MODULE CODE: GRLA09003

MODULE TITLE: Software Engineering Practice

MODULE LEADER: Dr Joanna Isabelle Olszewska

TITLE: Summative Assignment 2

STUDENT: Grant Martin

BANNER NUMBER: B00340010

UNIVERSITY: University of the West of Scotland

PROJECT TITLE: Android Football Team and Player table

PROJECT PURPOSE: The purpose of the project is to implement an Android mobile application written in the coding language Java, the purpose of the Android mobile application is to allow non-professional football teams to create leagues and monitor their teams statistics and player statistics over the duration of the football season. This document aims to explain how the student was able to implement the project and details what data structure, search algorithm and sort algorithm was used to deliver the project.

WORKING DOCUMENT

LAST UPDATED: 26.04.2020

# 1 Contents

[1 Contents 2](#_Toc38825393)

[2 Introduction to Project 5](#_Toc38825394)

[2.1 Overview 5](#_Toc38825395)

[3 Detailed Requirements Specification 6](#_Toc38825396)

[3.1 MoSCoW Analysis 7](#_Toc38825397)

[3.1.1 Must 7](#_Toc38825398)

[3.1.2 Should 7](#_Toc38825399)

[3.1.3 Could 7](#_Toc38825400)

[3.1.4 Won’t 7](#_Toc38825401)

[3. 2 User Stories 8](#_Toc38825402)

[3.2.1 Must User Stories 8](#_Toc38825403)

[3.2.2 Should User Stories 9](#_Toc38825404)

[4 UML Use Case Diagrams 12](#_Toc38825405)

[4.1.1 Login and Registration Use Cases 12](#_Toc38825406)

[4.1.2 Search and Sort Teams Use Cases 12](#_Toc38825407)

[4.1.3 Search and Sort Player Use Cases 13](#_Toc38825408)

[4.1.4 Mobile Application Use Cases 14](#_Toc38825409)

[5 UML Activity/Sequence Diagrams 15](#_Toc38825410)

[5.1.1 Activity/Sequence Diagram: Create a New Account 15](#_Toc38825411)

[5.1.2 Activity/Sequence Diagram: Login 15](#_Toc38825412)

[5.1.3 Activity/Sequence Diagram: Search Teams 16](#_Toc38825413)

[5.1.4 Activity/Sequence Diagram: Search Players 16](#_Toc38825414)

[5.1.5 Activity/Sequence Diagram: Sort Teams 17](#_Toc38825415)

[5.1.6 Activity/Sequence Diagram: Sort Players 17](#_Toc38825416)

[6 Class Diagram 18](#_Toc38825417)

[7 Interface Designs for the Final Android Mobile Application 19](#_Toc38825418)

[7.1 Registration Page 19](#_Toc38825419)

[7.2 Login Page 21](#_Toc38825422)

[7.3 Landing Page 23](#_Toc38825426)

[7.4 Teams Page 24](#_Toc38825427)

[7.4.1 Team Sort Page 25](#_Toc38825429)

[7.4.2 Team Search Page 26](#_Toc38825431)

[7.5 Players Page 27](#_Toc38825432)

[7.5.1 Players Sort 28](#_Toc38825434)

[7.5.2 Players Search 29](#_Toc38825436)

[8 Software Implementation 30](#_Toc38825438)

[8.1 Firebase Authentication 31](#_Toc38825439)

[8.1.1 Register 31](#_Toc38825440)

[8.1.2 Login 31](#_Toc38825441)

[8.2 Data Structure 33](#_Toc38825442)

[8.3 Data Structure Implementation 34](#_Toc38825443)

[8.3.1 Implementation of ArrayList 34](#_Toc38825444)

[8.4 Sort Algorithms 36](#_Toc38825445)

[8.4.1 How the Quick sort works? 37](#_Toc38825446)

[8.5 Bubble Sort Implementation 38](#_Toc38825447)

[8.6 Search Algorithms 39](#_Toc38825448)

[7.6.1 How Binary Search Algorithms work? 39](#_Toc38825449)

[8.7 Binary Search Algorithm Implementation 40](#_Toc38825450)

[9 Adopted Software Testing Process 41](#_Toc38825451)

[9.2 Test Plan and Test Results 41](#_Toc38825452)

[9.2.1 White-box Testing 42](#_Toc38825453)

[10 Conclusion 44](#_Toc38825454)

[10.1 SWOT Analysis of Implemented app 45](#_Toc38825455)

[11 References 46](#_Toc38825456)

[12 Appendix 48](#_Toc38825457)

[12.1 Appendix 1: Proposed Interface Design 48](#_Toc38825458)

[Homepage of Football Team and Player Data Store Android app 48](#_Toc38825459)

[Teams Page of Football Team and Player Data Store Android app 49](#_Toc38825460)

[Players Page of Football Team and Player Data Store Android app 50](#_Toc38825462)

[12.2 Appendix 2: Data Flow Diagrams 51](#_Toc38825463)

[Data flow diagram for navigating to Players Page and presenting the Players table 51](#_Toc38825464)

[Data flow diagram for navigating to Teams Page and presenting the Teams table 51](#_Toc38825465)

[Data Flow Diagram for applying sort to Teams table 51](#_Toc38825466)

[Data Flow Diagram for applying sort to Players table 52](#_Toc38825467)

[Data Flow Diagram for applying search to Players table 52](#_Toc38825468)

[Data Flow Diagram for applying search to Teams table 53](#_Toc38825469)

[12.3 Appendix 3: Risk Analysis 54](#_Toc38825470)

[12.4.1 Test Cases 57](#_Toc38825471)

[12.4.2 Test Results 69](#_Toc38825472)

# 2 Introduction to Project

## 2.1 Overview

The purpose of the students delivered software project was to implement an Android mobile app using the object oriented language Java, to allow non-professional football teams to create leagues and monitor their teams statistics, as well as individual player statistics to help teams and players track their performances over a season. As part of the learning objectives the delivered software uses data structures and algorithms for sorting and searching the data, which allows the users to extract and find specific statistics about teams and players efficiently and easily. As part of the project all the functional and non-functional requirements have be identified, described in detail and priorities using MoSCoW analysis. In addition, before implementation of the project began the student prepared UML diagrams to visually demonstrate the flow of the application and to identify which classes and frameworks; the student would need to investigate and use to implement the project.

Moreover, the student has detailed in this document the software concepts and techniques used to implement the project, by explaining the data structure, sort and search algorithms used and how they were implemented into the project. Furthermore, the students adopted testing approach has been documented and test results have been used to help reflect on the implementation of the project. Finally, the student has reflected on the overall implementation and performance of the project delivered to draw up conclusions about the success of the implementation.

# 3 Detailed Requirements Specification

In this section of the project document the student has identified the functional and non-functional requirements of the project software and using MoSCoW Analysis, the student has prioritised these requirements into 4 categorises: Must, Should, Could and Won’t. The following table explains what each of these categorises means.

|  |  |
| --- | --- |
| Category | Explanation |
| Must | Mandatory requirement which need to be part of the software |
| Should | Requirements that are not the most important to the software but add significant value if added to the software |
| Could | Requirements that are nice to have but don’t offer the greatest impact to the software |
| Won’t | Requirements that are not urgent priority and can be added later in future versions of the software. (Productplan.com. 2020) (YouTube. 2020). |

## 3.1 MoSCoW Analysis

### 3.1.1 Must

* Identify and implement the data structure to organize the data being used in the software project
* Identify and implement the searching algorithm to check or retrieve elements from any data structure stored in the football team table
* Identify and implement the searching algorithm to check or retrieve elements from any data structure stored in the football player table
* Identify and implement the sorting algorithm to arrange the football team data stored according to sorting choice selected
* Identify and implement the sorting algorithm to arrange the football player data stored according to sorting choice selected
* Implement a table to store the data containing the information about the team and identify what data to capture
* Implement a table to store the data containing the information about the player and identify what data to capture

### 3.1.2 Should

* Implement the ability for the user to add a team to the Team table
* Implement the ability for the user to add a player to the Player table
* Implement the ability for the user to delete a team and remove from the data store
* Implement the ability for the user to delete a player and remove from the data store
* Implement the ability for the user to Edit a team to update the data store
* Implement the ability for the user to Edit a player to update the data store
* Implement Registration and Login functionality to add security to the application and to protect user data (mock data is being used for this implementation, however, if this was a real application available to the general public the app would contain real data)
* Implement a Logout button to allow users to logout of the app

### 3.1.3 Could

* Implement the ability for users to view game fixtures
* Implement the ability for the data stored in the tables to automatically update every time a user updates the data on all other user’s devices. Implement a table that is not stored on the device but on a cloud table
* Implement login and authentication so users can only access leagues and teams they play for
* Implement the ability for the user to add a league and add to the data store
* Implement the ability for the user to delete a league and remove from the data store
* Implement the ability for the user to Edit a league to update the data store

### 3.1.4 Won’t

* Implement the ability for the team to write a biography about the team
* Implement the ability for the player to write a biography about the team
* Implement the ability for the team to add team news and updates to fans
* Implement the ability for push notifications to be sent to users after the data has been updated

## 3. 2 User Stories

### 3.2.1 Must User Stories

**User Story 1**: As a user I want to be able to view a table containing the information about the football teams

GIVEN the user has the app install on their device

AND the user has launched the app on the homepage

WHEN the user selects the button labelled ‘Teams’

THEN the user will be presented with the football team table with the fields: League Position, Team Name, Wins, Draws, Losses, Games played, Points

**User Story 2**: As a user I want to be able to view a table containing the information about the football players

GIVEN the user has the app install on their device

AND the user has launched the app on the homepage

WHEN the user selects the button labelled ‘Players’

THEN the user will be presented with the football player table with the fields: Player Name, Position, Appearances, Yellow Cards, Red Cards, Assists, Goals

**User Story 3**: As a user I want to be able to search the team table to find a specific team quickly

GIVEN the user is on the football team page

THEN the user will be presented with a ‘search’ textbox at the top of the page

AND the user can enter a team name in the ‘search’ textbox

AND the user can click the ‘search’ button

THEN the user will be presented with the result of the search on the page

**User Story 4**:As a user I want to be able to search the player table to find a specific team quickly

GIVEN the user is on the football player page

THEN the user will be presented with a ‘search’ textbox at the top of the page

AND the user can enter a player name in the ‘search’ textbox

AND the user can click the ‘search’ button

THEN the user will be presented with the result of the search on the page

**User Story 5**: As a user I want to be able to sort the football team table to find a specific statistic quickly and easily

GIVEN the user is on the football team page

THEN the user will be presented with a ‘sort’ dropdown box for sorting the football teams

AND the user will be presented with the default sort of ‘Most Points’ on navigating to the teams page

THEN the user can select the ‘sort’ options: Most Points, Ascending (A-Z), Descending (Z-A)

AND the user will be presented with the table view sorted by the users selected option

**User Story 6**: As a user I want to be able to sort the football player table to find a specific statistic quickly and easily

GIVEN the user is on the football player page

THEN the user will be presented with a ‘sort’ dropdown box for sorting the football players

AND the user will be presented with the default sort of ‘Top Goal Scorer’ on navigating to the players page

THEN the user can select the ‘sort’ options: Top Goal Scorer, Most Appearances, Ascending (A-Z), Descending (Z-A)

AND the user will be presented with the table view sorted by the users selected option

### 3.2.2 Should User Stories

**User Story 1**: As a user I want to be able to add a team to the Team table

GIVEN the user is on the football team page

WHEN the user clicks the table

THEN the user will be presented with the option to add a team

AND the user enters a new Team

THEN the user’s new team is added to the Team table

**User Story 2**: As a user I want to be able to add a player to the Player table

GIVEN the user is on the football player page

WHEN the user clicks the table

THEN the user will be presented with the option to add a player

AND the user enters a new Player

THEN the user’s new player is added to the Player table

**User Story 3**: As a user I want to be able to delete a team from the Team table

GIVEN the user is on the football team page

WHEN the user clicks the table

THEN the user will be presented with the option to delete a team

AND the user selects a Team

THEN the team is removed from the table

**User Story 4**: As a user I want to be able to delete a player from the Player table

GIVEN the user is on the football player page

WHEN the user clicks the table

THEN the user will be presented with the option to delete a player

AND the user selects a player

THEN the player is removed from the table

**User Story 5**: As a user I want to be able to edit a team from the Team table

GIVEN the user is on the football team page

WHEN the user clicks the table

THEN the user will be presented with the option to edit a team

AND the user selects a Team

THEN the team is edited on the table

**User Story 6**: As a user I want to be able to edit a player from the Player table

GIVEN the user is on the football player page

WHEN the user clicks the table

THEN the user will be presented with the option to edit a player

AND the user selects a Player

THEN the player is edited on the table

**User Story 7:** As a user I want to be able to register to use the Android app

GIVEN the user is not registered

WHEN the user launches the app, the user will see the Login and Register options

THEN the user will click Register

AND the user is presented with the Register page

THEN the user enters a username and password

AND clicks Register

THEN the user is registered to use the app

**User 8:** As a user I want to be able to Login to the app

GIVEN the user is already registered to use the app

AND the user is on the Login page

WHEN the user enters the correct username and password combination

AND the user clicks Login

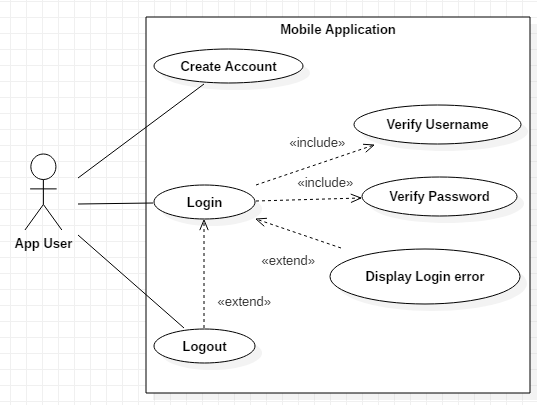
THEN the user is presented with the homepage of the app

*Please Note that the student has only written user stories for the ‘Must’ and ‘Should’ requirements because these are all the requirements the student was likely to be able to achieve within the timeframe of the assignment.*

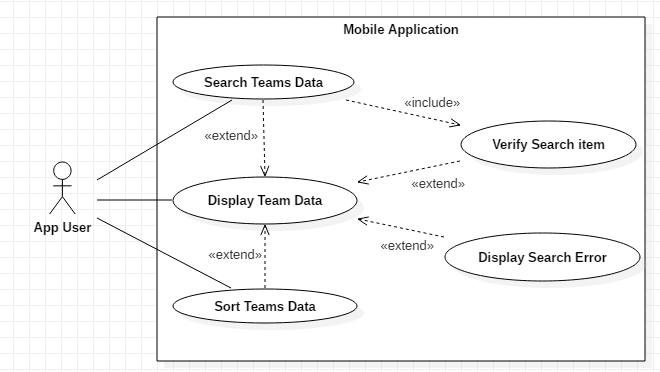
# 4 UML Use Case Diagrams

The UML Case diagrams are used to depict how the user interacts with the Android mobile application and how the system functionality interacts with each other.

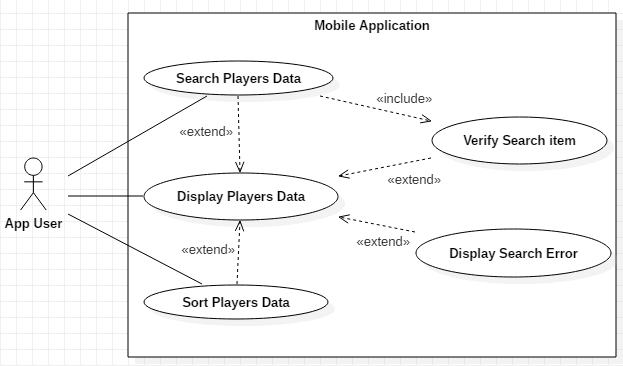
## 4.1.1 Login and Registration Use Cases



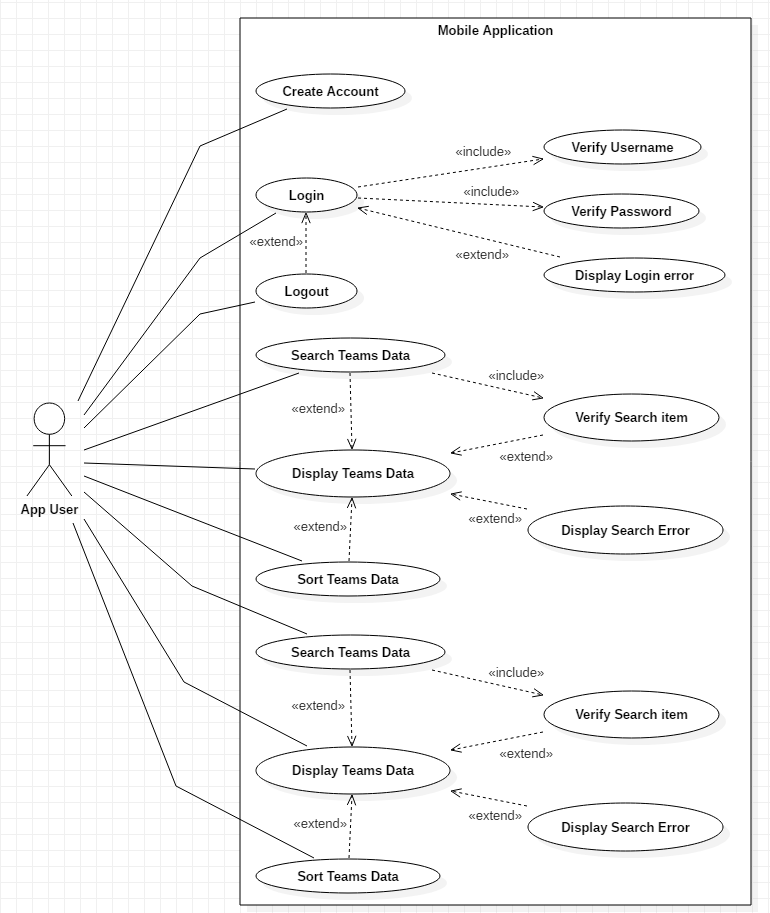
## 4.1.2 Search and Sort Teams Use Cases



## 4.1.3 Search and Sort Player Use Cases



## 4.1.4 Mobile Application Use Cases

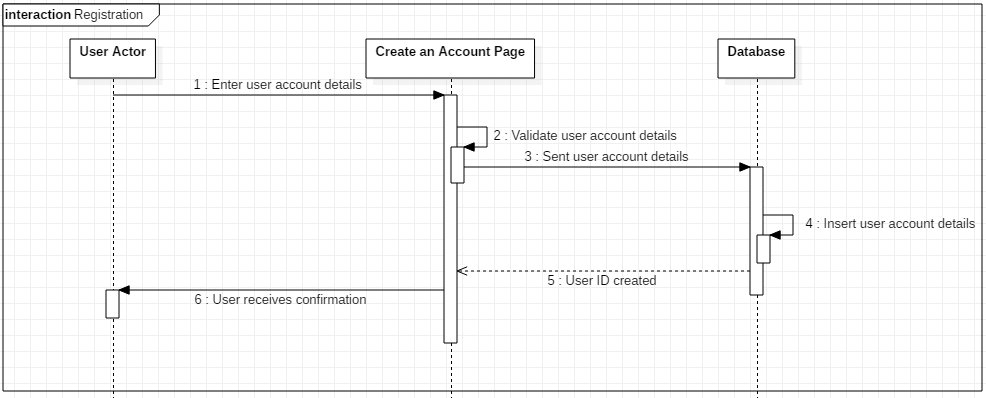


# 5 UML Activity/Sequence Diagrams

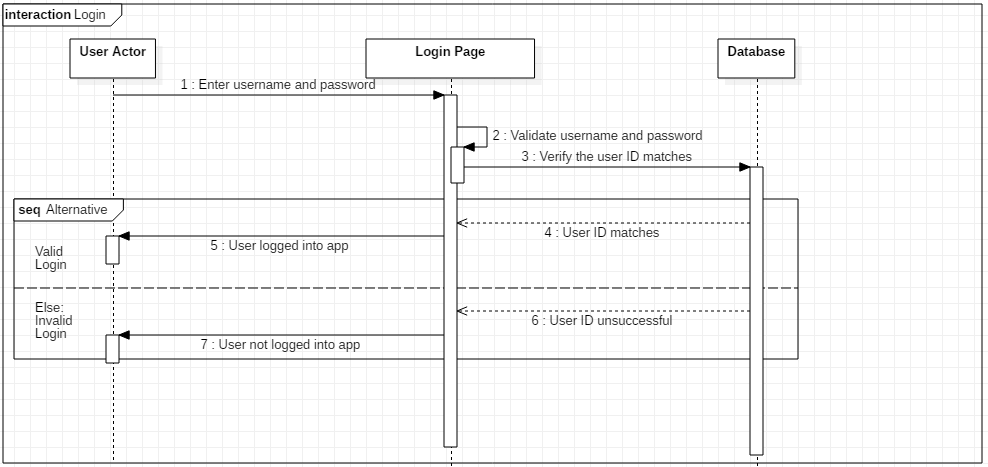
The process views will show the non-functional requirements of the mobile application, these diagrams and views will display the process of grouping tasks which will form the executable unit for the features contained within the mobile application. Furthermore, the process view diagrams will show the order of the component interactions and how the components interact with each other.

The following interaction diagrams will display how the components of the feature of the app interact with each other and in which order.

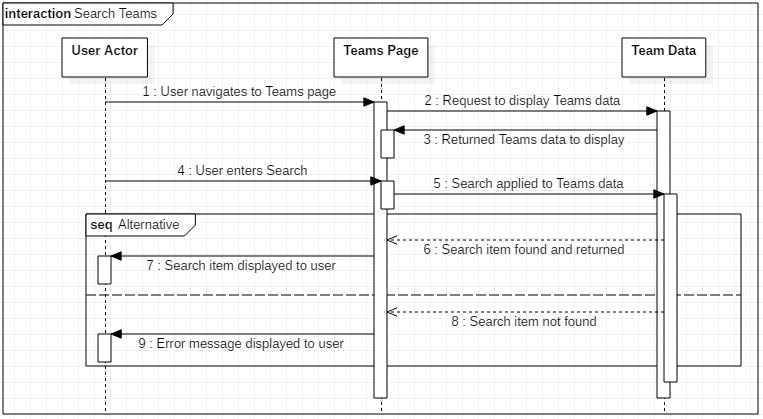
## 5.1.1 Activity/Sequence Diagram: Create a New Account



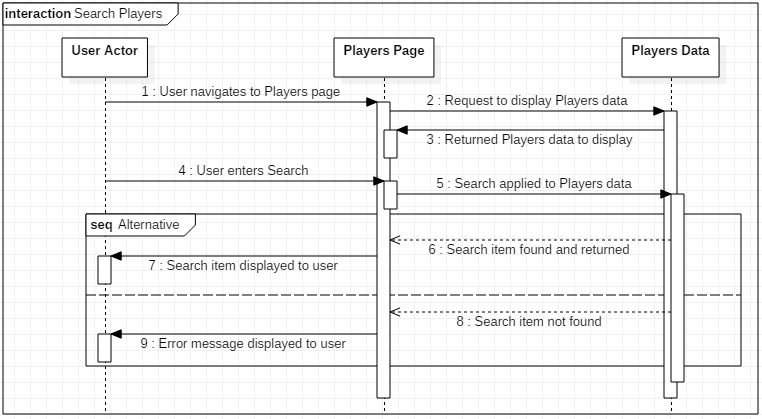
## 5.1.2 Activity/Sequence Diagram: Login



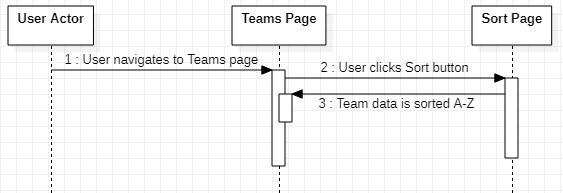
## 5.1.3 Activity/Sequence Diagram: Search Teams



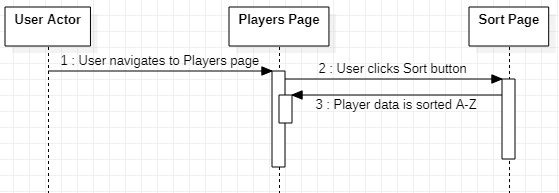
## 5.1.4 Activity/Sequence Diagram: Search Players



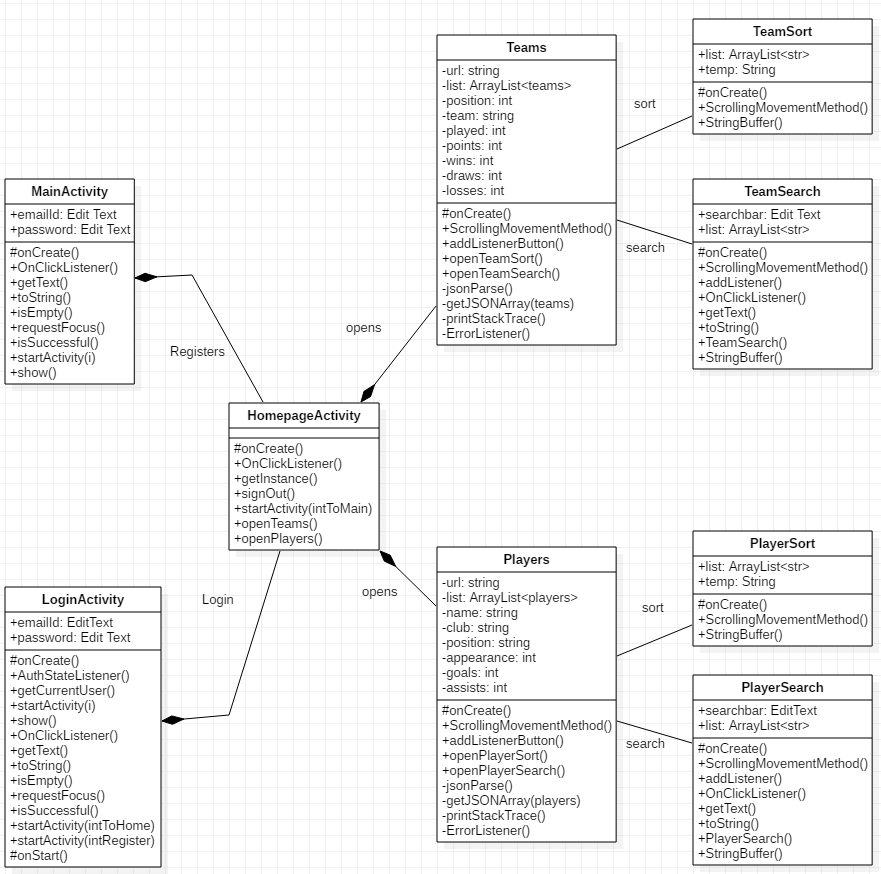
## 5.1.5 Activity/Sequence Diagram: Sort Teams



## 5.1.6 Activity/Sequence Diagram: Sort Players



# 6 Class Diagram



# 7 Interface Designs for the Final Android Mobile Application

## 7.1 Registration Page

The Registration page is the first page of the app displayed to the user when the app is launched.

This page has a header “Football Data Store” which is the name of the Android mobile application.

This page also shows the app icon

Below are the “Email” and “password” fields which are used for authentication and registration

The email field detects for invalid and valid email address format

The password field detects the user input

The ‘CREATE AN ACCOUNT’ button passes the entered email and password fields to firebase for authentications and adds the user if not already registered and if the account is already registered the user is asked to ‘Login’

Clicking the text ‘Already Registered? Sign in here’ takes the user to the Login Page

# 

This screenshot shows the presented error message if the user attempts to register with invalid or no email address

# 

If the user enters already registered user details or invalid details, then the user remains on the Registration page and the user is presented with the toast error message “Register unsuccessful. Please try again”

If the user enters valid registration details, then the user is presented with the Landing page of the app and the user’s details are stored in Firebase.

This screenshot shows the presented error message if the user attempts to register with invalid or no password

## 7.2 Login Page

This page has a header “Football Data Store” which is the name of the Android mobile application.

This page also shows the app icon

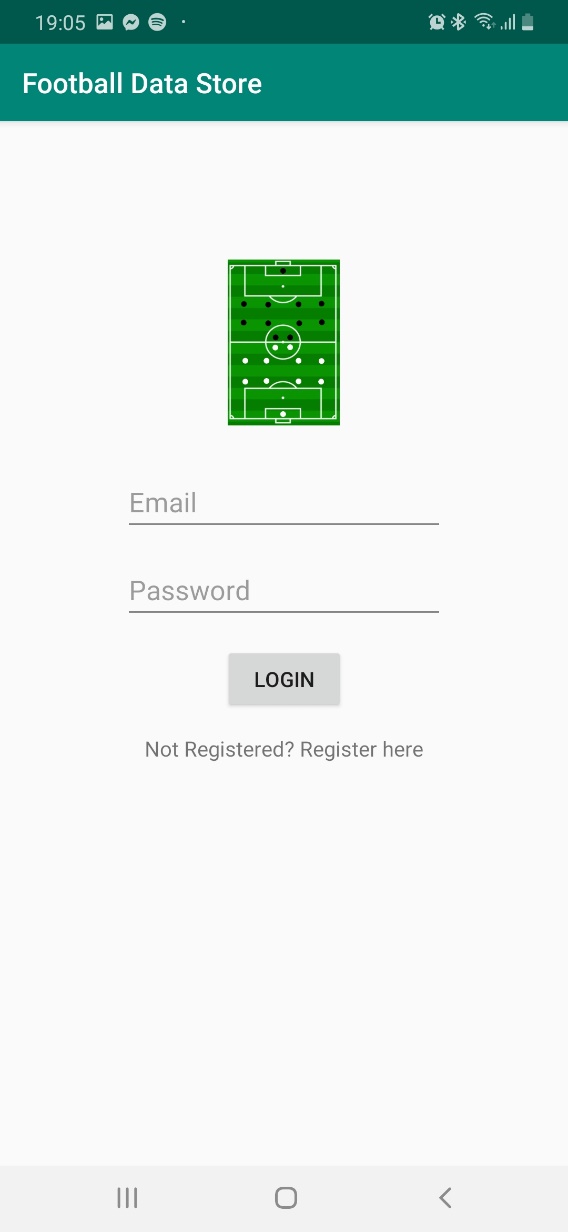
Below are the “Email” and “password” fields which are used for authentication and login

The email field detects for invalid and valid email address format

The password field detects the user input

The ‘LOGIN’ button passes the entered email and password fields to firebase for authentications and if the account is already registered then the user is logged into the app

Clicking the text ‘Not Registered? Register here’ takes the user to the Landing Page



# 

This screenshot shows the presented error message if the user attempts to login with invalid or no email address

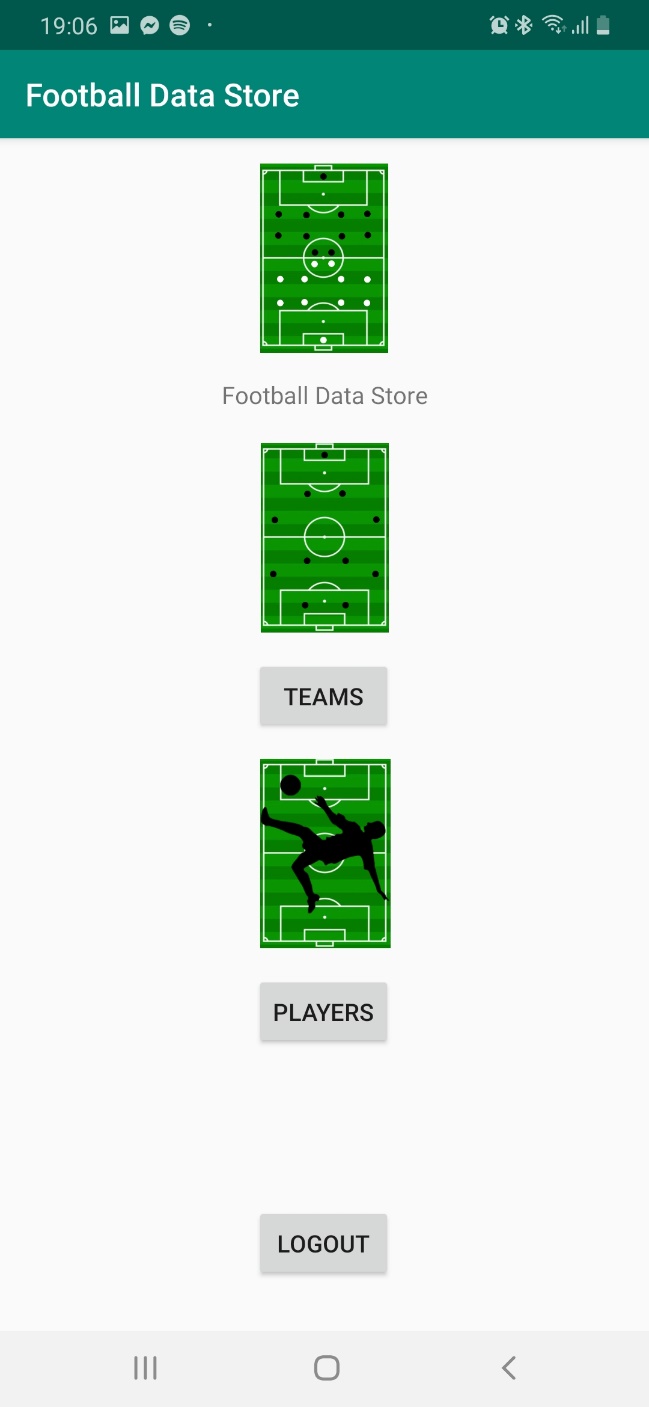
# 

This screenshot shows the presented error message if the user attempts to login with invalid or no password

# 

When the user login is successful then the user is presented with the Landing page of the app and the toast message ‘You are logged in’ is presented

## 7.3 Landing Page



The Landing page of the app is the homepage and provides the navigation for the user, so they can navigate to ‘Teams’, ‘Players’ or ‘Logout’ of the app

This page has a header at the top of the page which is the name of the app

Under the header the app icon and name of the app ‘Football Data Store’ is displayed

Then the user is presented with the ‘Teams’ icon and ‘Teams’ button which navigates the user to the Teams page.

Then the user is presented with the ‘Players’ icon and ‘Players’ button which navigates the user to the Players page.

Finally, at the bottom of the page the user is presented with the ‘Logout’ button, which logs the user out of the app and takes the user back to the Registration page

## 7.4 Teams Page



Clicking the ‘GET TEAM DATA’ button will fetch and display the team’s data from the json file on the page.

# 

This is the first page of the ‘Teams’ page and at the top of the page the user is presented with the ‘Teams’ icon and heading

At the bottom of the page the user is presented with 3 buttons

Sort button takes the user to the ‘Teams Sort’ page

Search button takes the user to the ‘Teams Search’ page

### 7.4.1 Team Sort Page

At the top of the page the user is presented with the header

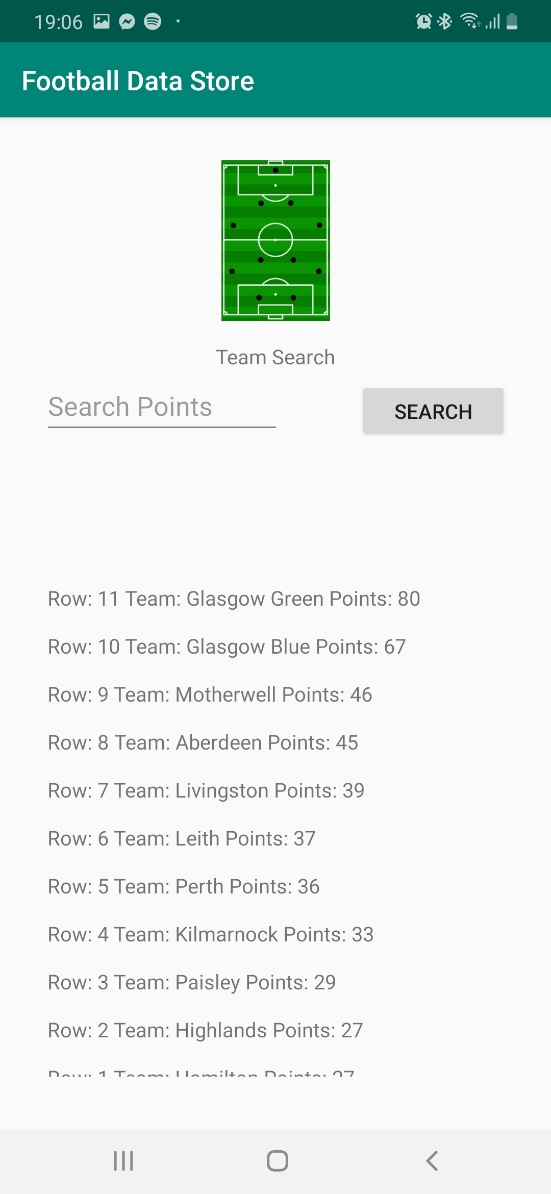
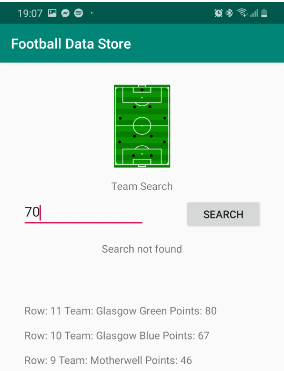
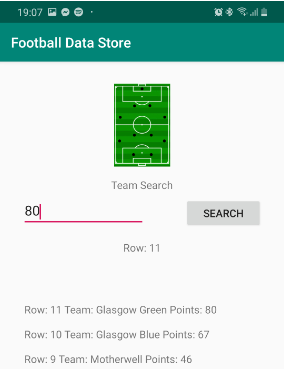
Then the user is presented with the Teams icon and heading for the sorting page: Teams Sorted A-Z

Finally, the user is presented with the list of teams in ascending order from A – Z in a scrollable text view

To navigate back to the previous page the user is presented with the native back button on android devices

# 

### 7.4.2 Team Search Page



These two images show what the user is presented with when the enters a successful and unsuccessful search input.

At the top of the page the user is presented with the header

Then the user is presented with the Teams icon and the heading for the searching page: Team Search. On this page the user can only search for ‘Points’

Next the user is presented with the search edit text field where the user can enter a search input, and on the same row the user is presented with the ‘Search’ button. Clicking the ‘Search’ button will start the search and below in the next row a message will be presented to tell the user which row the search input was found or if the search was not found.

Finally, in a scrollable text view the user is presented with rows of teams and the ‘Points’ value for each team, which is what the user is searching.

## 7.5 Players Page

## 

Clicking the ‘GET PLAYER DATA’ button will fetch and display the player’s data from the json file on the page.

This is the first page of the ‘Players’ page and at the top of the page the user is presented with the ‘Players’ icon and heading

At the bottom of the page the user is presented with 3 buttons

Sort button takes the user to the ‘Players Sort’ page

Search button takes the user to the ‘Players Search’ page

### 7.5.1 Players Sort

At the top of the page the user is presented with the header

Then the user is presented with the Players icon and heading for the sorting page: Players Sorted A-Z

Finally, the user is presented with the list of Players in ascending order from A – Z in a scrollable text view

To navigate back to the previous page the user is presented with the native back button on android devices

# 

### 7.5.2 Players Search

At the top of the page the user is presented with the header

Then the user is presented with the Players icon and the heading for the searching page: Player Search. On this page the user can only search for ‘Goals’

Next the user is presented with the search edit text field where the user can enter a search input, and on the same row the user is presented with the ‘Search’ button. Clicking the ‘Search’ button will start the search and below in the next row a message will be presented to tell the user which row the search input was found or if the search was not found.

Finally, in a scrollable text view the user is presented with rows of players and the ‘Goals’ value for each player, which is what the user is searching.

# 

These two images show what the user is presented with when the enters a successful and unsuccessful search input.

# 8 Software Implementation

To implement the project to build an Android Mobile Application to allow non-professional football teams to capture their league, teams and players performance and statistics throughout a football season. The student used the application Android Studio which is powered by the IDE IntelliJ and the student wrote the code in the object-oriented language Java. Finally, to implement the project the student used a data structure and sorting and searching algorithms.

IDE: Android Studio (which is built on JetBrains IntelliJ)

Programming Language: Java

Data Structure: Array

Searching Algorithms: Binary Search Trees

Sorting Algorithms: Bubble sort

In addition, to data structures and search and sort algorithms the student used Firebase for authentication. Firebase authentication was used for the registration and login functionalities of the mobile application. In this section of the document, the student will explain in detail the concepts, frameworks, tools, and libraries used in the delivered software and how they were implemented into the Android mobile application.

## 8.1 Firebase Authentication

To allow users to register to use the app and to login to the app, the student implemented Firebase Authentication. As in future versions of the application users will be able to add, edit and delete data, so to ensure data is protected in the future, the student implemented Firebase Authentication into the first version of the app and as part of the minimum viable product (MVP).

Firebase Authentication is used to provide the backend service for authenticating the app and the Firebase SDK was implemented into the app to help enable this service. In the app the user is authenticated by email address and password. (Firebase. 2020)

To implement Firebase Authentication the student created and added their Firebase project to the app code and the student added the dependency for Firebase Authentication Android library to the build.gradle file: implementation **'com.google.firebase:firebase-auth:19.3.0'**

### 8.1.1 Register

To register a new account the student implemented the following code in the ‘MainActivity’ class. Firstly, the student gets the shared instance of the FirebaseAuth object in the ‘onCreate’ method.

**mFirebaseAuth** = FirebaseAuth.*getInstance*();

Secondly, the student registers the new user by passing the new users valid email address and password to ‘createUserWithEmailAndPassword’.

**else** {  
 **mFirebaseAuth**.createUserWithEmailAndPassword(email,pwd).addOnCompleteListener(MainActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
 *//In the above line of code the users entered email and password are captured and passed to Firebase to be authenticated* @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if**(!task.isSuccessful()){  
 Toast.*makeText*(MainActivity.**this**, **"Register unsuccessful, Please try again"**,Toast.***LENGTH\_SHORT***).show();  
 *//If AuthResult is not successful the user is presented with a toast pop-up error message* }  
 **else** {  
 startActivity(**new** Intent(MainActivity.**this**,HomepageActivity.**class**));  
 *//If the AuthResult is successful then the user is taken to the HomepageActivity* }  
 }  
 });

### 8.1.2 Login

To login an existing user the student implemented the following code in the ‘LoginActivity’ class. Firstly, the student gets the shared instance of the FirebaseAuth object in the ‘onCreate’ method.

**mFirebaseAuth** = FirebaseAuth.*getInstance*();

Secondly, the student checks if the user is currently logged into the app or not.

**protected void** onStart(){  
 **super**.onStart();  
 **mFirebaseAuth**.addAuthStateListener(**mAuthStateListener**);

Next, when the user enters a valid email address and password combination and clicks login, the user email and password is pass to Firebase using ‘signInWithEmailAndPassword’.

**mFirebaseAuth**.signInWithEmailAndPassword(email, pwd).addOnCompleteListener(LoginActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
 *//In the above line of code the users entered email and password are captured and passed to Firebase to be authenticated* @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if**(!task.isSuccessful()){  
 Toast.*makeText*(LoginActivity.**this**, **"Login Error, Please Login Again"**,Toast.***LENGTH\_SHORT***).show();  
 *//If AuthResult is not successful the user is presented with a toast pop-up error message* }  
 **else**{  
 Intent intToHome = **new** Intent(LoginActivity.**this**,HomepageActivity.**class**);  
 startActivity(intToHome);  
 *//If the AuthResult is successful then the user is taken to the HomepageActivity* }  
 }  
});

Finally, the user is successfully logged into the app and the ‘FirebaseUser’ is returned. (Firebase. 2020)

## 8.2 Data Structure

For the implementation of the project the student proposed the Tree data structure. The Tree is a non-linear data structure and is also an undirected graph, which has no cycles and is connected. One of the main reasons for initially selecting the Tree data structure in the proposal stage was efficiency, as there are no cycles which means the chances of performing unnecessary work, caused by going around in cycles is avoided.

Tree is a data structure used for data storage purposes and this was ideal for the Students project of storing football team and player statistics and allowing users to query the data by searching and sorting. Also, the Tree data structure is hierarchical, and each node can only have a maximum of 2 children.

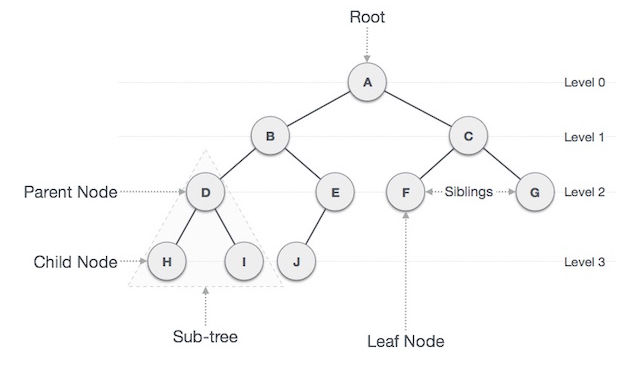


Figure 3: Structure of Tree data structures. Source: Tutorialspoint.com. (2020).

The below table explains which all the terms shown on the diagram above mean.

|  |  |
| --- | --- |
| Term | Explanation |
| Root | Point of reference for the whole table and the root has no parent |
| Parent Node | Every node apart from the root has one edge upward to a node called parent |
| Child Node | The node below a given node connected by its edge downward is its child node. |
| Siblings | Nodes with the same parent |
| Leaf Node | Node with no child |
| Edge | Edge is a connection between one node to another. (Tutorialride.com. 2020). |

## 8.3 Data Structure Implementation

For the implementation of the Android app the student adopted the data structure of Arrays. The main reason for the change of data structure during implementation was the student was struggling to implement the Tree data structure and the Array data structure was working a lot better on the mobile application and was more straight forward to implement into the project.

Arrays are a collection of items or values stored in a memory location. The data stored in the array are all the same data type and the array contains multiple items or value. (GeeksforGeeks. 2020)

In the implementation of the app the student has used arrays in the following classes: Players, PlayerSearch, PlayerSort, Teams, TeamSort and TeamSearch.

Examples of Arrays used in the implementation of the app:

Example of string array used in the ‘TeamSort’ class.

String **str** [] = {**"Glasgow Green"**,**"Glasgow Blue"**,**"Motherwell"**,**"Aberdeen"**,**"Livingston"**,**"Leith"**,**"Perth"**,**"Kilmarnock"**,**"Paisley"**,**"Highlands"**,**"Hamilton"**,**"Edinburgh"**};

Example of integer array used in the ‘TeamSearch’ class.

**int** intArr[]= { 80, 67, 46, 45, 39, 37, 36, 33, 29, 27, 27, 23 };

### 8.3.1 Implementation of ArrayList

For the ‘Teams’ and ‘Players’ data the student used the Volley library to parse a JSON Object containing the JSON array from a URL into the TextView on the pages of the apps. The JSON array’s can be found at the following links provided:

Teams Data: <https://raw.githubusercontent.com/grantmartin161096/FootballDataStore/master/TeamData.json>

Player Data: <https://raw.githubusercontent.com/grantmartin161096/FootballDataStore/master/PlayerData.json>

Firstly, to retrieve the JSON array from the internet the student added the following permission into the ‘AndroidManifext.xml’ file to allow to app to communicate with the internet:

<**uses-permission android:name="android.permission.INTERNET"** />

Next, the student imported the Volley HTTP library into the project to perform the network requests to retrieve the data in the ‘build.gradle’ file.

implementation **'com.android.volley:volley:1.1.0'**

The reason why the student decided to use volley was because the requests are added to a single queue which will result in automatic scheduling of request and the user can call volley from the UI thread and Volley handles all the backend network calls.

Next, the student creates the ‘RequestQueue’ variable which sends the request to the network and the ‘Request’ contains all the necessary information for making the web API call is stored within it. (GeeksforGeeks. 2020)

Please find below the code with comments from retrieving the arraylist with the ‘Teams’ and ‘Players’ data using the volley library:

**mQueue** = Volley.*newRequestQueue*(**this**);  
 *//Volley library is implemented in the build.gradle file  
 //RequestQueue makes the request to the network* buttonParse.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 jsonParse();  
 }  
 });  
 *//The jsonParse method is called when the get data button is pressed* }  
  
 **private void** jsonParse() {  
 *//This method gets the json file from the internet using the string url below and the internet permission set-up in the AndroidManifest.xml file* String url = **"https://raw.githubusercontent.com/grantmartin161096/FootballDataStore/master/PlayerData.json"**;  
 JsonObjectRequest request = **new** JsonObjectRequest(Request.Method.***GET***, url, **null**,  
 **new** Response.Listener<JSONObject>() {  
 @Override  
 **public void** onResponse(JSONObject response) {  
 **try** {  
 JSONArray jsonArray = response.getJSONArray(**"players"**);  
 *//Player is the name of the object  
  
 //Below the Player array fields are called to get the data types for the following fields* **for** (**int** i = 0; i < jsonArray.length(); i++) {  
 JSONObject players = jsonArray.getJSONObject(i);  
  
 String name = players.getString(**"Name"**);  
 String club = players.getString(**"Club"**);  
 String position = players.getString(**"Position"**);  
 **int** appearances = players.getInt(**"Appearances"**);  
 **int** goals = players.getInt(**"Goals"**);  
 **int** assists = players.getInt(**"Assists"**);  
  
 *//The data retrieved from the json file are captured and displayed in the TextView* **mTextViewResult**.append(**"Name: "** + name + **" Club: "** + club + **" Position: "** + position + **" Appearances: "** + String.*valueOf*(appearances) + **" Goals: "** + String.*valueOf*(goals) + **" Assists: "** + String.*valueOf*(assists) + **"\n\n"**);  
 }  
 *//catching exceptions* } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }, **new** Response.ErrorListener() {  
 @Override  
 **public void** onErrorResponse(VolleyError error) {  
 error.printStackTrace();  
 }  
 });  
 *//volley library makes request  
 //request sends up the information to GET and Post* **mQueue**.add(request);

## 8.4 Sort Algorithms

For the implementation of the project the student investigated the following sorting algorithms, before selecting the Quick sort algorithm in the proposal:

* Bubble Sort (quadratic algorithm)
* Selection Sort (quadratic algorithm)
* Insertion Sort (quadratic algorithm)
* Shell Sort (quadratic algorithm)
* Merge Sort (Linear-logarithmic algorithm)
* Quick Sort (Linear-logarithmic algorithm)

The main reason the student selected the Quick sort algorithm was because it is a Linear-logarithmic algorithm and has a significantly quicker computational speed than quadratic algorithms when computing large amounts of data. Furthermore, it is one of the most popular and common sorting algorithms used. The image below shows the comparison of the different sorting algorithms.

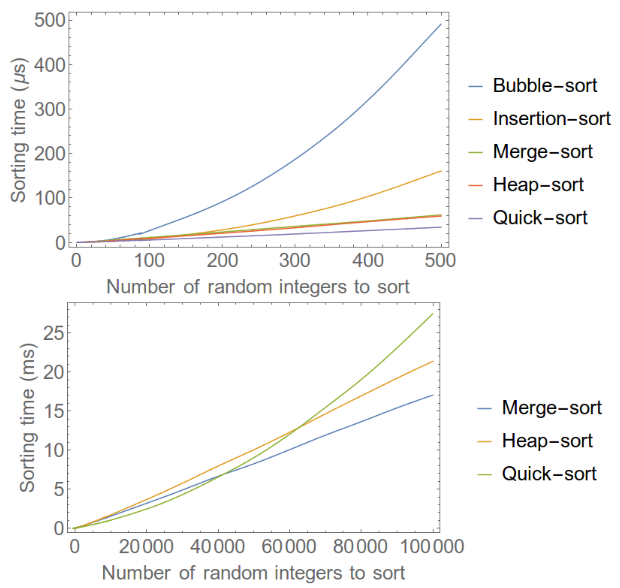


Figure 3: shows the sort speed different sorting algorithms. Source: C++, S., B., J. and York, M. (2020).

However, the Quick sort algorithm does have some disadvantages, for example, it is more difficult to understand and implement into the programme compared to the Bubble sort algorithm and other quadratic algorithms. In addition, speed of computational is not as significant when only sorting small amounts of data.

### 8.4.1 How the Quick sort works?

Quick sort is a divide-and-conquer method for sorting. It works by partitioning the list into 2 sub-lists based on a pivot element. In one list the elements are ordered to be smaller than the pivot element and in the other list the elements are ordered to be larger than the pivot element. The recursive arranging of the two list continues until the list is ordered. The image below shows how the Quick sort works in practice. (Tutorialspoint.com. 2020). (Sciencing. 2020).



Figure 4: Shows how the Quick sort works. Source: GeeksforGeeks. (2020).

## 8.5 Bubble Sort Implementation

During Implementation of the Quick sort algorithm the student was struggling to implement the more complex Quick sort algorithm and as a result the student decided to implement the bubble sort algorithm because it is a simple sorting algorithm, also for this Android application the amount of data being sorted is not massive, so the performance or speed of the app was not greatly impacted by implementing the slower quadratic algorithm.

The Bubble sort is a comparison-based algorithm were a pair of adjacent elements in the array are compared and the elements are swapped if the elements are not in order. The Bubble sort starts with the first two elements in the array and the algorithm checks which is the highest element and the highest element is then compared with the next element in the array until the array is successfully sorted. (Tutorialspoints.com. 2020)

Below the Bubble sort is executed to the Teams and the Players data

*//Below the bubble sort is applied***for** (**int** i = 0; i < **str**.**length**; i++) {  
 *//The compareTo method compare the strings in the array to be sorted  
 //The string array is sorted from A-Z  
 //If the list is not in order the code is swapping values with a temp (temporary) string value* **for** (**int** j = i + 1; j < **str**.**length**; j++) {  
 **if** (**str**[i].compareTo(**str**[j])>0) {  
 **temp** = **str**[i];  
 **str**[i] = **str**[j];  
 **str**[j] = **temp**;  
 }  
 }  
}

## 8.6 Search Algorithms

For the implementation of the project the student investigated the following searching algorithms, before selecting the Binary Search algorithm to use in the software:

* Graph Traversal Algorithms
* Graph Adjacency Matrix/List Algorithms
* Binary Search Algorithms

The main reason the student selected Binary Search algorithms is because they are efficient and it gets a hint at each step, about which sub-tree has the searched element. Furthermore, the data structure being used for this project is binary trees, so it is logical to adopt binary search algorithms to execute searching in this project. (www.javatpoint.com. 2020).

### 7.6.1 How Binary Search Algorithms work?

Binary Search Algorithm is a divide-and-conquer algorithm, which assumes that the array to be searched has already been sorted. In a binary search tree, the left sub-tree of a node has a key less than or equal to the parent node key and the right sub-tree of a node has a key greater than the parent node tree. The nodes have a key and value connected to each of them; when the search is taking place, the desired key is compared against the keys in the binary search tree and if the desired key is discovered, it is retrieved.

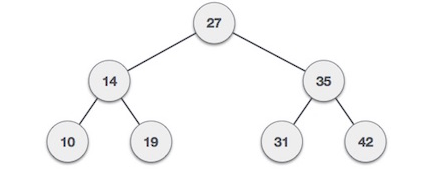


Figure 5: shows a binary search tree. Source: Tutorialspoint.com. (2020).

In figure 5 above the root node key (27) has the nodes with key values less than 27 in a sub-tree to the left and has nodes with key values more than (27) in a sub-tree to the right.

## 8.7 Binary Search Algorithm Implementation

For the implementation of the Binary search the student implemented the java.util.Arrays.binarySearch(int[] a, int key) method which searches the array for the users inputted search value using the binary search algorithm. Before the search is executed the array is sorted in order to prevent the result coming back as undefined. Below the binary search code is shown with comments on what the code is doing:

**buttonSearch**.setOnClickListener( **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 String value1 = **searchbar**.getText().toString();  
 *//The users search input is captured and used to search the intArr* TeamSearch ob = **new** TeamSearch();  
 *// initializing int array* **int** intArr[]= { 80, 67, 46, 45, 39, 37, 36, 33, 29, 27, 27, 23 };  
 *// sorts the int array* Arrays.*sort*(intArr);  
  
 *// let us print all the elements available in list* System.***out***.println(**"The sorted int array is:"**);  
 **for** (**int** number : intArr) {  
 System.***out***.println(**"Number = "** + number);  
 }  
  
 *// Captures the users entered value to be searched* **int** x = Integer.*parseInt*(value1);  
  
 **int** result = Arrays.*binarySearch*(intArr,x);  
 *//below the result of the search is handled for successful and unsuccessful searches* **if** (result == -13)  
 **text**.setText(**"Search not found"**);  
 **else  
 text**.setText(**text**.getText() + **"Row: "** + result +**"\n\n"**);  
  
 }  
  
});

After the binary search is executed the search key is returned and if the search input is found within the array then an entry is returned in the result edit text view, however, if the search input is not found a negative value is returned where the search input would have been positioned in the array.

(Tutorialspoints.com. 2020)

# 9 Adopted Software Testing Process

For this project, a combination of Black-box and White-box testing was adopted for the testing of the project. Black-box testing is a high-level form of testing that focused testing on the behaviour of the mobile app, and testing was carried out from an end user point of view. This is a less exhaustive form of testing and was performed as part of user acceptance testing when the software had completed unit and integration testing. However, for this project the Developer also executed White-box testing to probably test the algorithms, Black-box testing is not good for testing algorithms. White-box testing is a form of testing that focuses on the internal code of the software and was required to be carried out by the Developer. (Guru99.com. 2020).

## 9.2 Test Plan and Test Results

Test execution for the Black-box testing was carried out by the student using the Pixel 2 emulator on Android Studio and by an end user using a physical Android device, Samsung A50.

To test the application the student prepared 48 test cases and scripts based on the designs of the mobile app, the 48 test scripts were manually run against the physical device and the emulator. The test scripts and test results for the tests run against the Samsung A50 can be viewed in the appendix of this document. In total 96 tests were executed with 92 tests passing and 4 tests failing, the test cases which failed were numbers: 37 and 48. Both of these tests failed as a result of the same defect, in the Player and Team search page when the user enters a search value which is a value between two valid search inputs, the app returns the row where this search input would be positioned in the array with the negative symbol in front of the row number, instead of the error message ‘search not found’.

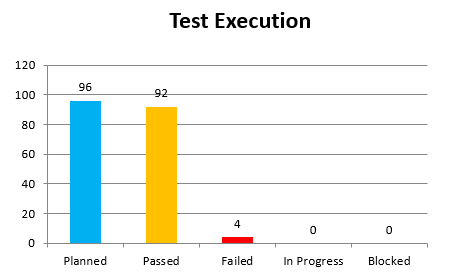


Figure 6: Shows the number of tests executed and the results

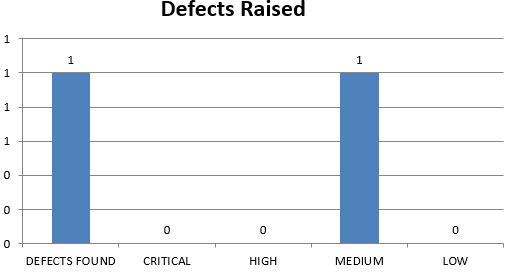


Figure 7: Shows the number of defects found during testing and the priority assigned

Note: Test Cases can be found in appendix 12.4.1 and the test results can be found in appendix 12.4.2

### 9.2.1 White-box Testing

The student planned to use Junit to unit test the sorting and searching algorithms to White-box test this functionality of the app, because it is difficult to properly test algorithms using Black-box testing. White-box testing was intended to look at the internal code of the software, however, during implementation of the unit tests the student encounter issues. The main error the student continually faced was:

|  |  |
| --- | --- |
| “*error: no suitable method found for onCreate(String[]) method Activity.onCreate(Bundle) is not applicable (argument mismatch; String[] cannot be converted to Bundle) method androidx.core.app.ComponentActivity.onCreate(Bundle) is not applicable (argument mismatch; String[] cannot be converted to Bundle) method androidx.activity.ComponentActivity.onCreate(Bundle) is not applicable (argument mismatch; String[] cannot be converted to Bundle) method FragmentActivity.onCreate(Bundle) is not applicable (argument mismatch; String[] cannot be converted to Bundle) method AppCompatActivity.onCreate(Bundle) is not applicable (argument mismatch; String[] cannot be converted to Bundle) method TeamSort.onCreate(Bundle) is not applicable (argument mismatch; String[] cannot be converted to Bundle)”* |  |

This error was caused because the sorting and searching algorithms are within the ‘onCreate’ method in their respected classes. Furthermore, as the unit tests were performed at the end of the project the student did not have time to fix the unit tests or make changes to the software code.

Please find below screenshots of the errors experienced:

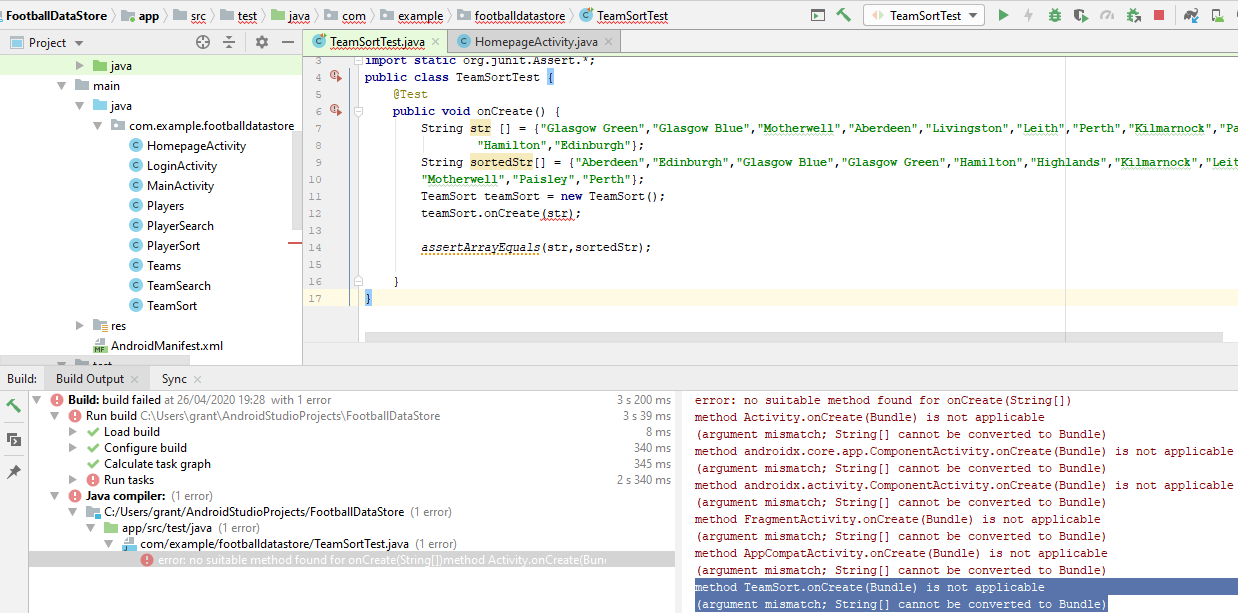
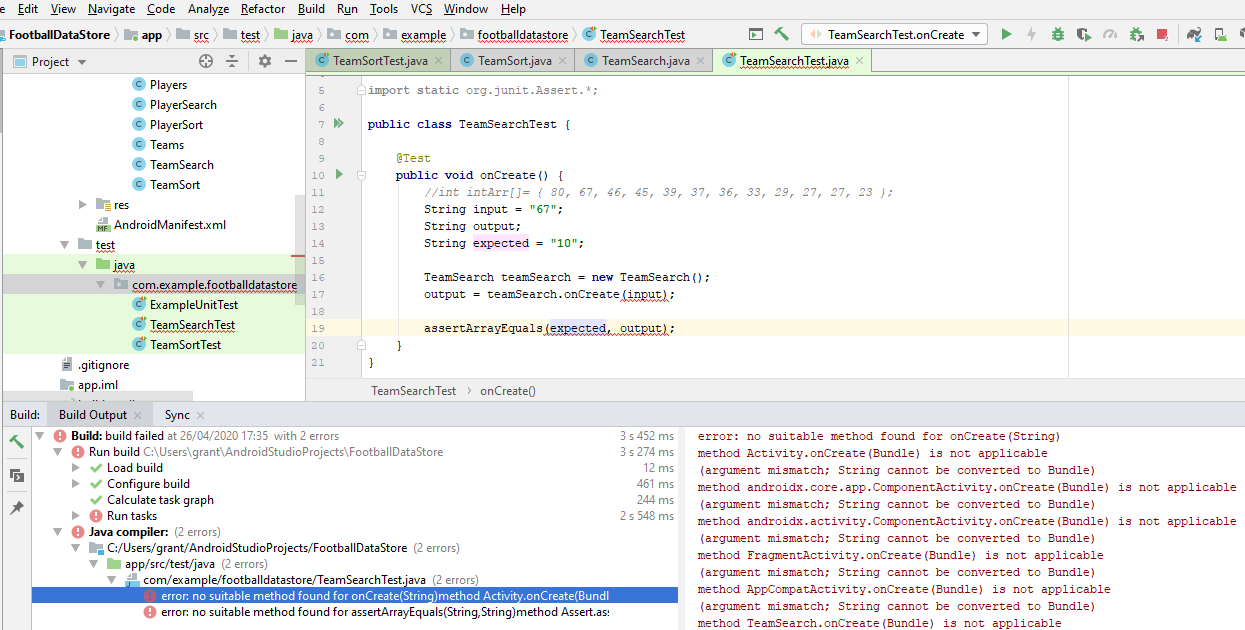
Figure 8: Shows the error generated when attempting to unit test the Sorting algorithm

Figure 9: Shows the error generated when attempting to unit test the Searching algorithm

# 10 Conclusion

SWOT analysis of the developed software, reflection on Professional, Social, Ethical, and Legal aspects, reflections on possible improvements).

In conclusion the student has successfully delivered and implemented the ‘Football Data Store’ Android mobile application proposed at the beginning of the semester, which fulfils the minimum viable product (MVP) because the app uses data structures and adopts a sorting ad searching algorithm. Furthermore, the app delivered is user friendly and has a logically navigation through the app pages. In addition, the student has been able to successfully integrate Firebase Authentication for registration, login, and logout which will help provide security and protect user data in future versions of the mobile app when dummy data is no longer used.

In reflection the adopted Rapid Application Development method allowed the student to plan and manage the implementation of the app effectively, as the business modelling, data modelling and process modelling processes where completed before any code was written, these process ensured the user requirements, data structure and sort and search algorithms were identified, and UML diagrams were complete. This made the implementation of the app more manageable. However, the student did face several challenges during the implementation process as the student struggled when attempting to implement the Tree data structure, as a result the user had to implement the Array data structure. The reason why the student struggled to implement the Tree data structure for the app was because there was not a lot of tutorials or documentation for implementing a Tree data structure into an Android app, but there was a lot more information on using the Array data structures.

Furthermore, the student also struggled to implement the proposed Quick sort algorithm because this is a complex sorting algorithm. To execute the sorting of the Player and the Team data the student adopted the quadratic sorting algorithm called the Bubble sort, which is a less complex sorting algorithm but the speed and effectiveness of the sorting execution in the implemented app is not impacted by the adoption of a less complex sorting algorithm because the student is not sorting huge amounts of data.

The major ethical consideration for this project was data and what kind of data the student used for populating the Teams and Players tables. To protect actual Teams and Players data and statistics, for the project the student used mock data of fictional Teams and Players, which do not exist. This is one of the measures taken to protect users of the app. Furthermore, to protect data and users this Android app will not be distributed to the public or be available for download on the Google Play store. Instead this app is only for academic purposes and will only be tested and used by the Developer and Lecturer.

To help improve the implementation of the Android mobile app, the student would have liked to handle better or fix the issue currently presented in the Teams and Players search pages, when the user enters a input between two valid elements in the array or list. Furthermore, for this implementation the student would have liked to have implemented the functionality to add, edit and delete Teams and Players data. However, the delivered software performs the minimum functionality and uses data structures, sorting and searching algorithms.

## 10.1 SWOT Analysis of Implemented app

Strengths

* Firebase Authentication is built into the app which provides security and control of access to the app and data
* The app adopts a binary search algorithm for searching the Team and Player data
* The app adopts a Bubble sort algorithm for sorting the Team and Player data

Weaknesses

* Currently the user can only search Team data by number of points
* Currently the user can only search Player data by number of goals
* A medium defect exists in the search pages

Opportunities

* The developer could add the functionality to add new teams and player to the data
* The developer could add the functionality to edit teams and players data
* The developer could add the functionality to delete teams and players data

Threats

* Potential security threats, however, this is reduced by having registration and login functionality
* Potential data issues if the app used real data, however, the app currently only uses dummy data

# 11 References

C++, S., B., J. and York, M. (2020). *Simple comparison of sorting algorithms in C++*. [online] Code Review Stack Exchange. Available at: https://codereview.stackexchange.com/questions/87085/simple-comparison-of-sorting-algorithms-in-c [Accessed 10 Feb. 2020].

GeeksforGeeks. (2020). *QuickSort - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/quick-sort/ [Accessed 12 Feb. 2020].

Guru99.com. (2020). *What is RAD Model? Advantages & Disadvantages*. [online] Available at: https://www.guru99.com/what-is-rad-rapid-software-development-model-advantages-disadvantages.html [Accessed 6 Feb. 2020].

McKay, V. (2020). *What is Rapid Application Development? (RAD) | Analysis and Design | FAQ*. [online] Selectbs.com. Available at: http://www.selectbs.com/analysis-and-design/what-is-rapid-application-development [Accessed 6 Feb. 2020].

Medium. (2020). *Why RAD is better in the current software development market*. [online] Available at: https://itnext.io/why-rad-is-better-in-the-current-software-development-market-9e101a4b45e2 [Accessed 6 Feb. 2020].

Productplan.com. (2020). *What is MoSCoW Prioritization? | Overview of the MoSCoW Method*. [online] Available at: https://www.productplan.com/glossary/moscow-prioritization/ [Accessed 7 Feb. 2020].

Sciencing. (2020). *The Advantages & Disadvantages of Sorting Algorithms*. [online] Available at: https://sciencing.com/the-advantages-disadvantages-of-sorting-algorithms-12749529.html [Accessed 11 Feb. 2020].

Software Testing Books. (2020). *What is Rapid Application Development Model (RAD)? Advantages and Disadvantages - Software Testing Books*. [online] Available at: http://softwaretestingbooks.com/rapid-application-development [Accessed 6 Feb. 2020].

Tutorialride.com. (2020). *Trees in Data Structure*. [online] Available at: https://www.tutorialride.com/data-structures/trees-in-data-structure.htm [Accessed 10 Feb. 2020].

Tutorialspoint.com. (2020). *Data Structure - Binary Search Tree - Tutorialspoint*. [online] Available at: https://www.tutorialspoint.com/data\_structures\_algorithms/binary\_search\_tree.htm [Accessed 12 Feb. 2020].

Tutorialspoint.com. (2020). *Data Structure and Algorithms - Tree - Tutorialspoint*. [online] Available at: https://www.tutorialspoint.com/data\_structures\_algorithms/tree\_data\_structure.htm [Accessed 10 Feb. 2020].

Tutorialspoint.com. (2020). *Data Structure and Algorithms - Quick Sort - Tutorialspoint*. [online] Available at: https://www.tutorialspoint.com/data\_structures\_algorithms/quick\_sort\_algorithm.htm [Accessed 11 Feb. 2020].

www.javatpoint.com. (2020). *Binary Search Tree - javatpoint*. [online] Available at: https://www.javatpoint.com/binary-search-tree [Accessed 12 Feb. 2020].

YouTube. (2020). *What is MOSCOW Analysis? Project Management in Under 5*. [online] Available at: https://www.youtube.com/watch?time\_continue=3&v=lO5gftPdQT8&feature=emb\_title [Accessed 7 Feb. 2020].

Cdn.bcs.org. (2020). [online] Available at: https://cdn.bcs.org/bcs-org-media/2211/bcs-code-of-conduct.pdf [Accessed 12 Feb. 2020].

Guru99.com. (2020). *Black Box Testing Vs. White Box Testing: Key Differences*. [online] Available at: https://www.guru99.com/back-box-vs-white-box-testing.html [Accessed 12 Feb. 2020].

Firebase. 2020. *Authenticate With Firebase Using Password-Based Accounts On Android*. [online] Available at: <https://firebase.google.com/docs/auth/android/password-auth> [Accessed 22 April 2020].

Firebase. 2020. *Firebase Authentication*. [online] Available at: <https://firebase.google.com/docs/auth/> [Accessed 22 April 2020].

GeeksforGeeks. 2020. *Array Data Structure - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/array-data-structure/> [Accessed 22 April 2020].

GeeksforGeeks. 2020. *Volley Library In Android - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/volley-library-in-android/> [Accessed 22 April 2020].

Tutorialspoint.com. 2020. *Data Structure - Bubble Sort Algorithm - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/data\_structures\_algorithms/bubble\_sort\_algorithm.htm> [Accessed 22 April 2020].

Tutorialspoint.com. 2020. *Java.Util.Arrays.Binarysearch() Method - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/java/util/arrays\_binarysearch\_int.htm> [Accessed 22 April 2020].

# 12 Appendix

## 12.1 Appendix 1: Proposed Interface Design

The student has only provided interface app designs for portrait orientation and will update the manifest file to disable landscape orientation for this mobile app. The main reason for this decision is to limit the amount of potential layout and design issues caused by the user rotating the screen between portrait and landscape, also if landscape was enabled then the student would need to also create designs for landscape views and set-up constraints for this view type.

### Homepage of Football Team and Player Data Store Android app



Players Label: Tapping the Players label will redirect users to the Players page

Players Logo: Tapping the Players logo will redirect users to the Players page

Teams Label: Tapping the Teams label will redirect users to the Teams page

Teams Logo: Tapping the Teams logo will redirect users to the Teams page

App Name

App Logo

### Teams Page of Football Team and Player Data Store Android app

# 

Team Table: Displays the team data for the following columns:

* P – Position
* Team – Team Name
* W – Wins
* D – Draws
* L – Losses
* GP – Games Played
* Pts - Points

Sort Button: Applies the selected sort option to the table

Sort Dropdown: The default sort value is Most Points and when the user selects the dropdown arrow, the user is presented with the other options: Ascending (A-Z), Descending (Z-A)

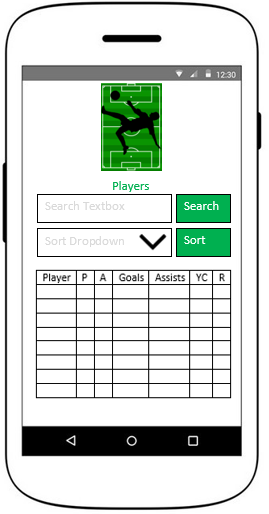
Search button: When this button is clicked the search is actioned and the output is returned on the table.

Search Textbox: Free textbox for the user to input search text

Teams Label

Teams Logo

### Players Page of Football Team and Player Data Store Android app



Team Table: Displays the team data for the following columns:

* Player – Player Name
* P – Position
* A – Appearances
* Goals – Goals
* Assist – Assists
* YC – Yellow Cards
* RC – Red Cards

Sort Button: Applies the selected sort option to the table

Sort Dropdown: The default sort value is Top Goal Scorer and when the user selects the dropdown arrow, the user is presented with the other options: Most Appearances, Ascending (A-Z), Descending (Z-A)

Search button: When this button is clicked the search is actioned and the output is returned on the table.

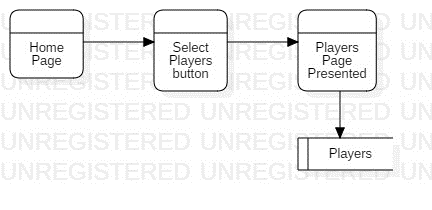
Search Textbox: Free textbox for the user to input search text

Players Label

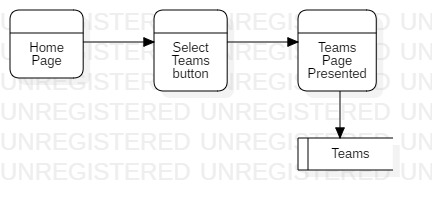
Players Logo

## 12.2 Appendix 2: Data Flow Diagrams

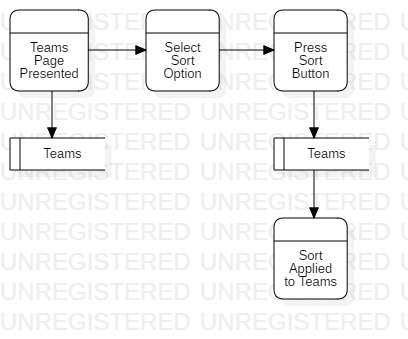
### Data flow diagram for navigating to Players Page and presenting the Players table



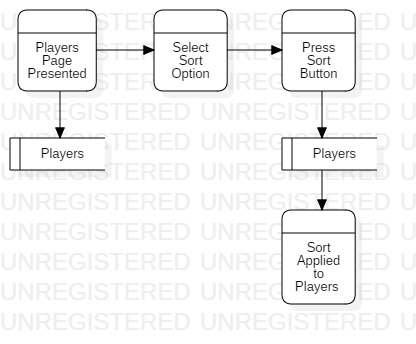
### Data flow diagram for navigating to Teams Page and presenting the Teams table



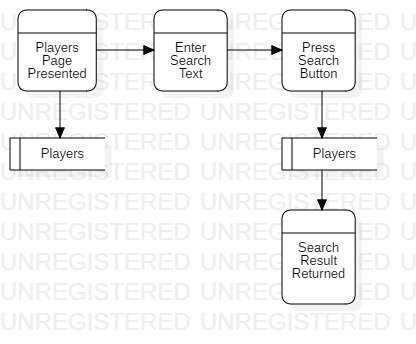
### Data Flow Diagram for applying sort to Teams table



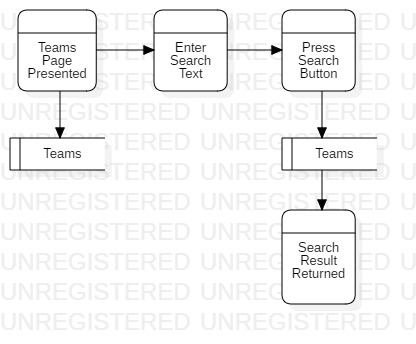
### Data Flow Diagram for applying sort to Players table



### Data Flow Diagram for applying search to Players table



### Data Flow Diagram for applying search to Teams table



## 12.3 Appendix 3: Risk Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Severity | Likelihood | Risk Owner | Action |
| Losing unsaved work and code, which would cause stress, delays and work would be required to be complete again | 10 | 3 | Grant Martin | Take regular backups of completed work, so work can be recovered quickly, and no time will be lost from the project timeline |
| Not identifying all the requirements during planning, this will result in extra unaccounted work taking place during the project | 7 | 3 | Grant Martin | Use the RAD software development lifecycle and make sure to follow this method and take full advantage of the Business Modelling phase |
| Not having access to all types of physical Android devices for testing the layout and designs of the software project, this could result in problems with UI design on some devices that aren’t found during testing | 7 | 7 | Grant Martin | Use emulators on Android studio of different android devices on different versions of Android for testing the app and specifically the layout and designs |
| Scope creep if more and more requirements are introduced to the project over its lifecycle | 7 | 3 | Grant Martin | The student will need to monitor the Gantt chart and project progress regularly to ensure there is no scope creep and ensure all requirements are identified during planning |
| Selecting the wrong data structure type for the project requirements, this could result in the software not working as expected | 10 | 4 | Grant Martin | Use the RAD software development lifecycle to take fully advantage of the data modelling and process modelling phases to ensure all planning and system processes are understood before selecting a data structure |
| Selecting the wrong searching algorithm, this could impact searching the team and player table and could result in the wrong data being returned | 9 | 5 | Grant Martin | Use the RAD software development lifecycle to take fully advantage of the data modelling and process modelling phases to ensure all planning and system processes are understood before selecting a searching algorithm |
| Selecting the wrong sorting algorithm, this could impact sorting the team and player table and could result in the table being ordered incorrectly | 9 | 5 | Grant Martin | Use the RAD software development lifecycle to take fully advantage of the data modelling and process modelling phases to ensure all planning and system processes are understood before selecting a sorting algorithm |

**Risk:** The Risk column describes the risk and what would a potential risk to the project, and what would be the impact on the project as a result of this risk happening.

**Severity**: The severity of the risk on the project if the risk occurs (scale from 0 to 10 with 10 being the highest).

**Likelihood:** The likelihood that a risk (on a scale from 0 to 10 with 10 being the highest).

**Risk Owner:** The person assigned to monitor if the specific risk as occurred and if the risk occurs, they are the one responsible for managing the risk.

**Action: Describes what action will take place if the specific risk occurs during the project.**

## 12.4.1 Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case No. | Test Description | Test Steps | Test Data | Expected Result |
| 1 | Test the user can click the app icon and the app is launched | GIVEN the user has the app install on device  WHEN the user clicks the app icon  THEN the app is launched | n/a | User is presented with the Registration page of app |
| 2 | Test the user can register with valid email and password | GIVEN the user is on the register page  AND the user is not already registered  WHEN the user enters a valid email  AND the user enters a valid password  THEN the user clicks ‘Create an Account’  AND the user is presented with the homepage of the app | Email: [s@s.com](mailto:s@s.com)  Password: spider  (was unregistered before test was executed) | User will successfully register with email and password and be presented with the homepage |
| 3 | Test the user cannot register with invalid email and invalid password | GIVEN the user is on the register page  WHEN the user enters invalid email  AND the user enters invalid password  THEN the user clicks ‘Create an Account’  AND the user is presented with the error message ‘Register unsuccessful. Please try again’ | Email: abc  Password: abc | User will not be registered and will be presented with the error message ‘Register unsuccessful. Please try again’ |
| 4 | Test the user cannot register with invalid email and valid password | GIVEN the user is on the register page  WHEN the user enters invalid email  AND the user enters valid password  THEN the user clicks ‘Create an Account’  AND the user is presented with the error message ‘Register unsuccessful. Please try again’ | Email: abc  Password: abcd | User will not be registered and will be presented with the error message ‘Register unsuccessful. Please try again’ |
| 5 | Test the user cannot register with no email and no password | GIVEN the user is on the register page  WHEN the user enters no email  AND the user enters no password  THEN the user clicks ‘Create an Account’  AND the user is presented with the error message ‘Please enter email’ | Email:  Password: | User will not be registered and will be presented with the error message ‘Please enter email’ |
| 6 | Test the user cannot register with no email and valid password | GIVEN the user is on the register page  WHEN the user enters no email  AND the user enters valid password  THEN the user clicks ‘Create an Account’  AND the user is presented with the error message ‘Please enter email’ | Email:  Password: grant | User will not be registered and will be presented with the error message ‘‘Please enter email’ |
| 7 | Test the user cannot register with valid email and no password | GIVEN the user is on the register page  WHEN the user enters valid email  AND the user enters no password  THEN the user clicks ‘Create an Account’  AND the user is presented with the error message ‘Please enter password’ | Email: d@d.com  Password: | User will not be registered and will be presented with the error message ‘Please enter password’ |
| 8 | Test the user can click the ‘Already Registered’ text and be presented with the Login Page | GIVEN the user is on the Register page  WHEN the user clicks ‘Already Registered? Sign in here’  THEN the user is presented with the Login page | n/a | The user will be presented with the Login page |
| 9 | Test the user cannot register with existing registered email and password combination | GIVEN the user is on the register page  WHEN the user enters existing registered email  AND the user enters existing registered password  THEN the user clicks ‘Create an Account’  AND the user is presented with the error message ‘Register unsuccessful. Please try again’ | Email: b@b.com  Password: batman | User will not be registered and will be presented with the error message ‘Register unsuccessful. Please try again’ |
| 10 | Test the user can login with valid email and password | GIVEN the user is on the Login page  WHEN the user enters valid email  AND the user enters valid password  THEN the user clicks ‘Login’  AND the user is presented with the Homepage | Email: b@b.com  Password: batman | User will be presented with homepage |
| 11 | Test the user cannot login with invalid email and invalid password | GIVEN the user is on the Login page  WHEN the user enters invalid email  AND the user enters invalid password  THEN the user clicks ‘Login’  AND the user is presented error message ‘Login error, please login again’ | Email: abc  Password: abc | User is presented error message ‘Login error, please login again’ |
| 12 | Test the user cannot login with invalid email and valid password | GIVEN the user is on the Login page  WHEN the user enters invalid email  AND the user enters valid password  THEN the user clicks ‘Login’  AND the user is presented error message ‘Login error, please login again’ | Email: abc  Password: batman | User is presented error message ‘Login error, please login again’ |
| 13 | Test the user cannot login with no email and no password | GIVEN the user is on the Login page  WHEN the user enters no email  AND the user enters no password  THEN the user clicks ‘Login’  AND the user is presented error message ‘Please enter email’ | Email:  Password: | User is presented error message ‘Please enter email’ |
| 14 | Test the user cannot login with no email and valid password | GIVEN the user is on the Login page  WHEN the user enters no email  AND the user enters valid password  THEN the user clicks ‘Login’  AND the user is presented error message ‘Please enter email’ | Email:  Password: batman | User is presented error message ‘Please enter email’ |
| 15 | Test the user cannot login with valid email and no password | GIVEN the user is on the Login page  WHEN the user enters valid email  AND the user enters no password  THEN the user clicks ‘Login’  AND the user is presented error message ‘Please enter email’ | Email: b@b.com  Password: | User is presented error message ‘Please enter password’ |
| 16 | Test the user can click the ‘Not Registered’ text and be presented with the Register Page | GIVEN the user is on the Login page  WHEN the user clicks ‘Not Registered? Register here’  THEN the user is presented with the Register page | n/a | User is presented with the Register page |
| 17 | Test the user can click the ‘Teams’ button on the homepage and be presented with the Teams page | GIVEN the user is logged in  AND the user is on the homepage  WHEN the user clicks the ‘Teams’ button  THEN the user is presented with the ‘Teams’ page | Email: b@b.com  Password: batman | User is presented with the Teams page |
| 18 | Test the user can click the ‘Players’ button on the homepage and be presented with the Players page | GIVEN the user is logged in  AND the user is on the homepage  WHEN the user clicks the ‘Players’ button  THEN the user is presented with the ‘Players’ page | Email: b@b.com  Password: batman | User is presented with the Players page |
| 19 | Test the user can click the ‘Logout’ button on the homepage and be presented with the Register page | GIVEN the user is logged in  AND the user is on the homepage  WHEN the user clicks the ‘Logout’ button  THEN the user is presented with the ‘Registration’ page | Email: b@b.com  Password: batman | User is presented with the Registration page |
| 20 | Test the user can leave/kill the app when logged in and then reopen the app and still be logged into the app | GIVEN the user is logged into the app  WHEN the user kills the app  AND the user relaunches the app  THEN the user will be presented with the homepage | Email: b@b.com  Password: batman | User is presented with the Homepage |
| 21 | Test the user can select the ‘Get Team Data’ button on the Teams page and be presented with the Teams data | GIVEN the user is on the Teams page  WHEN the user clicks ‘Get Team Data’  THEN the user is presented with the team data | Email: b@b.com  Password: batman | User is presented with the team data in the scrollable text view |
| 22 | Test the user can scroll the Team Data returned in the text view | GIVEN the user is on the Teams page  WHEN the user clicks ‘Get Team Data’  THEN the user is presented with the team data  AND the user can scroll the team data | Email: b@b.com  Password: batman | User is presented with the team data in the scrollable text view |
| 23 | Test the user can select the ‘Get Player Data’ button on the Players page and be presented with the Players data | GIVEN the user is on the Players page  WHEN the user clicks ‘Get Player Data’  THEN the user is presented with the player data | Email: b@b.com  Password: batman | User is presented with the player data in the scrollable text view |
| 24 | Test the user can scroll the Player Data returned in the text view | GIVEN the user is on the Players page  WHEN the user clicks ‘Get Player Data’  THEN the user is presented with the player data  AND the user can scroll the team data | Email: b@b.com  Password: batman | User is presented with the player data in the scrollable text view |
| 25 | Test the user can click the ‘Sort’ button on the Teams page and be presented with the Team names sorted in ascending order (A-Z) | GIVEN the user is on the Teams page  WHEN the user clicks ‘Sort’  Then the user is presented with the sorted team data (A-Z) | Email: b@b.com  Password: batman | User is presented with the Team data sorted from A – Z by team name |
| 26 | Test the user can scroll the Sorted Team Data returned in the text view | GIVEN the user is on the Teams page  WHEN the user clicks ‘Sort’  Then the user is presented with the sorted team data (A-Z)  AND the user can scroll the team data | Email: b@b.com  Password: batman | User is presented with the Team data sorted from A – Z by team name |
| 27 | Test the user can click the ‘Sort’ button on the Players page and be presented with the Player names sorted in ascending order (A-Z) | GIVEN the user is on the Players page  WHEN the user clicks ‘Sort’  Then the user is presented with the sorted player data (A-Z) | Email: b@b.com  Password: batman | User is presented with the Player data sorted from A – Z by player name |
| 28 | Test the user can scroll the Sorted Player Data returned in the text view | GIVEN the user is on the Players page  WHEN the user clicks ‘Sort’  Then the user is presented with the sorted player data (A-Z)  AND the user can scroll the player data | Email: b@b.com  Password: batman | User is presented with the Player data sorted from A – Z by player name |
| 29 | Test the user can click the ‘Search’ button on the Teams page and be presented with the Teams Search Page | GIVEN the user is on the Teams page  WHEN the user clicks ‘Search’  Then the user is presented with the search page | Email: b@b.com  Password: batman | User is presented with the Team search page |
| 30 | Test the user is presented with the ‘Team Search’ data in the scrollable text view | GIVEN the user is on the Teams page  WHEN the user clicks ‘Search’  Then the user is presented with the search page  AND the user can scroll the team data | Email: b@b.com  Password: batman | User is presented with the Team search page and the user can scroll team data |
| 31 | Test the user is presented with the search bar with the hint text ‘Search Points’ | GIVEN the user is on the Teams page  WHEN the user clicks ‘Search’  Then the user is presented with the search page  AND the user is presented with the search bar and search button | Email: b@b.com  Password: batman | User is presented with the Team search page |
| 32 | Test the user is presented with the search button on the Teams page | GIVEN the user is on the Teams page  WHEN the user clicks ‘Search’  Then the user is presented with the search page  AND the user is presented with the search bar and search button | Email: b@b.com  Password: batman | User is presented with the Team search page |
| 33 | Test the user does not enter a search and clicks the search button and the user is presented with the Teams page | GIVEN the user is on the Team Search page  WHEN the user enters no search  AND the user clicks search button  THEN the user is presented with the Teams page | Email: b@b.com  Password: batman | User is presented with the Teams page |
| 34 | Test the user enters a valid search points number and the row of the searched points is returned in the result text view | GIVEN the user is on the Team Search page  WHEN the user enters valid search  AND the user clicks search button  THEN the user is presented with the row the search result was found | Email: b@b.com  Password: batman  Valid search value entered: 80 | User is presented with the row the search result was found in the result text view |
| 35 | Test the user enters letters in the search bar and the clicks ‘Search’ and the user is presented with the Teams page | GIVEN the user is on the Team Search page  WHEN the user enters invalid search  AND the user clicks search button  THEN the user is presented with the Teams page | Email: b@b.com  Password: batman  invalid search value entered: a | User is presented with the Teams page |
| 36 | Test the user enters an invalid points number above the highest points value on the list and then clicks the search button and ensure the user is presented with the result ‘Search not found’ | GIVEN the user is on the Team Search page  WHEN the user enters valid search higher than the highest points value  AND the user clicks search button  THEN the user is presented with the error message ‘Search not found’ | Email: b@b.com  Password: batman  invalid search value entered: 90 | User is presented with error message ‘search not found’ in result text view |
| 37 | Test the user enters an invalid points number under the lowest points value on the list and then clicks the search button and ensure the user is presented with the result ‘Search not found’ | GIVEN the user is on the Team Search page  WHEN the user enters valid search lower than the lowest points value  AND the user clicks search button  THEN the user is presented with the error message ‘Search not found’ | Email: b@b.com  Password: batman  invalid search value entered: 20 | User is presented with error message ‘search not found’ in result text view |
| 38 | Test the user enters an invalid points number in between two points values on the list and then clicks the search button and ensure the user is presented with the result ‘Search not found’ | GIVEN the user is on the Team Search page  WHEN the user enters valid search between two points values  AND the user clicks search button  THEN the user is presented with the error message ‘Search not found’ | Email: b@b.com  Password: batman  invalid search value entered: 75 | User is presented with error message ‘search not found’ in result text view |
| 39 | Test the user can click the ‘Search’ button on the Players page and be presented with the Players Search Page | GIVEN the user is on the Teams page  WHEN the user clicks ‘Search’  Then the user is presented with the search page | Email: b@b.com  Password: batman | User is presented with the Player search page |
| 40 | Test the user is presented with the ‘Player Search’ data in the scrollable text view | GIVEN the user is on the Players page  WHEN the user clicks ‘Search’  Then the user is presented with the search page  AND the user can scroll the player data | Email: b@b.com  Password: batman | User is presented with the Player search page and the user can scroll player data |
| 41 | Test the user is presented with the search bar with the hint text ‘Search Goals’ | GIVEN the user is on the Players page  WHEN the user clicks ‘Search’  Then the user is presented with the search page  AND the user is presented with the search bar and search button | Email: b@b.com  Password: batman | User is presented with the Player search page |
| 42 | Test the user is presented with the search button on the Players page | GIVEN the user is on the Players page  WHEN the user clicks ‘Search’  Then the user is presented with the search page  AND the user is presented with the search bar and search button | Email: b@b.com  Password: batman | User is presented with the Player search page |
| 43 | Test the user does not enter a search and clicks the search button and the user is presented with the Players page | GIVEN the user is on the Player Search page  WHEN the user enters no search  AND the user clicks search button  THEN the user is presented with the Players page | Email: b@b.com  Password: batman | User is presented with the Players page |
| 44 | Test the user enters a valid search goals number and the row of the searched goals is returned in the result text view | GIVEN the user is on the Player Search page  WHEN the user enters valid search  AND the user clicks search button  THEN the user is presented with the row the search result was found | Email: b@b.com  Password: batman  Valid search value entered: 12 | User is presented with the row the search result was found in the result text view |
| 45 | Test the user enters letters in the search bar and the clicks ‘Search’ and the user is presented with the Players page | GIVEN the user is on the Player Search page  WHEN the user enters invalid search  AND the user clicks search button  THEN the user is presented with the Players page | Email: b@b.com  Password: batman  invalid search value entered: a | User is presented with the Players page |
| 46 | Test the user enters an invalid goals number above the highest goals value on the list and then clicks the search button and ensure the user is presented with the result ‘Search not found’ | GIVEN the user is on the Player Search page  WHEN the user enters valid search higher than the highest points value  AND the user clicks search button  THEN the user is presented with the error message ‘Search not found’ | Email: b@b.com  Password: batman  invalid search value entered: 30 | User is presented with error message ‘search not found’ in result text view |
| 47 | Test the user enters an invalid goals number under the lowest points value on the list and then clicks the search button and ensure the user is presented with the result ‘Search not found’ | GIVEN the user is on the Player Search page  WHEN the user enters valid search lower than the lowest points value  AND the user clicks search button  THEN the user is presented with the error message ‘Search not found’ | Email: b@b.com  Password: batman  invalid search value entered: 0 | User is presented with error message ‘search not found’ in result text view |
| 48 | Test the user enters an invalid goals number in between two goals values on the list and then clicks the search button and ensure the user is presented with the result ‘Search not found’ | GIVEN the user is on the Player Search page  WHEN the user enters valid search between two points values  AND the user clicks search button  THEN the user is presented with the error message ‘Search not found’ | Email: b@b.com  Password: batman  invalid search value entered: 18 | User is presented with error message ‘search not found’ in result text view |

## 12.4.2 Test Results

|  |  |  |  |
| --- | --- | --- | --- |
| Test case No. | Actual Result | Pass or Fail | Evidence |
| 1 | User is presented with the Registration page of app | PASS |  |
| 2 | User will successfully register with email and password and be presented with the homepage | PASS |  |
| 3 | User will not be registered and will be presented with the error message ‘Register unsuccessful. Please try again’ | PASS |  |
| 4 | User will not be registered and will be presented with the error message ‘Register unsuccessful. Please try again’ | PASS |  |
| 5 | User will not be registered and will be presented with the error message ‘Please enter email’ | PASS |  |
| 6 | User will not be registered and will be presented with the error message ‘‘Please enter email’ | PASS |  |
| 7 | User will not be registered and will be presented with the error message ‘Please enter password’ | PASS |  |
| 8 | The user will be presented with the Login page | PASS |  |
| 9 | User will not be registered and will be presented with the error message ‘Register unsuccessful. Please try again’ | PASS |  |
| 10 | User will be presented with homepage | PASS |  |
| 11 | User is presented error message ‘Login error, please login again’ | PASS |  |
| 12 | User is presented error message ‘Login error, please login again’ | PASS |  |
| 13 | User is presented error message ‘Please enter email’ | PASS |  |
| 14 | User is presented error message ‘Please enter email’ | PASS |  |
| 15 | User is presented error message ‘Please enter password’ | PASS |  |
| 16 | User is presented with the Register page | PASS |  |
| 17 | User is presented with the Teams page | PASS |  |
| 18 | User is presented with the Players page | PASS |  |
| 19 | User is presented with the Registration page | PASS |  |
| 20 | User is presented with the Homepage | PASS |  |
| 21 | User is presented with the team data in the scrollable text view | PASS |  |
| 22 | User is presented with the team data in the scrollable text view | PASS |  |
| 23 | User is presented with the player data in the scrollable text view | PASS |  |
| 24 | User is presented with the player data in the scrollable text view | PASS |  |
| 25 | User is presented with the Team data sorted from A – Z by team name | PASS |  |
| 26 | User is presented with the Team data sorted from A – Z by team name | PASS |  |
| 27 | User is presented with the Player data sorted from A – Z by player name | PASS |  |
| 28 | User is presented with the Player data sorted from A – Z by player name | PASS |  |
| 29 | User is presented with the Team search page | PASS |  |
| 30 | User is presented with the Team search page and the user can scroll team data | PASS |  |
| 31 | User is presented with the Team search page | PASS |  |
| 32 | User is presented with the Team search page | PASS |  |
| 33 | User is presented with the Teams page | PASS |  |
| 34 | User is presented with the row the search result was found in the result text view | PASS |  |
| 35 | User is presented with the Teams page | PASS |  |
| 36 | User is presented with error message ‘search not found’ in result text view | PASS |  |
| 37 | User is presented with error message ‘search not found’ in result text view | FAIL |  |
| 38 | User is presented with error message ‘search not found’ in result text view | FAIL |  |
| 39 | User is presented with the Player search page | PASS |  |
| 40 | User is presented with the Player search page and the user can scroll player data | PASS |  |
| 41 | User is presented with the Player search page | PASS |  |
| 42 | User is presented with the Player search page | PASS |  |
| 43 | User is presented with the Players page | PASS |  |
| 44 | User is presented with the row the search result was found in the result text view | PASS |  |
| 45 | User is presented with the Players page | PASS |  |
| 46 | User is presented with error message ‘search not found’ in result text view | PASS |  |
| 47 | User is presented with error message ‘search not found’ in result text view | FAIL |  |
| 48 | User is presented with error message ‘search not found’ in result text view | FAIL |  |