [Accessed 15 April 2019], [online].

Laracasts (2016) upload project laravel. Available at: *https://laracasts.com/discuss/channels/laravel/upload-project-laravel*. [Accessed 24 April 2019], [online].

Laracasts (2017) Call to undefined method Illuminate\Database\Query\Builder::all(). Available at: *https://laracasts.com/discuss/channels/laravel/call-to-undefined-method-illuminatedatabasequerybuilderall*. [Accessed 16 March 2019], [online].

Laracasts (2017) How to retrieve image from database and display it on view using blade in Laravel. Available at: *https://laracasts.com/discuss/channels/laravel/how-to-retrieve-image-from-database-and-display-it-on-view-using-blade-in-laravel*. [Accessed 11 March 2019], [online].

Laracasts (2017) laravel eloquent get 10 records from id = 10. Available at: *https://laracasts.com/discuss/channels/eloquent/laravel-eloquent-get-10-records-from-id-10*. [Accessed 15 April 2019], [online].

Laracasts (2017) Laravel how to get values from a table with an id from another table. Available at: *https://laracasts.com/discuss/channels/laravel/laravel-how-to-get-values-from-a-table-with-an-id-from-another-table*. [Accessed 15 April 2019], [online].

Laracasts (2017) Unable to get route::resource working with new route.. Available at: *https://laracasts.com/discuss/channels/laravel/unable-to-get-routeresource-working-with-new-route*. [Accessed 11 March 2019], [online].

Laravel (n.d.) Authentication. Available at: *https://laravel.com/docs/5.8/authentication*. [Accessed 7 March 2019], [online].

Laravel (n.d.) Collections. Available at: *https://laravel.com/docs/5.5/collections#method-chunk*. [Accessed 19 March 2019], [online].

Laravel (n.d.) Database: Migrations. Available at: *https://laravel.com/docs/5.8/migrations#creating-columns*. [Accessed 11 March 2019], [online].

Laravel (n.d.) Installation. Available at: *https://laravel.com./docs/5.8*. [Accessed 22 April 2019], [online].

Laravel (n.d.) Laravel Scout. Available at: *https://laravel.com/docs/5.8/scout#searching*. [Accessed 11 March 2019], [online].

Laravel (n.d.) Routing. Available at: *https://laravel.com/docs/5.8/routing*. [Accessed 19 March 2019], [online].

Laravel (n.d.) The PHP Framework for Web Artisans. Available at: *https://laravel.com/*. [Accessed 23 April 2019], [online].

Laurence, W. (2019) Scott McTominay Signs New Contract With Manchester United Until June 2023. *90min.* Available at: *https://www.90min.com/posts/6276736-scott-mctominay-signs-new-contract-with-manchester-united-until-june-2023*. [Accessed 23 April 2019], [online].

Lemmons, S. & Simoens, M. (2018) Ext JS to React: Load, Sort and Filter Data with React. April 12. Available at: *https://moduscreate.com/blog/ext-js-to-react-load-sort-and-filter-data-with-react/*. [Accessed 22 February 2019], [online].

Liew, J. (2019) Poacher, creator and saboteur, how Jesse Lingard's diverse talents grease the Manchester United wheels. *The Independent.* Available at: https://www.independent.co.uk/sport/football/premier-league/manchester-united-jesse-lingard-tottenham-a8727041.html. [Accessed 23 April 2019], [online].

MAMP (2019) Downloads – MAMP & MAMP PRO. Available at: *https://www.mamp.info/en/downloads/?m=1530180376&*. [Accessed 22 April 2019], [online].

Mirror (2019) Man Utd's David de Gea a wanted man again after Zinedine's return. *GAMEYetu!.* Available at: https://www.standardmedia.co.ke/sports/article/2001316961/man-utd-s-david-de-gea-a-wanted-man-again-after-zinedine-s-return. [Accessed 23 April 2019], [online].

Nethala, A. (2018) Search functionality in Laravel. August 5. Available at: https://medium.com/justlaravel/search-functionality-in-laravel-a2527282150b. [Accessed 11 March 2019], [online].

Oracle (2019) 2.3.3 MySQL Installer for Windows. Available at: *https://dev.mysql.com/doc/refman/8.0/en/mysql-installer.html*. [Accessed 2 March 2019], [online].

Oracle (2019) 2.3.3.1 MySQL Installer Initial Setup. Available at: *https://dev.mysql.com/doc/refman/8.0/en/mysql-installer-setup.html*. [Accessed 2 March 2019], [online].

Oracle (2019) Download MySQL Installer. Available at: *https://dev.mysql.com/downloads/installer/*. [Accessed 2 March 2019], [online].

Oracle (2019) MySQL Downloads. Available at: *https://www.mysql.com/downloads/*. [Accessed 2 March 2019], [online].

pngimg.com (n.d.) Download PNG image: Manchester United logo PNG. Available at: *http://pngimg.com/download/21866*. [Accessed 10 April 2019], [online].

Premier League (2019) Anthony Martial. Available at: *https://www.premierleague.com/players/11272/Anthony-Martial/overview*. [Accessed 11 March 2019], [online].

Premier League (2019) Premier League Football News, Fixtures, Scores & Results. Available at: *https://www.premierleague.com/*. [Accessed 23 April 2019], [online].

Press Association (2018) Manchester United's Matteo Darmian says he misses Serie A. *Sky Sports.* Available at: https://www.skysports.com/football/news/11667/11576096/manchester-uniteds-matteo-darmian-says-he-misses-serie-a. [Accessed 23 April 2019], [online].

PyPA (2008) Installation. Available at: *https://pip.pypa.io/en/latest/installing/*. [Accessed 2 March 2019], [online].

Python.org (2001) Download Python | Python.org. Available at: *https://www.python.org/downloads/*. [Accessed 2 March 2019], [online].

Quora (n.d.) How can I handle a database with React.js?. Available at: https://www.quora.com/How-can-I-handle-a-database-with-React-js. [Accessed 26 February 2019], [online].

Quora (n.d.) What's the best Database to implement in a ReactJS project, I want to use MySQL but can't find a tutorial on how to use it with ReactJS? Pls advice.. Available at:

https://www.quora.com/Whats-the-best-Database-to-implement-in-a-ReactJS-project-I-want-to-use-MySQL-but-cant-find-a-tutorial-on-how-to-use-it-with-ReactJS-Pls-advice. [Accessed 26 February 2019], [online].

React (2019) React A JavaScript library for building user interfaces. Available at: *https://reactjs.org/*. [Accessed 29 April 2019], [online].

Reddit (2015) How to make functional search field?. Available at: *https://www.reddit.com/r/laravel/comments/2tyg5l/how_to_make_functional_search_field/?depth=1*. [Accessed 11 March 2019], [online].

Sevilleja, C. (2018) Using Font Awesome 5 with React. September 6. Available at: https://scotch.io/tutorials/using-font-awesome-5-with-react#toc-using-font-awesome-5-and-react. [Accessed 29 April 2019], [online].

Shaikh, S. (2015) Getting Started with Node.js + MySQL. April 28. Available at: https://www.codementor.io/codeforgeek/node-js-mysql-8b5nh45r9. [Accessed 26 February 2019], [online].

Shots, W. E. (2000) What is "the Shell"?. Available at: *http://linuxcommand.org/lc3_lts0010.php*. [Accessed 2 March 2019], [online].

Silas, K. (2018) Using data in React with the Fetch API and axios. *Articles.* August 3. Available at: https://css-tricks.com/using-data-in-react-with-the-fetch-api-and-axios/. [Accessed 22 February 2019], [online].

Sky Sports News (2019) Alexis Sanchez could leave Manchester United and return to Arsenal, says Marc Overmars. *Sky Sports.* Available at: https://www.skysports.com/football/news/11095/11638008/alexis-sanchez-could-leave-manchester-united-and-return-to-arsenal-says-marc-overmars. [Accessed 23 April 2019], [online].

Stack Academy (2017) *Laravel 5 Tutorials - Filter the posts by category - Part 1.* Available at: *https://www.youtube.com/watch?v=zRysRrzSS3Y*. [Accessed 16 March 2019], [online].

Stack Overflow (2010) PDOException "could not find driver". Available at: *https://stackoverflow.com/questions/2852748/pdoexception-could-not-find-driver*. [Accessed 7 March 2019], [online].

Stack Overflow (2013) Laravel - Pass more than one variable to view. Available at: *https://stackoverflow.com/questions/20110757/laravel-pass-more-than-one-variable-to-view*. [Accessed 16 March 2019], [online].

Stack Overflow (2013) Laravel 4 -> Routes on the same controller with multiple variables. Available at: *https://stackoverflow.com/questions/19633565/laravel-4-routes-on-the-same-controller-with-multiple-variables*. [Accessed 14 March 2019], [online].

Stack Overflow (2013) Laravel Sortby Filter Search Products by Select. Available at: *https://stackoverflow.com/questions/48398227/laravel-sortby-filter-search-products-by-select*. [Accessed 14 March 2019], [online].

Stack Overflow (2013) multiple routes in single Route::get() call Laravel 4. Available at: *https://stackoverflow.com/questions/17489492/multiple-routes-in-single-routeget-call-laravel-4/21970797*. [Accessed 19 March 2019], [online].

Stack Overflow (2014) How to stop lines of code from automatically shifting to the next line whenever I resize the window of the editor?. Available at: *https://stackoverflow.com/questions/24664325/how-to-stop-lines-of-code-from-automatically-shifting-to-the-next-line-whenever/24695920*. [Accessed 24 April 2019], [online].

Stack Overflow (2014) Laravel foreach with multiple objects?. Available at: *https://stackoverflow.com/questions/25490844/laravel-foreach-with-multiple-objects*. [Accessed 16 March 2019], [online].

Stack Overflow (2015) Filter with dropdown Laravel. Available at: *https://stackoverflow.com/questions/28294970/filter-with-dropdown-laravel*. [Accessed 14 March 2019], [online].

Stack Overflow (2015) React JSX, how to render text with a single quote? Example I've . Available at: *https://stackoverflow.com/questions/32979512/react-jsx-how-to-render-text-with-a-single-quote-example-pive-p*. [Accessed 22 February 2019], [online].

Stack Overflow (2015) SQLSTATE[42S22]: Column not found: 1054 Unknown column 'id' in 'where clause' (SQL: select * from `songs` where `id` = 5 limit 1). Available at: *https://stackoverflow.com/questions/29347253/sqlstate42s22-column-not-found-1054-unknown-column-id-in-where-clause-s*. [Accessed 16 March 2019], [online].

Stack Overflow (2015) Undefined variable: $ Laravel 5. Available at: *https://stackoverflow.com/questions/31264242/undefined-variable-laravel-5*. [Accessed 16 March 2019], [online].

Stack Overflow (2016) COMPOSER: choose the command line php you want to use. There is nothing there. Available at: *https://stackoverflow.com/questions/37391068/composer-choose-the-command-line-php-you-want-to-use-there-is-nothing-there*. [Accessed 22 April 2019], [online].

Stack Overflow (2016) Laravel : SQLSTATE[42S22]: Column not found: 1054 Unknown column. Available at: *https://stackoverflow.com/questions/40600550/laravel-sqlstate42s22-column-not-found-1054-unknown-column*. [Accessed 16 March 2019], [online].

Stack Overflow (2016) Loop row in bootstrap every 3 columns. Available at: *https://stackoverflow.com/questions/40561301/loop-row-in-bootstrap-every-3-columns/40562841*. [Accessed 10 April 2019], [online].

Stack Overflow (2017) Including a css file in a blade template?. Available at: *https://stackoverflow.com/questions/45279612/including-a-css-file-in-a-blade-template/45290308*. [Accessed 10 March 2019], [online].

Stack Overflow (2017) Laravel 5.5 Error 500 in Cpanel Shared Hosting. Available at: *https://stackoverflow.com/questions/47784923/laravel-5-5-error-500-in-cpanel-shared-hosting*. [Accessed 24 April 2019], [online].

Stack Overflow (2017) laravel 5.5 The page has expired due to inactivity. Please refresh and try again. Available at: *https://stackoverflow.com/questions/46149561/laravel-5-5-the-page-has-expired-due-to-inactivity-please-refresh-and-try-again/48719525*. [Accessed 14 March 2019], [online].

Stack Overflow (2017) Laravel Eloquent PHP how to get all rows in MySQL table with unique column values as keys. Available at: *https://stackoverflow.com/questions/44533000/laravel-eloquent-php-how-to-get-all-rows-in-mysql-table-with-unique-column-value*. [Accessed 15 April 2019], [online].

Stack Overflow (2017) Laravel not load a view for specific route. Available at: *https://stackoverflow.com/questions/44347986/laravel-not-load-a-view-for-specific-route*. [Accessed 16 March 2019], [online].

Stack Overflow (2017) Laravel vendor/autoload is missing. Available at: *https://stackoverflow.com/questions/46048519/laravel-vendor-autoload-is-missing*. [Accessed 22 April 2019], [online].

Stack Overflow (2017) Load local images in React.js. Available at: *https://stackoverflow.com/questions/44154939/load-local-images-in-react-js*. [Accessed 22 February 2019], [online].

Stack Overflow (2018) (Laravel) Get data from a table which an ID correspond with another table that link with that table. Available at: *https://stackoverflow.com/questions/42268591/laravel-get-data-from-a-table-which-an-id-correspond-with-another-table-that-l*. [Accessed 15 April 2019], [online].

Stack Overflow (2018) count(): Parameter must be an array or an object that implements Countable. Available at: *https://stackoverflow.com/questions/48343557/count-parameter-must-be-an-array-or-an-object-that-implements-countable*. [Accessed 16 March 2019], [online].

Stack Overflow (2018) How to use JOIN in Laravel when two columns of one table refer to one column of another table?. Available at: *https://stackoverflow.com/questions/53222776/how-to-use-join-in-laravel-when-two-columns-of-one-table-refer-to-one-column-of*. [Accessed 15 April 2019], [online].

Stack Overflow (2018) How to use multiple method in single route in laravel. Available at: *https://stackoverflow.com/questions/50612904/how-to-use-multiple-method-in-single-route-in-laravel*. [Accessed 16 March 2019], [online].

Stack Overflow (2018) Storing a photo in Laravel. Migration file structure,. Available at: *https://stackoverflow.com/questions/48945935/storing-a-photo-in-laravel-migration-file-structure*. [Accessed 11 March 2019], [online].

Stack Overflow (2019) MySQL Server on MAMP-Windows Will Not Start. Available at: *https://stackoverflow.com/questions/54296014/mysql-server-on-mamp-windows-will-not-start*. [Accessed 31 March 2019], [online].

The PHP Group (2001) PHP for Windows: Binaries and Sources Releases. php.net. The PHP Group. Available at: *https://windows.php.net/download#php-7.3*. [Accessed 22 April 2019], [online].

Thomas, L. & Cooper, J. (2019) Anthony Martial close to agreeing new Manchester United contract. *Sky Sports.* Available at: *https://www.skysports.com/football/news/11667/11611085/anthony-martial-close-to-agreeing-new-manchester-united-contract*. [Accessed 11 March 2019], [online].

Thomas, L. (2016) Juan Mata prepares for Man Utd match at Liverpool in Spain. *Sky Sports.* Available at: *https://www.skysports.com/football/news/11667/10612590/juan-mata-prepares-for-man-utd-match-at-liverpool-in-spain*. [Accessed 23 April 2019], [online].

Traversy Media (2017) *Deploy Laravel To Shared Hosting The Easy Way.* Available at: *https://www.youtube.com/watch?v=6g8G3YQtQt4*. [Accessed 24 April 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 1] - Series Introduction.* Available at: *https://www.youtube.com/watch?v=EU7PRmCpx-0*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 10] - Model Relationships.* Available at: *https://www.youtube.com/watch?v=42l4nHl_aUM*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 12] - File Uploading & Finishing Up.* Available at: *https://www.youtube.com/watch?v=AL8PCThJ9c4*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 2] - Environment Setup & Laravel Installation.* Available at: *https://www.youtube.com/watch?v=H3uRXvwXz1o*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 3] - Basic Routing & Controllers.* Available at: *https://www.youtube.com/watch?v=sLFNVXY0APk*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 4] - Blade Templating & Compiling Assets.* Available at: *https://www.youtube.com/watch?v=bSG2YMqJJys*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 5] - Models & Database Migrations.* Available at: *https://www.youtube.com/watch?v=neSHAWdE44c*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 6] - Fetching Data With Eloquent.* Available at: *https://www.youtube.com/watch?v=emyIlJPxZr4&list=PLillGF-RfqbYhQsN5WMXy6VsDMKGadrJ-&index=6*. [Accessed 10 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 7] - Forms & Saving Data.* Available at: *https://www.youtube.com/watch?v=-QapNzUE4V0*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 8] - Edit & Delete Data.* Available at: *https://www.youtube.com/watch?v=PAP8IS_ak6w*. [Accessed 11 March 2019], [online].

Traversy Media (2017) *Laravel From Scratch [Part 9] - User Authentication.* Available at: *https://www.youtube.com/watch?v=ORus3-By4lk*. [Accessed 11 March 2019], [online].

Tutorials4urHelp (2017) *How to create controller and view in Laravel 5.5.* Available at: *https://www.youtube.com/watch?v=Wf4dRTFib4o*. [Accessed 10 April 2019], [online].

W3Schools (1999) CSS Media Queries - Examples. Available at: *https://www.w3schools.com/css/css3_mediaqueries_ex.asp*. [Accessed 23 April 2019], [online].

W3Schools (1999) Font Awesome 5 Arrow Icons. Available at: *https://www.w3schools.com/icons/fontawesome5_icons_arrows.asp*. [Accessed 19 April 2019], [online].

W3Schools (1999) Font Awesome 5 Sport Icons. Available at: *https://www.w3schools.com/icons/fontawesome5_icons_sports.asp*. [Accessed 23 April 2019], [online].

W3Schools (1999) How TO - Search Bar. Available at: *https://www.w3schools.com/howto/howto_css_searchbar.asp*. [Accessed 10 March 2019], [online].

W3Schools (1999) How TO - Search Bar. Available at: *https://www.w3schools.com/howto/howto_css_searchbar.asp*. [Accessed 29 April 2019], [online].

W3Schools (1999) HTML <meta> charset Attribute. Available at: *https://www.w3schools.com/tags/att_meta_charset.asp*. [Accessed 23 April 2019], [online].

W3Schools (1999) HTML <select> Tag. Available at: *https://www.w3schools.com/tags/tag_select.asp*. [Accessed 10 March 2019], [online].

W3Schools (1999) HTML Color Picker. Available at: *https://www.w3schools.com/colors/colors_picker.asp*. [Accessed 24 April 2019], [online].

W3Schools (1999) Icons Reference. Available at: *https://www.w3schools.com/icons/icons_reference.asp*. [Accessed 23 April 2019], [online].

W3Schools (1999) Node.js MySQL. Available at: *https://www.w3schools.com/nodejs/nodejs_mysql.asp*. [Accessed 2 March 2019], [online].

W3Schools (1999) Responsive Web Design - Media Queries. Available at: *https://www.w3schools.com/css/css_rwd_mediaqueries.asp*. [Accessed 23 April 2019], [online].

W3Schools (1999) Responsive Web Design - The Viewport. Available at: *https://www.w3schools.com/css/css_rwd_viewport.asp*. [Accessed 23 April 2019], [online].

W3Schools (1999) Tryit Editor v3.6. Available at: *https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_searchbar2*. [Accessed 19 February 2019], [online].

W3Schools (1999) Tryit Editor v3.6. Available at: *https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_searchbar2*. [Accessed 29 April 2019], [online].

W3Schools (1999) W3Schools Online Web Tutorials. Available at: *http://www.w3schools.com/*. [Accessed 23 April 2019], [online].

Wikimedia (2017) File:CSKA-MU 2017 (15).jpg. Available at: *https://commons.wikimedia.org/wiki/File:CSKA-MU_2017_(15).jpg*. [Accessed 23 April 2019], [online].

Wikimedia (2017) File:Romelu Lukaku 27 September 2017 cropped.jpg. Available at: *https://commons.wikimedia.org/wiki/File:Romelu_Lukaku_27_September_2017_cropped.jpg*. [Accessed 23 April 2019], [online].

Wikimedia Commons (2015) File:Shaw - July 2015 (cropped).jpg. Available at: *https://commons.wikimedia.org/wiki/File:Shaw_-_July_2015_(cropped).jpg*. [Accessed 23 April 2019], [online].

Wikimedia Commons (2017) File:Cskamu 43.jpg. Available at: *https://commons.wikimedia.org/wiki/File:Cskamu_43.jpg*. [Accessed 23 April 2019], [online].

Wikimedia Commons (2017) File:Cskamu 43.jpg. Available at: *https://commons.wikimedia.org/wiki/File:Cskamu_43.jpg*. [Accessed 23 April 2019], [online].

Wikimedia Commons (2017) File:Marcos Rojo vs Rostov.jpg. Available at: *https://commons.wikimedia.org/wiki/File:Marcos_Rojo_vs_Rostov.jpg*. [Accessed 23 April 2019], [online].

Wikimedia Commons (2017) File:Paul Pogba 2017-03-09.jpg. Available at: *https://commons.wikimedia.org/wiki/File:Paul_Pogba_2017-03-09.jpg*. [Accessed 23 April 2019], [online].

Wikimedia Commons (2018) File:CSKA-MU 2017 (6).jpg. Available at: *https://commons.wikimedia.org/wiki/File:CSKA-MU_2017_(6).jpg*. [Accessed 23 April 2019], [online].

Wikimedia Commons (2018) File:Sergio Romero (ManUtd).jpg. Available at: *https://commons.wikimedia.org/wiki/File:Sergio_Romero_(ManUtd).jpg*. [Accessed 23 April 2019], [online].

Wikipedia (2019) Ander Herrera. Available at: *https://en.wikipedia.org/wiki/Ander_Herrera*. [Accessed 23 April 2019], [online].

Wikipedia (2019) Ashley Young. Available at: *https://en.wikipedia.org/wiki/Ashley_Young*. [Accessed 23 April 2019], [online].

worldfootball.net (2000) Marcus Rashford. Available at: *https://www.worldfootball.net/player_summary/marcus-rashford/*. [Accessed 11 March 2019], [online].

worldfootball.net (2000) Nemanja Matić. Available at: *https://www.worldfootball.net/player_summary/nemanja-matic/*. [Accessed 23 April 2019], [online].

During the very initial stages, I was introduced to 'Laravel' as well as being helped to resolve a few issues by a fellow class colleague

I may have also used some of my notes from the introductory coding courses I attended at Bath College

I may have also looked at and used some lecture notes

I may have also used some help and advice from previous work experience at 'The ICE Agency' and 'Coullweb' as well as previous projects

THIS IS THE END OF THE DOCUMENT