

Common Cents

Final Year Project Report

DT282

BSc in Computer Science (International)

Alex Byrne – C15492952

Oisin Creaner

School of Computing

Technological University Dublin

12th April 2019

# Abstraction: Common Cents

Common Cents is a mobile application that helps the user save small amounts of cash on a regular basis and ensures that they have fun doing it as it is based on the English Premier League Football results. It allows the user to have fun saving and adds an extra element of excitement to watching mid-week or weekend football matches. Financial stability is a concern for many people, such as students, who struggle to save money. This application can solve this issue as it allows the user to make predictions on the English Premier League Football matches and allows them to also set a stated amount of money to this prediction as the figure they wish to save if their prediction is correct. Up to this point, it is similar to betting on football matches in the local book maker, but after this is where Common Cents differs greatly and allows the user to save money rather than lose excess money in the bookmakers.

Once a prediction has been set and saved to a remote database, a script will run two to three times a day and that script will check whether the set predictions have occurred or not. If the prediction has not materialised, then nothing happens to the money the user set on the prediction. However, if a prediction comes true, then the amount set will be transferred from the user’s current account to their savings account where it can gather a higher interest rate. Given restrictions to a real bank database, a mock bank database on a Heroku server has been created, which replicates a real bank database. The idea behind this is to set two to three predictions with a limited amount of money and if the predictions are correct, the user saves small amounts of money regularly which over time will growth at a compound rate. To help the user make predictions, Comment Cents’ machine learning model posts the predictions of each match to firebase and the application reads this and displays the probability of each possible outcome of each match. This helps the user make predictions on each match.

Common Cents also contains a myriad of other functionalities. For example, Common Cents contains a results page which displays the recent match results and the match facts. These two functionalities enable the user to educate them self about historical trends in the data to enhance their future predictions. Common Cents also has a profile page, where the user can update their profile information, and a home page that displays the user’s open predictions.

# Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:



Alex Byrne

12/04/2019

# Acknowledgements

I would Like to thank all my friends in college and outside of college for the any help provided throughout the year and they also provided support in helping me to achieve this project. Tips in areas such as Grammar and implementation were provided and came in very useful. To this I am very grateful for any and all help in the process of creating this project.

I would like to thank my family for being there for me this year and every year as they always are. They support me in everything I do, and I am very appreciative of that.

I would like to Thank Oisin Creaner, this projects Supervisor for the constant help throughout the year. The constant feedback on the project itself and the reports was very helpful.

Contents

[Abstraction: Common Cents 2](#_Toc5977712)

[Declaration 3](#_Toc5977713)

[Acknowledgements 4](#_Toc5977714)

[1. Introduction 9](#_Toc5977715)

[a. Overview 9](#_Toc5977716)

[b. Goals and Objectives 9](#_Toc5977717)

[c. Challenges and obstacles 10](#_Toc5977718)

[2. Research 11](#_Toc5977719)

[a. Overview 11](#_Toc5977720)

[b. Framework 11](#_Toc5977721)

[1. Swift – iOS devices 11](#_Toc5977722)

[2. Xamarin – iOS/Android/Windows Devices 12](#_Toc5977723)

[3. Android Studio – Android Devices 13](#_Toc5977724)

[c. Databases 14](#_Toc5977725)

[1. SQL 14](#_Toc5977726)

[2. NoSQL – Firebase 15](#_Toc5977727)

[d. Alternative and Existing Solutions 16](#_Toc5977728)

[1. Spending Tracker 16](#_Toc5977729)

[2. Revolut 17](#_Toc5977730)

[3. Money Lover 17](#_Toc5977731)

[e. Payment Methods 18](#_Toc5977732)

[1. Visa 18](#_Toc5977733)

[2. PayPal 18](#_Toc5977734)

[3. Dwolla 19](#_Toc5977735)

[4. Stripe 20](#_Toc5977736)

[5. Mock Bank Database 20](#_Toc5977737)

[f. REST APIs 21](#_Toc5977738)

[g. Machine Learning 22](#_Toc5977739)

[1. Data 22](#_Toc5977740)

[2. Outcome 23](#_Toc5977741)

[3. Algorithm – K Nearest Neighbours 23](#_Toc5977742)

[4. Algorithm – Decision Tree 25](#_Toc5977743)

[5. Algorithm – Linear Regression 26](#_Toc5977744)

[6. Algorithm – Poisson Regression 27](#_Toc5977745)

[h. Football Data APIs 28](#_Toc5977746)

[1. XML Soccer 28](#_Toc5977747)

[2. API Football 28](#_Toc5977748)

[i. Servers 29](#_Toc5977749)

[1. Heroku 29](#_Toc5977750)

[2. JavaScript Checker 30](#_Toc5977751)

[j. Resultant Findings and Requirements 30](#_Toc5977752)

[1. Framework – Android Studio 30](#_Toc5977753)

[2. Database – Firebase 30](#_Toc5977754)

[3. Existing Solutions 31](#_Toc5977755)

[4. Payment System – Mock Bank 31](#_Toc5977756)

[5. REST APIs 31](#_Toc5977757)

[6. Machine Learning – Poisson Regression 32](#_Toc5977758)

[7. API Football 32](#_Toc5977759)

[3. Design and Architecture 34](#_Toc5977760)

[a. Approach and methodologies 34](#_Toc5977761)

[1. Overview 34](#_Toc5977762)

[2. Agile Crystal-Clear Methodology 34](#_Toc5977763)

[3. Waterfall Model 35](#_Toc5977764)

[4. Iterative Waterfall Model 36](#_Toc5977765)

[5. Priorities 38](#_Toc5977766)

[b. Application Architecture 38](#_Toc5977767)

[1. General Architecture 38](#_Toc5977768)

[2. Activity Architecture 39](#_Toc5977769)

[c. Screen Flow 40](#_Toc5977770)

[d. Main Screens Demonstration 41](#_Toc5977771)

[1. Overview 41](#_Toc5977772)

[2. Register 41](#_Toc5977773)

[3. User Area 42](#_Toc5977774)

[4. Results Information – Results List Clicked 43](#_Toc5977775)

[5. Fixture information – Fixture List Clicked 44](#_Toc5977776)

[6. Event Information – Event List Clicked 45](#_Toc5977777)

[7. Profile and Information Change 46](#_Toc5977778)

[e. System Architecture 47](#_Toc5977779)

[1. Overview 47](#_Toc5977780)

[2. Pipe – Filter Architecture 47](#_Toc5977781)

[3. Client Server Architecture 48](#_Toc5977782)

[4. Model View Controller Architecture 49](#_Toc5977783)

[5. Final System Architecture 50](#_Toc5977784)

[f. Use Cases 52](#_Toc5977785)

[1. Register 52](#_Toc5977786)

[2. Using Application 55](#_Toc5977787)

[3. Machine Learning 58](#_Toc5977788)

[g. Flow Chart: Processes in Application 61](#_Toc5977789)

[h. Sequence Diagram 62](#_Toc5977790)

[1. Main Application Sequences 62](#_Toc5977791)

[2. JavaScript Prediction Checking 63](#_Toc5977792)

[4. Implementation & Development 64](#_Toc5977793)

[a. Overview 64](#_Toc5977794)

[b. Databases 64](#_Toc5977795)

[1. Bank Database 64](#_Toc5977796)

[2. Firebase Database 65](#_Toc5977797)

[c. Servers 67](#_Toc5977798)

[1. Heroku & PHP Scripts 67](#_Toc5977799)

[2. JavaScript 69](#_Toc5977800)

[d. Android Application 71](#_Toc5977801)

[1. Classes 71](#_Toc5977802)

[2. List Activities & Adapters 72](#_Toc5977803)

[3. User Area Activity 74](#_Toc5977804)

[4. Verification Activities 75](#_Toc5977805)

[5. Register Activity 76](#_Toc5977806)

[6. Event Information 78](#_Toc5977807)

[7. Fixture Information 80](#_Toc5977808)

[8. Result Information 82](#_Toc5977809)

[9. Profile Fragment 82](#_Toc5977810)

[10. AESEncryption Class 84](#_Toc5977811)

[d. Python Machine Learning 86](#_Toc5977812)

[1. Libraries 86](#_Toc5977813)

[2. Data Management 86](#_Toc5977814)

[3. The Model 87](#_Toc5977815)

[4. Predictions Function 88](#_Toc5977816)

[5. Function Call and Matrix Operations 89](#_Toc5977817)

[6. Firebase Data Preparation and Posting 89](#_Toc5977818)

[5. Testing 91](#_Toc5977819)

[a. Overview 91](#_Toc5977820)

[b. Black Box Testing 91](#_Toc5977821)

[Q1. Were you able to register? 91](#_Toc5977822)

[Q2. Were you able to Log in immediately and consistently? 91](#_Toc5977823)

[Q3. Were you able to change profile information? 92](#_Toc5977824)

[Q4. Were you able to create and delete your predictions? 92](#_Toc5977825)

[Q5. Did the Live Scores work during matches? 92](#_Toc5977826)

[Q6. How did you find the match’s outcome probabilities (ML)? 92](#_Toc5977827)

[c. Black Box Testing Conclusion 93](#_Toc5977828)

[d. White Box Testing 93](#_Toc5977829)

[1. Code Walkthrough 93](#_Toc5977830)

[2. Loop Testing 94](#_Toc5977831)

[3. Path Testing 94](#_Toc5977832)

[e. White Box Testing Conclusion 95](#_Toc5977833)

[6. Project Plan 97](#_Toc5977834)

[a. Overview 97](#_Toc5977835)

[b. Project Plan 97](#_Toc5977836)

[1. Team Goals Trigger 97](#_Toc5977837)

[2. Machine Learning 97](#_Toc5977838)

[3. Payment System 98](#_Toc5977839)

[c. GANTT Chart 98](#_Toc5977840)

[d. Strengths 99](#_Toc5977841)

[1. Machine Learning 99](#_Toc5977842)

[2. Overall Application 99](#_Toc5977843)

[e. Weaknesses 100](#_Toc5977844)

[1. Machine Learning 100](#_Toc5977845)

[2. Real Life Banks 100](#_Toc5977846)

[7. Conclusion 101](#_Toc5977847)

[Figure Table 102](#_Toc5977848)

[Figure Table 103](#_Toc5977849)

[Resources 104](#_Toc5977850)

# Introduction

## Overview

This report will outline the key areas in researching, designing, implementing and testing of this project. Each section will outline the critical elements that were used in each part of the application.

The research section contains all the options considered when beginning the application and gives an insight into each aspect and why each aspect was or was not chosen. This report looks at the framework options for the project along with the different machine learning algorithms that were considered and why they would or would not have been suitable the application based on the data available.

The design section analyses what was considered in relation to overall design of the application. For example, what design methodologies were used and considered based on what was best for the Common Cents application along with multiple screenshots of the application and screen flows etc. The architecture will also be outlined within this section and the use-case narratives will be displayed in this section along with other design diagrams.

The implementation and development section will elaborate on how each area of the project was constructed using code snippets and explanations. Each component will be discussed along with the outcomes of each component and any issues discovered and addressed throughout development.

The testing section will describe how testing for the Common Cents application was conducted and addressed. This section will detail the key findings and issues discovered while testing, along with how they were addressed and fixed. A description of the testing methodologies will be provided to illustrate the testing process. The project plan will explain how the plan has developed and changed over time, and how everything came together. The conclusion will form an overarching view of the entire project and will outline the key paradigms of discussion from the report.

## Goals and Objectives

The central goal of this project is to have a fully functional mobile application which enables the user to save money in a fun, easy and simple manner. To achieve this goal, the REST APIs and machine learning application must function correctly. The REST APIs will ensure the application is easy to use, whilst the machine learning will focus on producing statically strong predictions based of a thorough analysis of each team’s seasonal record. This will aid the user to make informed financial allocation decisions.

A crucial goal of the project is simple design and strong aesthetics. Design and aesthetics account for a large proportion of the user experience and reflects greatly on the rest of the application. This would entail each screen not being too clustered, having all the relevant information and having the visuals on the screen look crisp, clear and concise.

Learning is key goal of the project. Through learning, obstacles can be overcome and opportunities to enhance Common Cent application and its user’s experience can be identified and implement.

## Challenges and obstacles

The main adversities that was identified upon starting the project were what machine learning Algorithm and API would best fit Common Cents’ concept and what payment method should be utilized.

For the machine learning, the major concern identified related to the data, datasets and the algorithm. It easy to locate a dataset for the seasons results. However, querying the dataset and assessing how the algorithm was going to be applied to the dataset was a major concern. Some of the categorical data can be predetermined from the start of the season, for example, who the referee is and where the match is being held. This data can be relevant to a machine learning algorithm, but it is difficult to find a dataset detailing this information until after the match has been played.

Another major adversity about the API mainly lies the possibility of the data layouts changing, and in turn, not working with the application even though it may have been working in the past.

Furthermore, a big threat that the project faced from the beginning was the payments and how they were going to be constructed as using platforms, such as Stripe or PayPal, would cause the feasibility of the project the drop considerably and hold the saving aspect of the project essentially void. If the user must pay a few cents per transaction using platforms, such as Strip or with PayPal, then there is a huge problem. A solution for this is critical.

# Research

## Overview

This section will cover all the research that was done throughout the project. Every aspect of the project needs to be researched in numerous ways to ensure the direction the project is going to go is correct. This saves time and energy in the future as a plan can be made and stuck to. This can prevent implementation iterations further down the line of the project. The areas of research include but aren’t limited to, the application frameworks considered, the machine learning algorithms considered, different payment methods, databases etc.

## Framework

Many types of frameworks were researched and considered for this project. The main frameworks that were considered were Swift for iOS devices, such as an iPhone, Android Studio for Android devices, and finally Xamarin with Visual Studio, which is used for developing applications for multiple mobile operating systems such as iOS, Android and Windows. The choice of framework is key as it outlines the specific target market your application will be aimed at, whether its iOS users, Android users or any other type of smart phone users. This decision limits your target audience by putting restrictions on who can use the application. Therefore, the correct choice is imperative to the success of the application.

### Swift – iOS devices

The initial path for the project was thought to be developing the application for an iPhone and iOS devices. As stated on the Apple website, “Swift is a powerful and intuitive Programming Language for macOS and iOS” [1]. Apple claim that their recent replacement for Object-C is “interactive and fun” [1]. “The Syntax is concise yet expressive” and a boasted “Lightning fast” software was hugely fitting for the project and the direction the project was taking[1]. Upon further research into the framework, it was found that there were issues with the language.

In a study conducted by Marcel Reboucas and his team, they found some issues with the language[2]. They found that 20% of the questions asked to the Swift Developer team are Syntax related even though Apple claim that it is a simple language to understand. However, they found that the people they interviewed did not find a statically significant amount of issues with the syntax[2].

A larger concern, in relation to the project, was the issue that Marcel Reboucas and his team found with developers having problems in creating user interface elements such as buttons in Swift due to Swift having Cocoa Touch as its core framework[2]. This framework is challenging for the development of user interface elements[2]. This was a major issue in terms of the project as the user interface plays a massive factor as a crucial goal in making application as easy and fluid as possible for the user.

The deciding factor to not use the Swift framework was the XCode paradigm that was discovered. Upon further consideration and research, it was discovered that XCode is needed to program for iOS devices and use the Swift framework[3]. XCode is a free software provided by Apple, but a device with macOS is necessary in order to run the software as Apple do not provide the software as a standalone product[3]. As the computer that will be used in this project is Windows and not Mac, this is an obstacle that cannot be left avoided. There is information about creating a virtual machine of macOS, so the framework could be used by someone with a windows device, but this has many issues, such as, it is illegal, and it is a very long and complicated process. Therefore, it was decided that Swift would not be the framework for the Common Cents project.

### Xamarin – iOS/Android/Windows Devices

Next Xamarin was considered as the software used to develop the mobile application. Xamarin uses Visual Studio from Microsoft. This tool allows the developer to create a user interface for Android, iOS and windows mobile devices[4]. The idea behind this is great for any application as the developer can create the one interface for a mobile device and the software will convert that interface into something usable for the other platforms. Following the research on Swift, there was a growing concern surrounding the development for an application that uses the Swift framework. Regardless, Xamarin was considered for the project.

Microsoft states that their Xamarin tools will ensure the applications interface controls and looks to the user how the developer intended[4]. Another selling point for Xamarin was the “Native API access” [4]. Microsoft state that their framework has access to a variety of great functionality for all devices that the application may be used on and they say it includes “platform specific capabilities” [4]. This was a huge factor for the project to have great functionality and make that functionality as easy to use as possible for the user.

Despite Xamarin offering many benefits, Xamarin has a large pitfall. Each platform a developer develops on has its own set of demands and needs when it comes to development[5]. The main issue still stands of developing for iOS is that one needs a Mac device to compile and debug the code written for the mobile device, meaning that the application cannot be developed for iOS for the time being[5].

There are three main operating systems that would be used currently[6]. Android and iOS mobiles hold the majority of world’s mobile operating system market share, leaving Window’s phones with a small percentage of the market share[6]. This means that Xamarin offers a far smaller target audience than Android and iOS [6]. Given this, using Xamarin to develop Common Cents was considered a weak strategic move Common Cents, and as such, the use of Xamarin was ruled out.

### Android Studio – Android Devices

Android Studio is based on the Java programming language, this is a large factor in choosing what software to develop an application[7]. As Android Studio and Java is currently taught in Dublin Institute of Technology, some familiarity with the software and the syntax from java has previously been obtained. This is a huge benefit to the project and its development.

The design elements of Android Studio such as buttons, widgets and layouts are mainly done through XML[7]. XML and Java-based classes create functionality and usability. With this structure in the software, it ensures that the elements are easy to create and that the functionality and syntax is developer-friendly to create. These paradigms are a large deciding factor in the choice of software to develop the application.

The Android Studio documentation online is a beneficial resource to have when developing for android devices as it concisely explains aspect of the software to help developers address with any issues they may have[8]. The documentation offers in-depth chapter analysis on all the different possibilities that Android Studio has to offer. For example, it will tell the developer how to debug your application if issues arise.

Android Studio also takes advantage of the IntelliJ Idea from Jet Brains. This is a Java IDE that is “specifically designed to maximize developer productivity[9].” This is a large asset to any software as it makes “development not only productive but also an enjoyable experience[9].” JAVA IDE predicts what the developer is going to type in terms of Java code and if the developer opts to go with the suggestion then it will print the prediction on the page, and often, it is accurate. Besides being accurate, JAVA IDE allows the developer to program their application faster than they normally can. For example, by creating the functionality on an OnClickListener to press enter rather than typing the whole function. Having this tool, as said earlier, is a great asset and a bonus to using Android Studio.

The fact that 76.6% of all smartphones are using the Android’s operating system [10], and the fact that Android is a market-share leader in its industry makes Android the ideal platform to utilize to develop Common Cents. The next decision is then between working with Android Studio or another Java based software like Eclipse. Looking into this topic further, Android studio was selected over Eclipse to building Common Cents for a myriad of reason.

Android studio uses an integrated version of Gradle which is a tool which assists developer “model, integrate and systemize the delivery of your software end to end[11].” It allows you to easily integrate third party applications onto your application with a Gradle build using “implementations”. This can be seen in the application level Gradle where the likes of Firebase, Volley and GitHub charts can be added to Common Cents.

The code completion also seems to be slightly better and more accurate in Studio[10]. Also, the graphical user interface of Android Studio seems to be second to none as it is very easy to follow and locate specific classes and libraries you may be looking for and Eclipse doesn’t seem to have one, the overall organization of the interface seems to be just more user friendly in Android Studio. All these Factors are what lead the decision to opt for the Android Studio software for developing the application.

## Databases

SQL and NoSQL were the two types of databases considered for this project. SQL is a relational database, whereas NoSQL is a non-relational database. A database was required for this project to store user related data, such as user predictions.

### SQL

Android Studio has a built in SQL Helper for applications using SQL databases[7]. The SQL Helper uses a local SQL database to allow developers to query off and post to when designing their application. This is an undesirable state for the developer.

The second option considered for building Common Cents involved using a remote SQL database supported by Volley Requests. This would have involved creating REST APIs to generate user profiles and to pull their information from the database. Given the type of data required for this project, it was clear that the SQL relational database was overly complicated for this project.

The Amazon Web Services Relational Database (AWSRD) was another consideration for Common Cents [12]. AWSRD is a remote database that uses SQL. Upon further analysis, it was decided that using a relational database would be going beyond the scope of what was needed for the user database [12]. The user database primarily collates user information, such as their email address, name, phone number, their predictions for each event they place a prediction on and the Machine Learning Predictions that have been received from a separate program.

Given points 1,2 and 3, SQL was ruled out as a suitable database for the applications information, such as, user information, crest URLs, probabilities and a user’s predictions.

### NoSQL – Firebase

In comparison to SQL and other relational databases, NoSQL databases have a simpler way to loop through data and offer developers a better way of querying large volumes of data [13]. Non-relational databases are generally of JSON structure. The JavaScript Object Notation offers developers more functionality than relational databases do as it consists of Key: Value pairs[13]. The expected queries for the application will run more effectively and efficiently with this type of database especially with large amounts of data queried.

Firebase was investigated as the main database system for the application. Firebase is an online, real time database that stores the data a developers sets in a JSON format[14]. Firebase is simple to integrate into Android Studio and offers a vast amount of functionality to help the user/developer in their project development.

The documentation for Firebase is concise and comprehensive, offering guides through all situations where you may need Firebase and its functionality[15]. Moreover, Firebase, like Android Studio, is user friendly and are both Google developed systems so can easily be used in conjunction with one another. Another key benefit of using Firebase is its ability to be used on all mobile device such as iOS device, Android device, a tablet and a device using JavaScript/HTML. This offers scalability opportunities for the future of the system.

“The Iconic SDK is developed for hybrid apps, powered by Cordova as the underlying architecture, using web technology (HTML, CSS, JavaScript), developed on Angular front-end architecture, and can be used on different mobile platforms, including iOS, Android and Windows phones[14]. With Iconic SDK, one can use one codebase to suit all mobile platforms.” This Real Time Database (RTD) is an excellent source of storing user and developer information. The RTD allows different “Controllers” to post data to the server[14], it acts as a server by receiving the data and it “forwards the data onto the Firebase Cloud”. Once the data is in the cloud, a device can retrieve that data[14]. To do this, the database takes a “Data Snapshot” of the database, stores a copy of the data in that indicated branch of the tree and allows the user / developer to place a query to gain access to the information they require[14]. This ensures that if the database is updated whilst the data is being retrieved, that the data being retrieved is not affected or corrupted on the mobile device.

For these reasons (POINT 1,2,3) it was clear that Firebase was the correct database to use for this project.

## Alternative and Existing Solutions

There are numerous mobile savings applications available on the market today. Most of them have a unique way of helping the user to save money over time. Most modern ways of saving money consist of firms incentivising their customers to take small amounts of money out of their current accounts on a regular basis and deposit this money in money-generating vehicle that the company controls. This can be seen in Revolut’s vault system[16], and in Loan repayments in banks, small regular payments. The behavioural science behind this is that by only taking a small portion of the customer’s income on a regular basis, the customer factors this small expense into their regular budget and pays little attention to it. Over time, their money builds interest and by the end of their investment horizon, they have, hopefully, build up a relatively large savings account. This money can be used to fund discretionary items such holidays and luxury goods.

The user is unlikely to pay a large amount of attention to the money leaving their current account. However, the user is likely to pay a large amount of attention to the large sum in their savings account. This is what makes this sort of saving attractive to people, especially college students. In today’s society, people believe the ideology that they should be “working smart” rather than “working hard”[17]. This paradigm applies to saving too. People do not want to sacrifice small things they enjoy like the cinema or going out for lunch. Saving smart allows the user to maintain their normal lifestyle, whilst still saving money for future events.

### Spending Tracker

Spender Tracker is an mobile application centred around the traditional idea of saving money. [18]. Spender Tracker is a free application for iOS and Android. Once an account is made, the user is asked to categorize all of their recent expenditure into categories, such as fuel, food and clothes [18]. This application allows the user to visualize all their spending and track their daily expenditure. They can see exactly what category their money is going into and, from there, they can assess how they can reduce their expenditure per category. If they feel they can reduce a given category, they can focus on that category and trying to limit how much they spend. Spending Tracker takes the old traditional way of spending and the modern phenonium that everything is accessible of our mobiles and combines these paradigms to give users a way to visualize and track their daily expenditure. Spending Tracker would target some segments of Common Cents user base. However, Common Cents differentiates itself from Spending Tracker as it uses machine learning to assist people save and because it is not a generic saving application, it is centred around sport. Unlike Spending Tracker that is aimed at anyone interested in saving, Common Cent has a clearer defined target marketing, being people who enjoy sports and fall within the age range of 18 to 40.

### Revolut

Revolut is a synonymous saving application. Each time a user completes a transaction, the application rounds up their transaction cost to the nearest euro and places the remainder into their “Vault” [16]. Revolut has launched “one of the easiest and fastest ways to save towards your financial goals.” As the market leader in the FinTech savings industry, Revolut would be one of Common Cents’ largest competitors. Common Cents saving account and the vault both originate from a similar concept. These small and regular influxes into user savings account will slowly compound and accumulate a considerable amount in the users “Vault” [16].

This vault can be accessed later as saved money, but it is not accessible from the user’s card[16], so the user is not inclined to spend this money as if it was a current account. How this differs from Common Cent is that it is based off the user’s daily transaction, and to use the platform, you also need a Revolut banking account. Many people do not want to set up another bank account but would still like the opportunity to save money. This difference between Common Cents and Revolut’s business model is what sets the two brands apart. As Common Cents does not require its users to set up a new bank account it is more user friendly. To access the Common Cents application, users simply need to complete the one-page registration which only takes a matter of minutes. From then on, their account is active.

### Money Lover

Money Lover is a third competitor of Common Cents. Money Lover is a free mobile application that helps the user to save money in a similar way to Spending Tracker [18]. This application allows the user to create a monthly or weekly budget or wallet. This is the users allowance for the stated timeframe set by the user. If the user spends over their allowance, the application will notify them that they are doing so[18]. However, this will not cap the users spending. Similar to Common Cents, Money Lover is a free mobile application and there is no requirement to sign up for a bank account like the Revolut option [18]. Common Cents would be aimed at a different target market to Money Lover, so the number of users being poached by Money Lover would be limited.

## Payment Methods

### Visa

One major aspect of this project is the way in which the money is transferred from the user’s current account to their savings account. Many directions have been considered for this area of the project. Initially, a sandbox form was considered. This would involve using a sandbox form of VISA payments to replicate and test linking dummy current accounts and savings account [19]. Theoretically, this method can be used to allow the user of the application to make payments from their actual current account to their savings account.

This type of technology is highly secure and difficult to obtain as it could be a security risk to the VISA payments. Upon further research and a discussion with the Project Supervisor[19], it was agreed that this direction for a payment system would not be feasible. Furthermore, Visa would charge a transaction fee per payment resulting in limited saving opportunities. So while this technology is available, it is suited to large businesses rather than small projects like this[19].

### PayPal

PayPal was the next payment method assessed. This type of payment would consist of special rules and methods for the payment[20, p. 7]. This payment method requires additional information for the payment, such as address, which would need to match the delivery address with the card billing credentials.

Firstly, the setting of the transaction would have to take place[19]. Next, the developer would need too initialize the “SET” with an initialization and a response for the buyer and the seller. The seller and buyer would need to be verified and all the payment information would need to be verified. Once this is complete, the purchase request data would have to be completed, and finally, authorized [20, p. 7]. This would be suitable for a business-to-business payment, like Visa, PayPal would take a percentage of the transaction as payment. PayPal have a sandbox like system to which a user can make a mock example and test it as if it was real. This causes issues for the payments as the developer/ user needs to make a PayPal account and have their card making payments to that account. This meaning the PayPal account would be acting as a savings account[21]. This is an issue as there would not be limited interest on this account, and overall, the account would not make sense in terms of a real-life practical project[21].

This payment method would flourish for the project if it was a business extracting payments off clients or a person taking payments off another person as it is a secure payment system, but it is not[21]. This is not ideal method of payment for a savings application due to the amount of money taken out of each transaction by the company who owns the payment method.

Also, the developer would not be able to implement this system directly into the Common Cents. Upon research into how it would be implemented into an Android application, it was discovered that each transaction would have to be initiated by the user of the application in a WebView in the application itself which would not have been conducive for Common Cents users [22]. This would be a huge drawback for the fluidity of the application. A main selling point would be the automation of the payments with as little user interaction as possible, so they do not notice the small payments. If the user had to initialize every transaction, then they could easily opt out and this would defeat the concept of the application.

### Dwolla

Dwolla is an alternative payment method. This is a payment system that can be integrated into a variety of programming languages such as Java, Python and Ruby[23]. This meant the software could be use it in Android Studio as it is based on Java. Like PayPal, users can engage in peer-to-peer transactions and also business-to-business transactions[23].

A large selling point for this method of payments is that for payments under $10, it was a free transaction[23]. Dwolla, like PayPal, have a sandbox system to replicate transactions on their software. One pitfall to this is that they do not have the sufficient guidance documentation on how to implement the software.

Furthermore, Dwolla do not have up-to-date documentation on the Java implementation nor do they maintain this documentation like they do for JavaScript or Ruby[23]. Dwolla also do not maintain the regular update to the Java program like they do for the other programs. If a developer contacts the company, they provided a link to a GitHub account that has an old example of how the software is implemented in Java, but this is not provide sufficient insight into the process of how to implement the software.

Similarly to PayPal, one of the issues is that a user of Common Cents would have to create a Dwolla bank account and use this account as a savings account which goes against the value proposition of this project[23] which is to allow customers to save money in a fun, easy and simple way. By forcing users to have Dwolla bank accounts, the developer is complicating the Common Cent customer experience.

Given the need to create a Dwolla savings account and lack of documentation on how to implement the software, this payment method was rejected for this project too.

### Stripe

Stripe was also considered as a potential payment method. This too was a third-party payment system to which the user enters their card details and then creates the payment and pays the set account the money assigned in the payment request. To use Stripe in the development of an application, multiple steps were needed[24].

Due to security reasons, Stripe does not allow developers to store the card details of the user on the application that uses Stripe[24]. Stripe does not want to allow developers to have permanent access to this data so once the user provides their card details, the function is called to make the payment and when that function is called all the card details are stored in a token. This token is then sent to the Stripe servers for validation and authentication[24]. From here, the payment is initiated and processes and finally completed. This was not suitable for the Common Cents application as the goal was to have an automated payment system to which the user would not have to interact with often to add fluidity.

Another benefit of using Stripe is that the software provides a widget for card details which looks appealing in terms of design. It has its own animations and looks professional [24].

Visa, PayPal, Dwolla and Stripe are business products and must make money. Therefore, they all charge a fee per transaction when customers use their software. This is not suitable for Common Cents because not only will the users profit margins per win decrease, if the user places a prediction and the prediction materialises, they will face a deficit as a result of the transaction cost the payment provider enforces on them. The value that Common Cents offers customers is to save money, not spend money aimlessly. For this reason, the Stripe software was dismissed, and a new way of payments was needed.

### Mock Bank Database

With a new form of payments needed and most third-party payment systems ruled out, a blue ocean strategy was utilised[25]. This is a pursuit of filling a gap in the market using a lost cost solution but being able to keep up demand requirements[25]. This involved creating a mock bank database that stores bank client information using a relational database. The relational database created follows a SQL format and data is accessible from there. This involved creating unique REST APIs that replicate a real-life product. [26]

A relational database is a database that is structured to recognize relations between stored items of information. For example, in the case of a bank, a customer may have 3 bank accounts for different things, so this would be a “one to many” relationship. This is a much more structured database then the likes of a JSON database. Querying this type of data from Android Studio form to a remote server is a complexed but doable feature.

The key advantages to using a relational database are simple operations and security. Furthermore, SQL is a user-friendly software and has a simple set of implementation instructions for a developer [26]. Given this, SQL is so widely used throughout the world today. The security in a relational database is excellent. Read-only permissions can be granted by the administrator. This means that the data can be accessed and only be read by a machine. However, data operations cannot occur from that machine[26]. For this reason, a relational database in SQL is secure and it cannot be changed without stringent procedures. The bank payment system that the project required specified that a fully functioning real bank system was not required. However, the project’s database required key components to be cited and meet. These components are reflected as table in the database, under names such as Customer, Account and Transaction. Given this, the SQL mock blue ocean database was the best option for the payment system and to create custom REST APIs.

## REST APIs

With the Plan to move forward with the SQL relational database some research on REST APIs was needed. “The REST architecture style assumes that client and server form a contract with content negotiation, not only on the data format but implicitly also on the semantics of the communicated data[27].” In simple terms a RESTful API is an application that uses the HTTP requests to “GET, PUT, POST and DELETE” data from a database[28]. REST stands for Representational State Transfer. These APIs are very practical for a project as it builds on the app that you are using APIs for and adds features and functionality to the application[28]. When considering what language to use to create the REST APIs, two main languages stood out, JavaScript using Node and PHP. These are the two most commonly used languages for creating APIs[28].

PHP has been the most widely user language for REST APIs since its release, for years it was thought of as the best but lately with the relatively new node.js framework it seems that PHP may be surpassed[29]. PHP powers millions of web portals worldwide but node.js is the up and coming younger web development tool. “Node is not a language but a run-time environment that uses JavaScript for the server-side application development. [29]” Node.js is a fast server- side solution[29]. With this you can create very scalable projects that use the likes of streams to process data which tend to be a very fast Asynchronous type of programming. Many great new development frameworks such as react use JavaScript also, so this would work very well in conjunction with that, although this project doesn’t use that much JavaScript.

Unfortunately, node is quite new, and it lacks a maturity that the likes of PHP has[29]. There is a large amount of small minor bugs in the system that can cause issues for developers as bugs like this can only be fixed properly over time with experience and usage of the environment. On the other side of things with PHP it has been around for years and therefore all those early bugs in the code have been dealt with. Also, PHP has a “Rich Code Base,” meaning that it includes popular platforms for building websites and different we applications[29]. PHP is very portable, and it works with almost any platform like Windows, Linux or Mac. Your code can be run anywhere once it is programmed. It rarely needs altering once it has completed unless new features are added.

One down side to using PHP would be it doesn’t fit too well with the Model View Controller architecture, but I feel this shouldn’t affect the project too much as it is a simple REST API rather than a full program[29]. Due to these reasons, it was clear that PHP is the best option for Common Cents and for creating the REST APIs for the banking system. For the use that it is needed, it will fit extremely well, and do exactly the job is needed of it and one big factor is the lack of bugs PHP will have as its been around for enough tie to have all those issues sorted.

## Machine Learning

### Data

The first thing to consider when conducting a machine learning project is the question or problem you are trying to solve and predict. In the case the question being asked is who is going to win the game and what percentage each outcome holds in terms of the probability of it happening. This would be the displayed to the user and it must be accurate to a certain degree. This question leads on to the algorithm most suited to answering the question which in turn leads to the examining of the data that project has at its disposal.

When the Machine Learning aspect of the project came into play one of the next important things to consider was the Algorithm to be used. With this question being asked a lot of research was needed as to ensure the right algorithm was picked for the job. The biggest factor for picking choosing an algorithm is the Data that you have. You must “understand your data” [30]. Some algorithms work better with smaller sample sets and others work better with huge datasets. E.g. Naïve Bayes works much better with categorical data rather than numerical data and is considered very sensitive to missing data.

The data available in this case is a .csv file from <http://www.football-data.co.uk/englandm.php> . Here you can find football data for the top 5 leagues in the UK. The data goes back to the 1990s. Each file has a huge amount of information on each match that has been played. For example, it has goals scored at half time by each team, at full time by each team, the referee, the stadium and much more. As there are only 380 matches per season this dataset is very good as it makes up for the lack of matches a season with the categorical and continuous data about each match.

### Outcome

The next thing to consider is what exactly I want from the Machine Learning program. Rather than giving a score prediction of each match I felt it was better to have a sort of odds situation for each game much like an actual betting service. What I mean by this is give and odds or percentage of each team’s likelihood of winning the match is and what is the likelihood of the match being a draw. This probability can be displayed to the user of the application and hopefully can give an indication of how each team is playing in the season in terms of their form but also it is good to have it this way, so the user can’t become reliant on the prediction of the application. This inclines the user to be more likely to watch the games on television or help them keep a better track of how the seasons football is going. It wouldn’t be ideal for the project if it just gave the user a score, so they could just copy it. It is best that the user can make their own predictions.

Image result for machine learning classifiers

Figure 1: ML Classifier Example Diagram

### Algorithm – K Nearest Neighbours

From here the next step is to investigate what algorithm is going to be used. The initial process for deciding was to look at well-known classifiers for the data to be put into. The first classifier looked at was K-nearest-Neighbours (KNN). This is a relatively simple classifier. Essentially, each of the rows of data are marked on a certain position on a plot graph and given a class e.g. A or B. Next the query row is plotted on the graph also and from the closest K (usually 3) nodes are taken in and whichever class has the majority closest to the query node is considered the class for the query node.

One issue with KNN is the lack of efficiency with the algorithm, this limits the applications that can use this classifier[31]. “It cannot be used in some dynamic web mining applications and its dependency on the selection on a good value for K” [31]. This algorithm works great with predicting a categorical outcome for the queries[31]. Although the data of the teams for the year contains categorical features, considering the type of output and how the output is not going to be like class A or class B, the KNN algorithm cannot be used. The example on the next page illustrates the KNN algorithm.

There are 2 classes, the green triangle and the red star. The query is plotted on the graph, the question mark, and then depending on the value for K the query is given a class. When K =3 the query is classed as a green triangle but when K = 7 then it is classes as a red star. This is a large issue for this sort of classifier, so another reason that it shouldn’t be used in my case. Example[32].

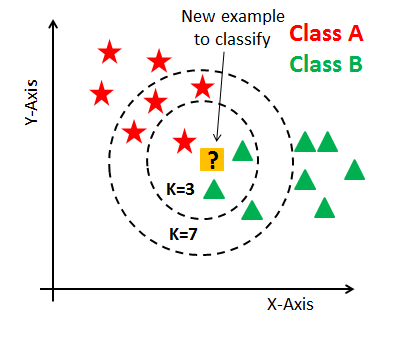


Figure 2: K-NN Algorithm Diagram

### Algorithm – Decision Tree

A Decision tree was also considered as the way forward for the projects machine learning algorithm. This classifier in a way asks a question at each node in the tree, at each question the sample gives a certain answer and that decides which node the sample goes to next. “The sample is classified into a class using one or several decision functions in a successive manner” [33].

The tree will consist of a root node. For good algorithm the is root node is decided by the granularity of each question[33]. In other words, the information gain from each decision function is what causes the tree to be a successful algorithm or not. Imagine a game of “Guess Who,” this is essentially the basis behind a decision tree algorithm. Each player starts with 24 people, each player then selects a card to which is a matching person of the other persons 24 people. Your goal is to determine who your opponent has by asking yes or no questions on at a time (like a decision tree, class A or B). the player will eliminate people based on their opponents answers until one is left or until you guess the right person.

The concept is the exact same as a decision tree as each question you ask, the better the granularity and information gain, the more people you can eliminate so you can get as many people down as fast as possible and for the tree get the best possible answers from the decision functions[33]. Although the decision tree is very flexible to different situations, the whole accuracy and efficiency of the tree is based on the design of it, so if there is a poor design then the classifier will be also not up to scratch[33]. This again unfortunately will not suit my program as it classifies the outcome into certain classes. The information that will be focused on will be the numerical data. Therefore, this classifier too isn’t best suited to the project. For this reason, this classifier will not be used. Example [34].

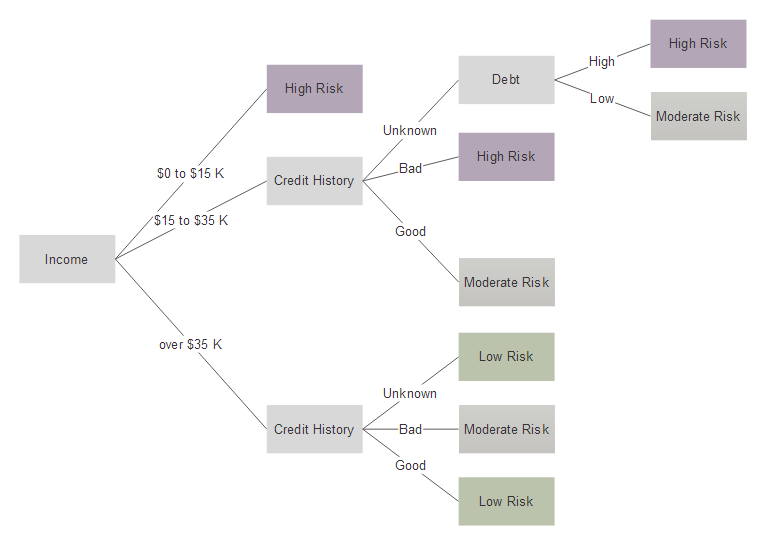


Figure 3: Decision Tree Algorithm Diagram

### Algorithm – Linear Regression

Linear Regression is also a very well recognized algorithm for machine learning. It is used a lot for numerical data when you want to “compute some continuous data as compared to a classification. [30]” In the project it deals with a mainly numerical data, so this is a step forward in the right direction.

This similarly to KNN plots all the data on a graph and models the relationship by fitting a linear equation onto that data[35]. Essentially, the data is plotted on the graph and each point on the graph is classified as one or the other etc. then, theoretically, a line is draw splitting the classes to sides on the graph[35]. This has a high risk of incorrectly fitting the line at this step. Aim is to get a good like that accurately represents the classes. Plotting this data really helps to class each sample query correctly to a high accuracy and that the line will represent the data well[35], rather than a line that isn’t split fairly over the data sue to outliers.

A type of regression most likely the best suited sort of algorithm for the data that has been received for the project. Although this is the case there are some drawbacks to be aware of. For example, Linear regression is limited to linear relationships as state in the name[36]. it only looks at the relationships between dependent and independent variables. This algorithm assumes a straight-line relationship between the dependent and independent variables, even though in real-life datasets sometimes it is not the case. Therefore, the fitting of the straight line is crucial in the algorithm. This could drastically affect the accuracy of the model. Example[37].

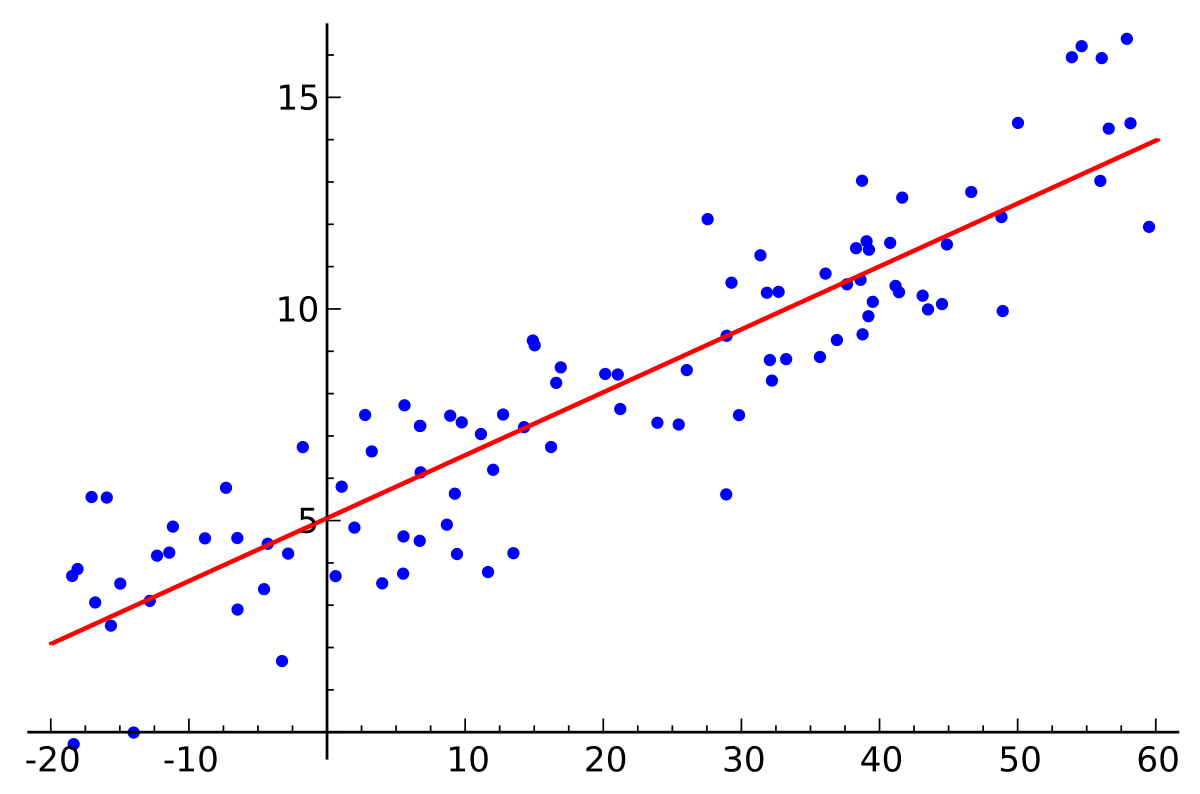


Figure 4: Linear Regression Algorithm Diagram

### Algorithm – Poisson Regression

Once it was decided that a form of regression for the machine learning algorithm was the way to go, Poisson Regression was investigated firmly, and it was very much a promising direction. This is similar to the linear regression algorithm talked about above bar a few differences[38]. The main difference is that the errors follow a Poisson distribution instead of a normal distribution.

This is a “discrete probability distribution that expresses the probability of a given number of events occurring” at a given time[39]. This distribution finds the probability of a number or events happening given a variable such as time. We can expect an independent event, in this case goals for a team, to occur N amount of time over a given time interval[39]. The data from the dataset can be put into the algorithm to get a Poisson distribution[39]. Trying to fit the data into this model would be something along the lines of, what is the probability of a certain amount of goals to be scored or conceded by the teams that are playing the match.

Counts of events based on the Poisson Distribution can be seen widely throughout the medical world[40]. They are widely seen in Asthma attacks and number of cells studies. Lambda (λ)is a parameter in the Poisson distribution and it is found by getting the total number of events (k) divided by the total number of units (n) in the dataset and seen as (λ = k/n) in mathematical terms[40]. λ will represent the average number of goals in this case and so the home and away goals are treated as independent events in the Poisson distribution.

In saying this, the teams towards the top of the league such as Manchester City and Arsenal will have a much more likely chance of scoring goals then the teams in the lower section of the league. “This model is also a type of Generalized Linear Model (GLM) model where the random component is specified by the Poisson distribution of the response variable which is a count and in this case is the count of goals for the team[41].” This model seems to very effective for the type of machine learning that I want in the project. This can give me the probability of the goals scored by the teams in the match. The decision of going forward with the Poisson Regression model can now be made as it is what is needed for the project.

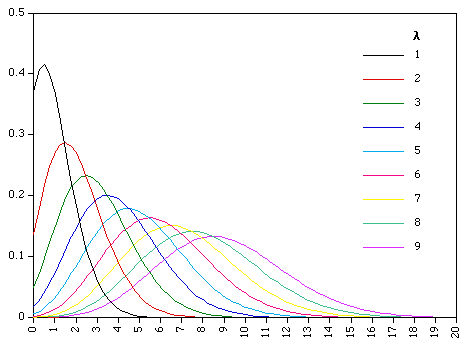


Figure 5: Poisson Regression Algorithm Diagram

## Football Data APIs

For the application to work and for the whole idea to function, a solid API is needed that stores all the match statistics that are needed for the project from numerous football matches. e.g. Football scores, fixtures, match facts and countless other information. The data from this API needs to be consistent and reliable. The data format and style must remain the same for the duration for the project to ensure no redesign is needed late into the project’s development.

This information is needed to make triggers for the payments and to display all the information to the user to give them the best user experience as possible. Therefore, it is needed to have a solid, consistent and reliable database that won’t change over the year. This would most likely break the application and as stated earlier a redesign of certain parts may be needed if that happens. There were 2 main contenders for the API that was going to be used for the application, which are XMLSoccer and APIFootball.

### XML Soccer

XMLSoccer was the first API considered. They offer an unofficial Java library, this is linked to a GitHub page from their website. Here you can request the information you need and then process it and do with it as you wish, much like any normal API[42]. As stated in the websites name, rather than returning an ideal JSON array, the API returns a list of XML objects. It is widely thought that XML is quite outdated and that JSON is the much better type of data structure. Ideally for the application Data of JSON structure would be obtained[42]. You must create the list of data you requested yourself and structure it as such. This would be very time consuming and on top of that it does not have the level of detail I wanted for the application in terms of match facts and overall data.

Using a JSON array would be a lot more efficient for the application as it has Key: Value pairs and this would be a lot easier to process. This would also fit into Android Studio much easier. With XML there is the risk of needing more third-party plugins such as XSD and XSL to reformat the XML data and display it to the user as required. For these reason XMLSoccer is not the source of data for the project.

### API Football

APIFootball was the next website considered for supplying the data in an API call. It turns out APIFootball is exactly what was wanted[43]. This is the direction that will be followed. This API provides a JSON array response when called. In this JSON array is a huge amount of data about each and every match or fixture to come[43]. It has match facts such as possession and shot on target etc.

One main page in the application is to have the match facts displayed to the user so they can see exactly how the match went. You can request a lot of information at a single time with this API, as long as you handle it correctly in the code you can request as much as you want. [43] On top of all this, they supply a great documentation for all sorts of requests that a developer may need to call. This documentation is very helpful in this sense. This is a great asset. The Key: Value pairs from the JSON array makes looping through the different parts of data very efficient and quick. This will boost efficiency and run time of the application.

All the possible information for the matches that could be needed or wanted is supplied by this API. Unfortunately, with this comes a price as for the application, the premier league is the ideal league to have as it is the most popular league worldwide. €25 is the price to get the European bundle which consists of the English premier league[43]. Fortunately, I can use the free version with the English championship as the league up until I want to add in the Premier League teams. I will just pay for the last 2 months to have the ideal league for the project and application. So, APIFootball is the API that will be pushing forward for the application and its data source for the Football information that it very much needed to progress further in development.

## Servers

### Heroku

For this project and especially the payment system with all the REST APIs, a server of some sort is needed. This server would hold all the PHP scripts for the application that are needed for the updating of the accounts for the user if the predictions they set come true[44]. Heroku was the solution to this problem. This is an online service to which it allows you to host a server free of charge. It allows you to commit your PHP or other scripts to their product online[44]. It generates a unique URL for your server and you can make requests to the URL that is provided[44]. This remote server is very useful as it eliminates any issues with a firewall on a localhost server. With the Scripts stored remotely, they can be requested by an application and will respond as programmed.

### JavaScript Checker

For a JavaScript file like this, the normal option is to host the Script on a server with the rest of the database that is being used for the other functionality of the system. This JavaScript can then be called a given number of times a day. The initial idea for completing this task was to use the JavaScript file in conjunction with a cron job. Upon further research cron jobs only work with Linux servers and usually work well with a node.js environment rather the PHP scripts. As this application is using Heroku and firebase as the systems to host the remote aspects of the application, a cron job was could not be utilised.

Given this, the solution was to go with a JavaScript but rather then running on a server, the “SetInterval()” method was to be implemented. This allows the function in question to be executed repeatedly after that interval has completed. E.g. The function will run once every second if the interval is set to 1000. The interval can be set any time in milliseconds. This script can be executed from Google Chrome on any computer.

Due to this offering the same functionality of a cron job, the JavaScript file checker was decided that it was the solution Common Cents was going to proceed with.

## Resultant Findings and Requirements

### Framework – Android Studio

with all this research it was clear to where the project was going as many conclusions had been come to especially in terms of what was going to be used for each section of the application. One of the biggest conclusions and decisions was the choice of using Android Studio over any other framework for any other mobile device. This was the best decision as a huge amount of the mobile phone market is Android, so it is a great starting point to use Android Studio, as this branches out the target market as much as possible due to the Android devices market share. Another massive help and bonus to using Android Studio is the vast amount of information about all topics in the online documentation that they provide to developers.

### Database – Firebase

The next conclusion that this research found was that firebase was a great factor for the main database that the application will use. This easy to follow JSON database is great as it is real-time and provides a second to none user interface. It is the best remote server database for the application. The strong connection between Firebase and Android Studio is also very helpful for development as it makes the integration and initial setup very easy to follow which in turn makes sure it is done correctly. The Key: Value pairs that are used in Firebase will be a great help for efficiency and for looping through to find different aspects of the application and database.

### Existing Solutions

The other, rival, application that help users save money have also been outlined and taken note of. The discovery of these applications reiterates the unique idea behind this project in how it offers the user a much more hands on and more fun approach to saving money. This idea will encourage the user to save money through its unique integration of the English Premier League matches. This will also bring more enjoyment to the user out of their weekly football matches. They can watch with more enjoyment knowing there is something on the line but also knowing that if they lose then there are no repercussions financially. As many students and even others will know it is difficult to save money so this can be a new way that helps people do this and brings enjoyment it also.

### Payment System – Mock Bank

The payment system has been decided to be a mock Bank System to which it replicates a real bank system. So, the idea is to have this application set up with bank accounts, so they work in conjunction with banks and their customers so have this feature to save the money they wanted. This direction is being pursued rather than a sandbox of a third-party payment system due to the fact that most of them charge per transaction which defeats the whole purpose of saving money. You will end up losing money going down that direction as in often cases the charge is very high on the transaction. This also adds to the functionality of the application and adds to the complexity as it means custom REST APIs will be created for the project which add a lot of complexity and will help show more skills that have been obtained through the completion of this project.

### REST APIs

The REST APIs are done in PHP rather than any other language or environment. Node.js was the biggest contender but due to its new and recent development, it was felt that the option of PHP was slightly better as it would be slightly more reliable as it has been around for longer and any minor bug are more than likely fixed now but the same cannot be said for node.js. although node.js would have been helpful as using JavaScript throughout the project would be nice but showing off PHP skills also can be a big benefit. JavaScript will be used elsewhere in the checks for the predictions and whether they come true or not. This check will run once a day If not more and will update any bank accounts that need be. Then that prediction will be removed off firebase. The next day more checks will be run and so on.

### Machine Learning – Poisson Regression

Lastly is the Machine Learning Algorithm. It was decided to proceed with the Poisson Regression algorithm as it fits perfectly with the application and can do exactly what is required for the project. The model can give the probability of how many goals will be scored and conceded by each team in a given match based on the goals the team has been scoring throughout the season so the better the team in the league the better chance they have of having a higher amount of goals scored and vice versa for the teams at the bottom of the table. This will also consider the “home team advantage” as there is a higher chance of a team winning if they are home and by a considerable amount.

This section of the project will be conducted in Python. There was never any question for this. This will give a great model with a relatively accurate prediction of what team will win or whether it’ll be a draw or not. It will produce a percentage for the home team winning, for a draw and for the away team winning. Once this is predicted python allows you to post directly to firebase from the program which is great news. Once these predictions are in firebase then the application can pull this information off firebase and use it in the application.

### API Football

The API to retrieve football data has also been decided and concluded. Form the get go a JSON format response was wanted and with XMLSoccer that was not going to happen as they only do XML files, but with APIFootball a JSON formatted response was there to be used. With the JSON the application can run more efficiently. XML is somewhat outdated for the likes of this. It is used more in enterprise systems and business systems as with the likes of XSL and XSD can help the XML be more secure and more efficient and useful for larger companies. In the case of an Android Application JSON is superior. IT has endless amounts of data for the football matches that are requested, and the archives go back many years. They also claim to offer live scores for the football matches but that is yet to be seen, so hopefully if they get that working it can be incorporated into the application. Although this costs money I feel it is worth is for the application and project as it will give a better usability if the teams are premier league related rather than the championship.

# Design and Architecture

## Approach and methodologies

### Overview

A software development or system development methodology in software engineering is a framework that is used to plan out structure and control the processes in the development of an application or system. Three different methodologies were considered for this project upon the initial approach. There was an agile methodology known as “Crystal Methodology”, a waterfall methodology and finally an iterative methodology. There is a huge list of different types of methodologies, but these are the most common methodologies in real-life situations when it comes to software development. Fundamentally these methodologies are used to highlight the different processes and different areas involved in creating an application or software system. This is very important as it brings to light which processes are critical and may need revisiting and highlights the priorities of the processes. This can be a key detail to what makes or breaks an application.

### Agile Crystal-Clear Methodology

The first of the three methodologies considered was the Agile methodologies known as “Crystal Methodology”. This methodology was developed by Alistair Cockburn[45]. This focuses greatly on the users of the system as it is based off the belief that the community of users, their skills, talents and communications are “what have the first order effect on performance.” In the case of this project, Common Cents, this would consist, mainly of the project focusing more so on the interaction of the users, the needs of the users and the wants of the community using the application.

The name crystal refers to the faces or sides of a crystal. There are many different faces on the crystal and in this case, it implies that there are many different teams of developers working on the application all with different skills, it derives from the different views of the method in terms of the principles and the values. These views are basically a portrayal of different techniques, tools, standards and rules[45]. This translates to each team on the project, which it does not have, having different talents and skills and “therefore each team should use a process uniquely tailored to it.” This project would have to processes and programming to be second to the people involved in the production of it. For this reason, in this case it is not the correct approach for the application and its success. The programming, complexity and the applications features are what are most important along with the documentation and report. Example[46].

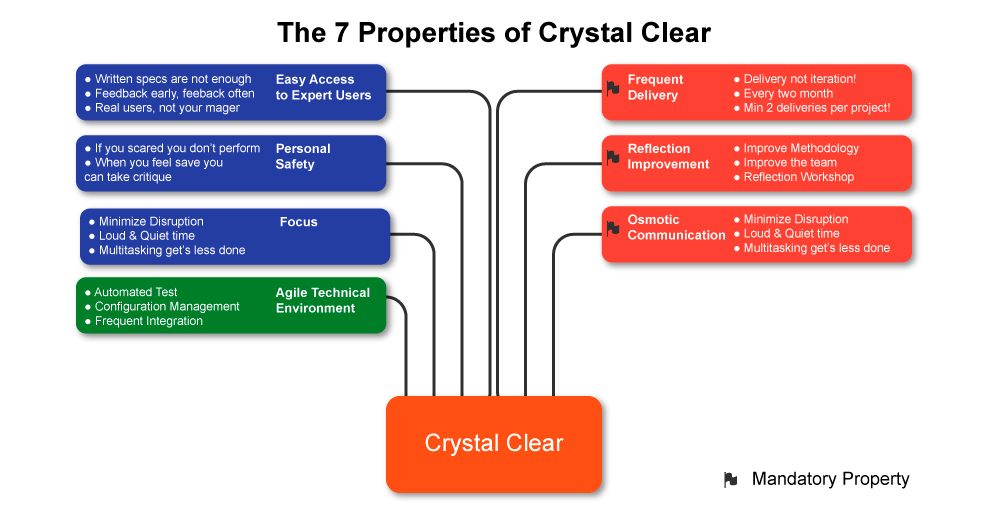


Figure 6: Agile Crystal-Clear Methodology

### Waterfall Model

The Waterfall methodology was also examined as the approach of the project. This waterfall model is very popular in software development and most people have come across it in one way or another. This is a very traditional methodology for the processes in an application. This model is a very linear model that only “flows” one way with a specified sequence of steps that helps users understand each level of the application.

The fact that it is easy to understand is definitely a pro along with it being easy to handle and how much time it saves if done correctly from the start[47]. In saying that, this methodology isn’t great for long term and ongoing projects as if aspects of the project need to be revised or redone the developers essentially must start from scratch. The lifecycle of the project is metaphorically described as the water in a waterfall and how it can only flow down from step to step[47]. This meaning the development is very rigid and linear, not fitting for a project like this as there is no way to revert to previous steps. This could cause huge time issues for the project as it is very constrictive. Due to the demand of the application needing to be changed and altered over time due to the design changes over the lifecycle of development, this methodology was ruled out, but the idea behind it would be fitting if it just had space for this change. Example[47].

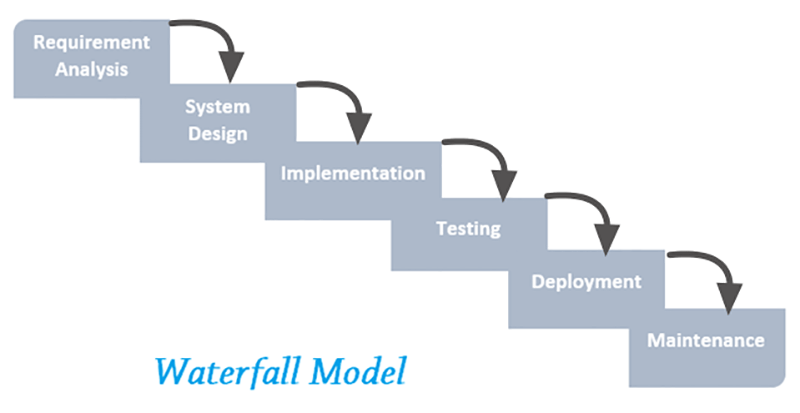


Figure 7: Waterfall Model Diagram

### Iterative Waterfall Model

A new methodology was researched for the progression of the project. Something similar to the waterfall in terms of its usability, but a lot more forgiveness was needed for this project. A way for reverting to other previous steps for improvement of change was very much needed. This led to the iterative model. This is for projects that don’t entirely know exactly what they want the finished product to be and where not all the requirements are outlined and set in stone at the beginning. These can be worked out or changed throughout the process of developing the application.

All this can be done iteratively as suggested in the name of the model[48]. This is basically another form of an Agile methodology but also in the waterfall model format. The development cycle is a lot shorter in this case, but the processes ca continues to occur even after the project has been launched. This could be used in conjunction with the likes of the waterfall model as it is a linear development and an iteration to a different step is allowed once this methodology is applied to the project[48]. This gives the project much more forgiveness in the development stages as it allows the other steps to be revisited. This is what this project needs.

Traditionally in the waterfall model implementation of the code won’t start until the full design of the application is completed[48]. Applying this agile methodology allows you to design specific aspects of the application and then begin programming them and then go back to the design for the next part and so on through each process of the application development. This can also be seen working very well in the testing of the application. It allows you to program parts of the application and then test it, if it doesn’t work go back and fix it[48]. Then it allows you to test other parts of the application without a specific order giving the developer more freedom and forgiveness when it comes to the development of the application.

This would suit the project perfectly as all the requirements and functionality was not determined when beginning the project. This would allow more features and qualities to be added and even taken away if necessary on the go throughout the year. Without having to scrap any new material that may have been developed in the meantime. One drawback to this methodology is the time that it has the potential to consume. It will take extra time to go back and add bits and pieces to the application and then test it and so on. This could consume a lot of the most important resource in this project that is time. Regardless this is the methodology for the project that will be implemented. Example[48].

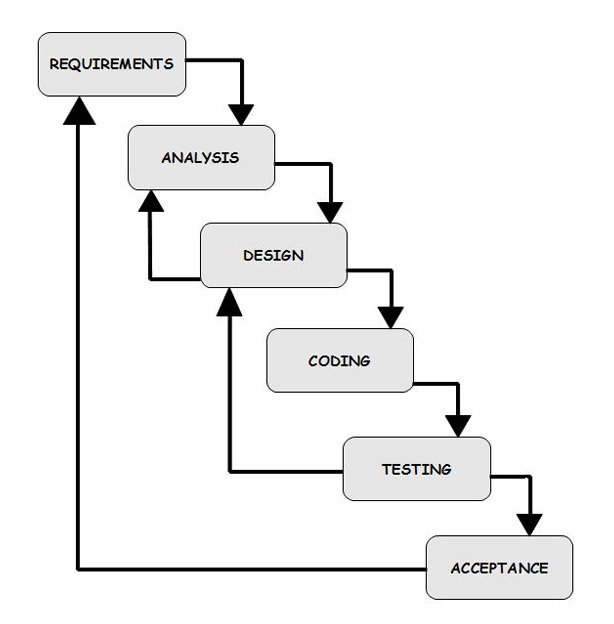


Figure 8: Iterative Waterfall Model Diagram

With the methodology chosen, the stages of design and programming will be completed in phases. The ability to design an aspect of the project then to code it and then test it will be extremely helpful. Once one part if relatively finished, the next part can be started and then it is possible to come back to the other almost finished aspects and complete them. Also, if need be, new requirements can be added by going back to that phase and then they can be designed and implemented. These new features could be large parts of the application as they will be newly discovered, yet they could improve the section of the project greatly.

Ideally each section won’t have new features added after they are finished but in case they are it is goof to have this methodology. You can see this in the way the application has completed different segments already. For example, the first plan of action was to connect to a basic database, which was done. the next task was to add users to the database and allow the user to sign in. Once this was completed work on the design of the login and register pages was done until I was satisfied with how they looked, although they may be subject to change. Then again, going back to a programming aspect of the application after the design which was connecting to the API and pulling the relevant information off it. Then again, a phase of design in terms of how the data received looked on the application.

### Priorities

There are a few very high priorities for this project that need to be addressed and researched before anything. The main priorities are the Payment System, the API, the Machine Learning and the databases. There are sub sections under each of the previous topics that have a variation of priority. For example, Although, the Database is high priority, the sub sections of the database such as a user profile picture or the likes of that, as a medium to low priority, whereas the high priority is the user can register and login with regards to the database, and the app checks and authenticates the user.

Much like the Results and fixtures pages of the application. This was predominantly to connect the front end and the API and to request data. The machine learning stands alone in its priority in terms of this. The final data that will be stored and viewed here will thoroughly thought through, but it has changed from a high priority to a medium priority as the application has connected to the API database.

The machine learning is a huge aspect of the project that holds no subsections. This will add complexity and functionality to the project, and that is one reason it has a very high priority in the project. With the Machine learning being conducted in Python, the dataset retrieved online and with the algorithm that will be used known already the programming of it just needs to be worked on. Regardless, this remains a high priority in the project.

All of this shows how that the priorities of the application are dynamic and ever changing. This is important to the project as it fits exceptionally well with the methodology and how it is iteratively. The iterative nature, iterative model and dynamic priorities all converge and really fit together.

## Application Architecture

### General Architecture

The applications architecture is very important in software development as it structures every detail about the application such as the screens, the activities, the extra classes and activities components. it also outlines how they are all going to fit together as one project. These techniques and patterns used and reused throughout the application are used to fully develop the structure of the mobile application.

Many layers need to be looked at when doing this. Mainly for an application like this you can focus on 2 layers. These being the presentation layer which “presents” the application to the user and is the overall appearance of the application and then the Data layer, this being the layer that acts with the data needed for the application such as the data gotten from a database of from an API. Some major things to look at are the specification of the phones using the application and will the application hold up to any devices lacking in certain departments like CPU power, storage space or RAM. Another big concern is the different bandwidths that the different devices may have, and once again will the application hold up to this? Predominantly for this section though we will be looking at the User interface and how it looks and is navigated.

### Activity Architecture

The activities in the application are as follows: Login, Register, Pin verification, user Area (with 4 fragments), match facts, fixture predictions, user’s prediction. The user will open the application and will have 2 options. They can Register, opening the register page or they can log in. if they choose to register they will be brought to a new page to which they must fill out a form regarding their personal information and their banking details. If they enter valid Ibans then the user will be asked to verify that the Iban is theirs with a pin verification page. They get a certain number of attempts here and if they’re successful then they have created an account.

Once an account has been created the user is free to log in. they will be brought to the home fragment in the user area and then are free to navigate using the navigation bar at the bottom. They can navigate to previous matches where they can see the last 2 weeks’ worth of matches. They can select a match which will display that matches statistics, this giving a good indication of how the match was sided. They can also navigate to the fixtures fragment to where they will see the next 2 weeks’ worth of matches. Here they select a match and a page is brought up where they can predict the score of a match and put a price on it. Here the machine learning prediction of which team will win the match and what the probability of each outcome happening is also displayed to the user to help them in their prediction.

Now once they go to the home page they will see all the predictions that they have set in a list. They can select one and it will open another page displaying a pie chart of the machine learning prediction and other details about their prediction. Lastly, they can go to the profile page and here they can change all the personal data that the application saves, for example, email, Ibans and phone number etc. here they also have the option to sign out of the application. This is a very basic description of the screens in the application giving an outline of the architecture of the mobile application.

## Screen Flow

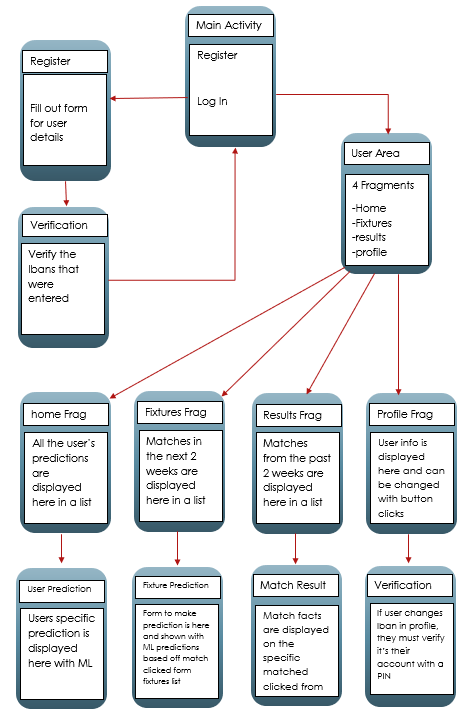


Figure 9: Application screen Flow

## Main Screens Demonstration

### Overview

This section will briefly go over each screen in the application and will highlight the different design features of the screen. Overall for the application, the aim was to make it look as minimal as possible, so it is not confusing or over whelming for a new or existing user. In other words, each screen has aa unique function to itself and none of the screens are clustered. This insures the user is not overwhelmed at any section of the application and can use it correctly, but in saying this it in no way takes away from the functionality or complexity of the application. On each screen the background theme is a dark gradient from blue to navy to black. This gives the application an overall sleek look and is also very easy on the eyes.

The text throughout the application is either white or greyish colour. Both contrast to the dark background and means all details are very to read. All screens, excluding one, have a banner on the top of the screen either indicating what page in the application the user is on or else is displaying information from the list item they clicked on the page previous, this indicating the position in the list they clicked and indicating what teams are involved in the screen. E.g. “Arsenal v Tottenham”. The banner on each page also contains an English Premier League logo to indicate and reiterate what the application is about.

### Register

The main Screen is simple, it is the first screen the user sees when opening the application. It is very basic in design as it only contains the option to log in or the option to resister for an account. The logo of the project is above the log in area, displayed in white for a good design. This task is easy, but the issue is coming up with a logo.

Also displayed below is the Register page. Here also showing the simple yet ideal design. The user is faced with a form on this page to which they must complete in order to create an account. This for contains the normal information that you should be entering when creating any account. It contains the likes of name, date of Birth, email, phone number and so on. As seen below it also contains two text boxes to fill out for Ibans of the user. In the first box the user entered their current account Iban. In the second box they enter their savings account Iban. These are what uniquely identify the account to which is sending or receiving money to or from the other. Once all the information here is entered correctly, the verification screen will open.

Here the user is asked to enter the PIN for their 2 accounts to ensure the account belongs to that person. This is just standard security for the application as it is using money and bank accounts. If anything cannot fit on the screen it is in a scroll view These three screens are as follows.

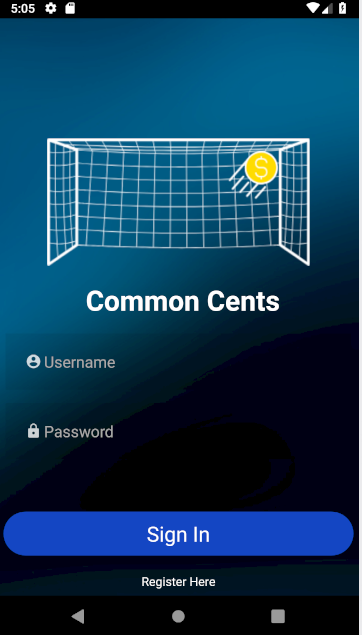
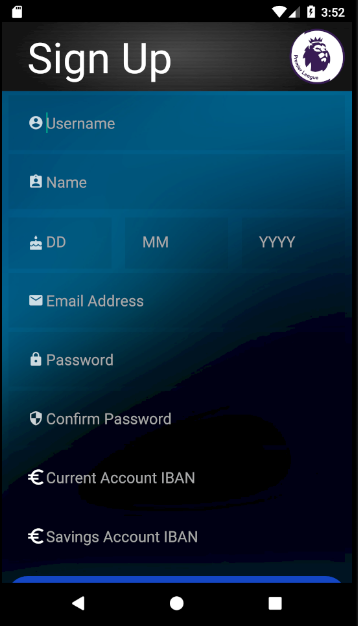
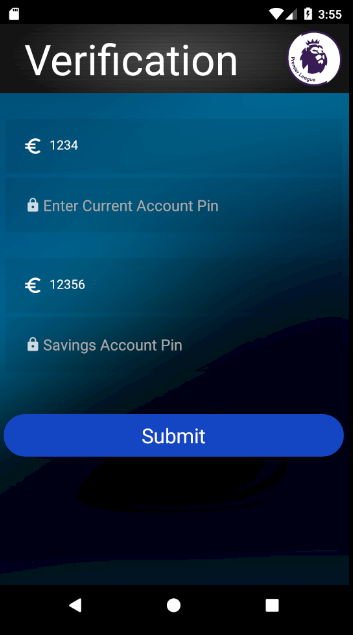
  

Figure 10: Home Activity Figure 11: Register Activity Figure 12: Verification Activity

### User Area

The Next three screens are the 3 main screens in the application. Once you have logged in you will come to these screens. These screens are the fixtures, results and home screens. They are all fragments of the larger user area activity. The design of these screens, and purposely so, are all similar. This gives a reoccurring theme to the application. All are basically list views, displaying different information in a list based on which fragment you are in.

For example, if you’re on the fixtures fragment you will be displayed a list of all the upcoming fixtures for the next two weeks. The user can scroll through this list to see all the matches. The same idea for the results list, it displays all the previous results going back 2 weeks. This keeps the app up to date, so it doesn’t slow down retrieving more matches. The results list harbours the teams playing the match and the date and score of the match, whereas the fixtures list displays the team names like the results list but the date and time of the match.

Finally, there is the home fragment which displays to the user all the predictions they have active on the server. It will display all the information the server has on the prediction. Each row in these lists are custom rows designed specifically for this application. Each one has unique information, but they are all similar and that reiterates the theme throughout the application. To navigate through these different fragments there is a custom navigation bar at the bottom of the activity. Clicking on each option will enlarge that specific image and it will queue a custom animation for the current fragment to leave the screen and for the next fragment to come onto the screen. These three screens are as follows below.

Figure 13: Home Fragment Figure 14: Fixtures Fragment Figure 15: Results Fragment

### Results Information – Results List Clicked

Each Item in all three of the lists is clickable. Depending on which list item is clicked a different screen will be shown to the user. Once you click an item in the results list a Results Info activity will be displayed. On this screen, the user is shown details of the match that they clicked on. They will be showing an in dept display of the match facts. They user will be shown the score of the match, the possession each side had, the shots on target that each team had, the fouls, the corners and offsides. Along with each statistic, a graph will be displayed to the user giving them a visual representation of the percentage of each statistic for each team. The crest of each club is also displayed at the top of the screen to aid with the visuals in the interface. That page looks the next image.

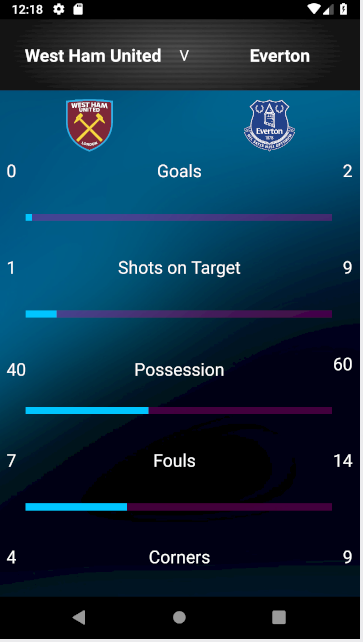
 

Figure 16: Results info example 1 Figure 17: Results Info example 2

### Fixture information – Fixture List Clicked

The next page to be looked at after clicking an item in a list is the is the Fixture Info activity. This is displayed after clicking an item in the fixture list. The main purpose of this activity is for the user to set the prediction that want on that specific match that they have selected form the list. The percentage of each outcome, calculated by the Machine Learning program, is also displayed here for the user. The idea behind this is to help the user get an idea of how the match is going to go and give an idea of which team is going to be dominant based on the home team advantage and how each team has been playing home and away throughout the season. This is displayed at the top of the activity along with the teams and their crests. Home team win, away team win and draw percentages are all shown to the user.

Below that is the area that the user can set the prediction they want. They have a home team score, away team score and an amount area to set. They must set all parameters here on what they think that outcome of the match is going to be. Then they click the button below that to set the prediction to Firebase. Once this score prediction is set to Firebase, it will appear in the list titled “Your Predictions” on the home fragment. The fixture Info activity looks like the following.

Figure 18: Fixture Info example 1 Figure 19: Fixture Info example 2

### Event Information – Event List Clicked

The Last page to be looked at after clicking an item in a list is the Event Info activity that is displayed to the user after clicking an item in the list on “Your predictions” home page. Once you click into one of these items, much like the other lists the home and away team names are displayed. Below that you will find a Pie chart. This Pie chart is a visual representation of the predictions on that given match as an outcome of the Machine Learning Model.

This Pie chart is interactive. You can click on one of the three options that are displayed to you. Blue is the home team, Pink is the away team and grey is a draw. A title if each section of the pie chart is displayed so the user knows which represents each team. Once you click on one of the sections, the area below will be altered based on which section you click. The section below shows the crest and the percentage that team has of winning. It changes when you click different section.

Below all that again, you have the details on the prediction that the user has set. The date and time of the match is also displayed along with the score prediction of each team in bold, the crests of the teams are also displayed beside the score, again to aid in the visual representation of the application. Then finally just below that you will find the delete button which allows the user to delete the specific prediction that they are on in the list. This removes the prediction from firebase and refreshes the home list a confirmation fragment asking the user if they are sure they want to delete it.

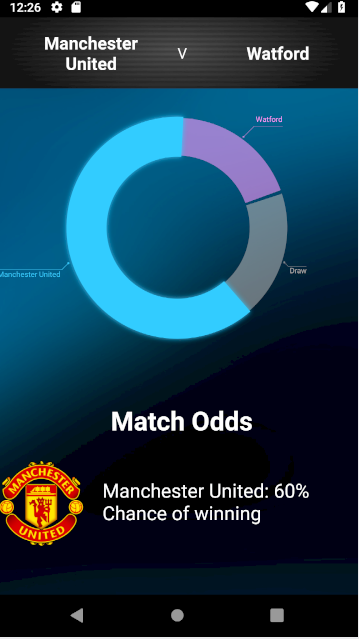
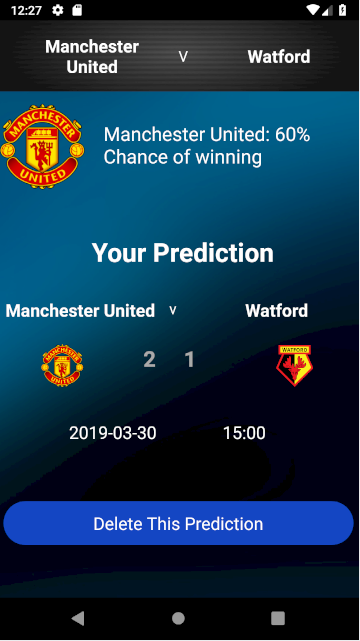
 

Figure 20: Event Info example 1 Figure 21: Event Info example 2

### Profile and Information Change

The Final pages of the application that needed design were the profile activity. There are 3 areas of data that the user can change. The first area is the personal information. Here the user can change their Name, their username (if it’s not taken), and their email. The current data for each of the fields is displayed when they navigate to this activity. If they want to change it then they must alter the text. The check will occur and if the fields are valid and not in the database they will be updated in Firebase.

The next area of data they can change is the Bank account area. Here they can enter new Ibans if necessary. If new Ibans are entered and the button is clicked, then a similar verification page from the register activity will appear to which they must again verify that the accounts are theirs. If the PINs don’t match 5 times, then they cannot verify the account. If they can successfully verify the accounts, then the accounts on firebase will be updated on profile and on the events that they have set.

Lastly on the profile page the user can change their password for their account on the application. For verification of the user they must enter their old password and it must be correct, if that password is correct and that the new password matches the confirm new password, the password will be updated on firebase, to which they must now use that new password to log in.

All the buttons have an alert that pops up that the user must agree to change the data they are changing to ensure it is not done accidently.

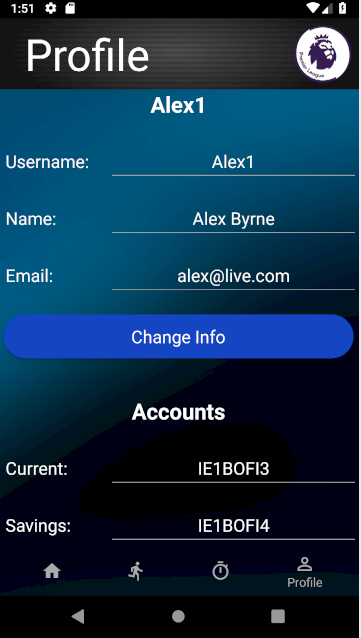
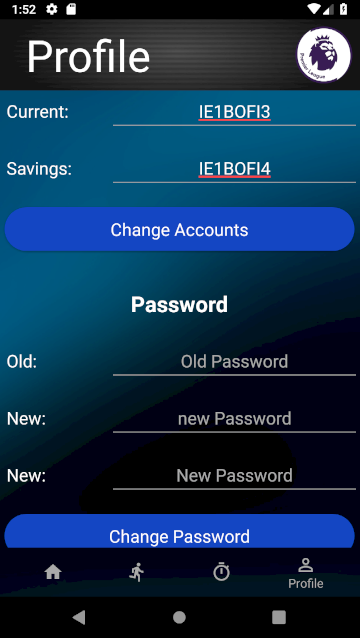
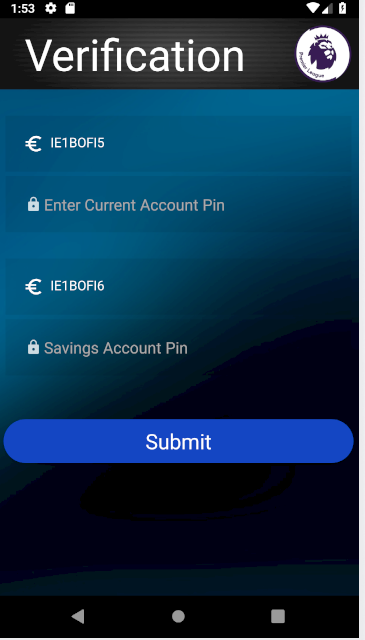
  

Figure 22: Profile example 1 Figure 23: Profile example 2 Figure 24: Profile example 3

## System Architecture

### Overview

In this section, the style of architecture will be reviewed, and the possible types will be discussed. What is meant by system architecture is the concept model that officially defines the behaviours, the processes, the structures and the views of a software system. All systems have different component structures and this section defines how each component in the Structure is linked to each other in the grand scheme of the project.

### Pipe – Filter Architecture

Initially, the pattern looked at for the System architecture was the “Pipe - Filter pattern.” This consists of four components in the system, which are the “pump”, the “sink”, the “pipe” and the “filter” [48]. This mainly consists of many “filters” that transform and alter the data that is passed into them by the “pump” and are on their way to the “Sink”. The pump is the beginning of the model[48]. This is the producer of the data that is being used throughout the model. This can be many different things, such as an API, a text file, keyboard input, remote databases etc.

The next aspect of the model is the pipes. These are the components of the model that transfer the data from one area to the next[49]. It can transfer data from filter to filter, pump to filter or filter to sink. This is a unidirectional stream of data. It is “usually implemented by a buffer to store all data[49].” This allows the next filter or the sink to have time to process all the data properly.

The next aspect is the filters. This is the section of the model that controls, alters and transforms the data to be passed on by the next pipe[49]. It takes the data from the previous pipe and applies the mapping of the data that is was programmed to do. it can drop data, compute data or any other operations like this. A filter can have any number of input or output pipes attached to it[49]. On the whole model at any one time, it can have numerous filters working with the data.

This type of architecture can be seen in many UNIX programs and also many compilers[49]. The popularity of this architecture escalated from the UNIX operating system. The creator of UNIX, Ken Thomson, and Dennis Ritchie “decided to Limit the architecture to a linear pipeline[49].” This sort of architecture should be considered when you have a lot of transformations and alterations of the data that is being used in the application. Flexibility is a huge necessity when using this architecture.

There are a few main problems when it comes to this architecture. Deadlock is a large problem[49]. This occurs if a filter must receive all the data for the operation to be completed but the data buffer from the pipe may be limited in terms of size and cannot fit all the data needed for the filter. The buffer then may overflow resulting in a deadlock[49].

Another big problem is in the way some filters may only allow one datatype[49]. If this is the case, then a lot of parsing may be needed. Parsing may be needed going into the filter and leaving the filter to the next pipe. This adds a huge amount of complexity to the application which is will result in the slowing down of the data processing[49]. Also, if you create different pipes to deal with different datatypes you cannot link any pipe to any filter. Due to these issues, the Pipe – Filter architecture would not be suitable for this project. The main operations don’t transform data often which is the main issue but also the other problems mentioned are a big concern. Example from [50].



Figure 25: Pipe - Filter Architecture

### Client Server Architecture

This type of architecture consists or two main aspects, that are the Client and the server. The server waits for requests from clients[51]. A client is a device that is being used by the user. For example, a phone is a client and a remote database is the server. Once a request is received by the server, it processes the request it was sent from the client and the gathers the data that is relevant to the request and sends back a response of this data to the client[51]. The Client then processes that data in the appropriate was such as parsing the data and then does what it programmed to do with that data.

A server can be either stateful or stateless. A stateful server keeps state between connections where as a stateless server does not. Essentially, a stateful server will receive and store data whereas a stateless server will not, and they require backing storage. Stateful servers would be the likes of an SQL server or like Firebase.

This is sort of architecture was needed in the project as I have not one but 2 servers. One is for the users and other information, while there is a second server for the payment system replicating a bank. Therefore, in saying this, a Client – Server model is somewhat needed.

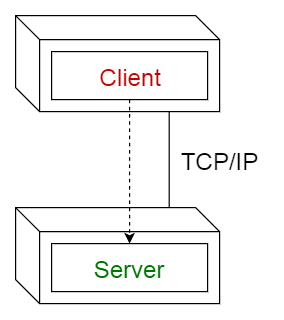


Figure 26: Client - Server Architecture

### Model View Controller Architecture

The “Model View Controller” (MVC) style of architecture is a Three-way factoring of a system. The three main parts are the model, the view and the controller[52]. All three have different functions in the application. These are the three classes that control the different aspects of the system.

The “Model” is the section that contains the core functionality of the application, it also stores the local data that the application will be using throughout its use[52]. The “View” is the section of the application that that displays the information from the model in some form of user interface for the user to see all the necessary data that the application is storing[52]. There can, and usually is, many different views defined in the application for the user to see. The “Controller” is the section of the application that reads in and “controls” the input from the user(51). This User input is handled, and processes are done to the data stored in the model based on the input that is read in. This is a strong pattern as it separates the internal functions of the application into different parts so there is no overlap and confusion.

This project would suit the model view controller system architecture very well as creating an application on a mobile device is essentially the methodology already. You get data from a source and it is put into the model, then the data is processed by the controller and finally displayed in a graphical user interface for the user. Therefore, it suits this project perfectly.

### Final System Architecture

Ultimately, the project has two main system architecture methodologies. The application aspect of the project needs the Model View Controller architecture for the mobile application itself that the user will be interacting with. This is needed to process all the data and display it to the user in a user-friendly fashion to ensure the usability of the application is good.

It has also adapted the Client Server pattern. This can be seen with the fact that it has a database storing user information and usernames used and a front-end application that remotely connects to the database. The Client Server pattern consists of both client and server(51). There can be many clients(users). The server side of the system will provide services to the client, such as log in credentials like the application(51). Clients can read and write to the server and the server responds to client writing by accepting the input and putting it into the database and then responds to a reading by giving the client access to the relevant information(51). Both architecture patterns coming together is fantastic for the application as it utilizes the MVC pattern hugely with the user input being a huge factor of the application but also the Client Server pattern as the user must sign into the application and this provides a great way of doing this remotely.

The project also needs the Client – Server architecture in the way that it contacts the mock bank database for the PINs of the bank accounts. This too is the contacting of the client, being the mobile application, to the server, being the bank database. This too will be remotely and therefore a Client - Server system.

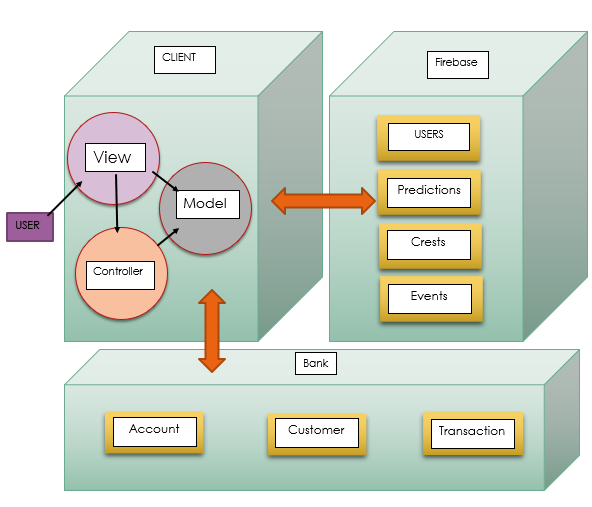


Figure 27: Common Cents System Architecture

## Use Cases

### Register

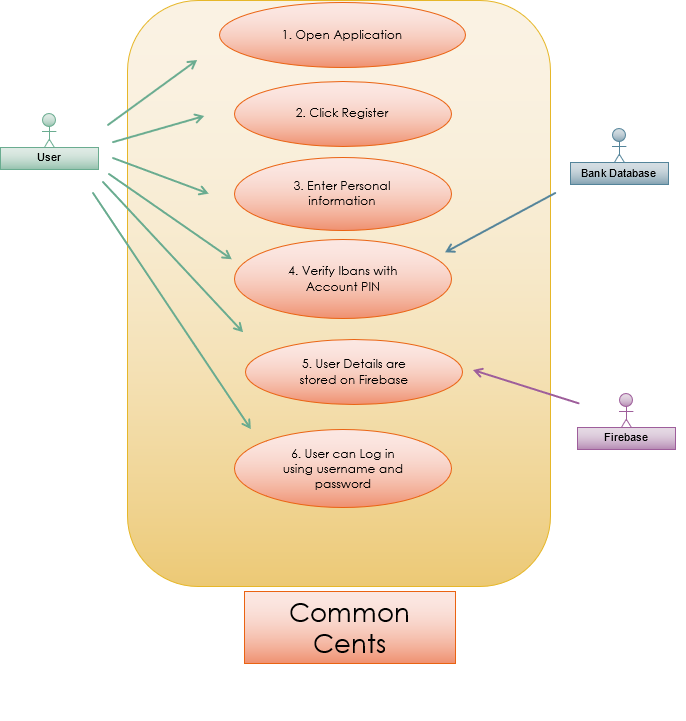


Figure 28: Use Case 1 - Register

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case 1** | | Register | |
| **Description of Goal in Context** | | This is how the user uses the application to create an account on Firebase, so they can use the application | |
| **Preconditions** | | User has successfully downloaded the application to their phone | |
| **Post conditions, Success End Condition** | | User can Log in and use the full capabilities of the application. | |
| **DESCRIPTION** | | When the application is opened the user can click “Register here” and a new activity will open to where they enter their personal information and banking details. A new activity is opened then to verify their bank accounts. Once they verify their accounts they can log in successfully. | |
| **Main Flow** | | | |
| **Step** | **Action** | | **Alternate** |
| 1.1 | Open Application | |  |
| 1.2 | User Selects “Register Here” | | User Logs in |
| 1.3 | Register screen loads | | User area of app loads |
| 1.4 | User enters information for new account | | User doesn’t enter details to make an account |
| 1.5 | Verification page loads | |  |
| 1.6 | User verifies their accounts with PIN | | User cannot verify their PIN, activity closes after X attempts |
| 1.7 | Register and verification page close | |  |
| 1.8 | User can Sign in | |  |
| 1.9 | **End Use Case** | |  |
|  | | | |
| **EXCEPTIONS ERRORS** | | | |
| **Step** | | **Branching Action** | **Alternate** |
| 1.2.1 | | -user doesn’t want to create account  - back button there to go back to log in screen |  |
| 1.4.1 | | -invalid data entered for sign up screen  - prompts indicating what information is invalid | - verification won’t open |
| 1.7.1 | | -user enters wrong log in data  -cannot log into an invalid account |  |
|  | | | |
| **ALTERNATIVE VARIATIONS** | | | |
| **Step** | | **Branching Action** | **Alternate** |
|  | | No alternate branches |  |

|  |  |  |
| --- | --- | --- |
| **RELATED INFORMATION** | Use Case 1 | Register |
| **Priority:** | Very High | |
| **Performance** | 2 minutes to enter information | |
| **Frequency** | Once off use for each account | |
| **OPEN ISSUES** | - Entering wrong inputs  - Failed Verification | |
| **Due Date** | Release 1.01 | |

### Using Application

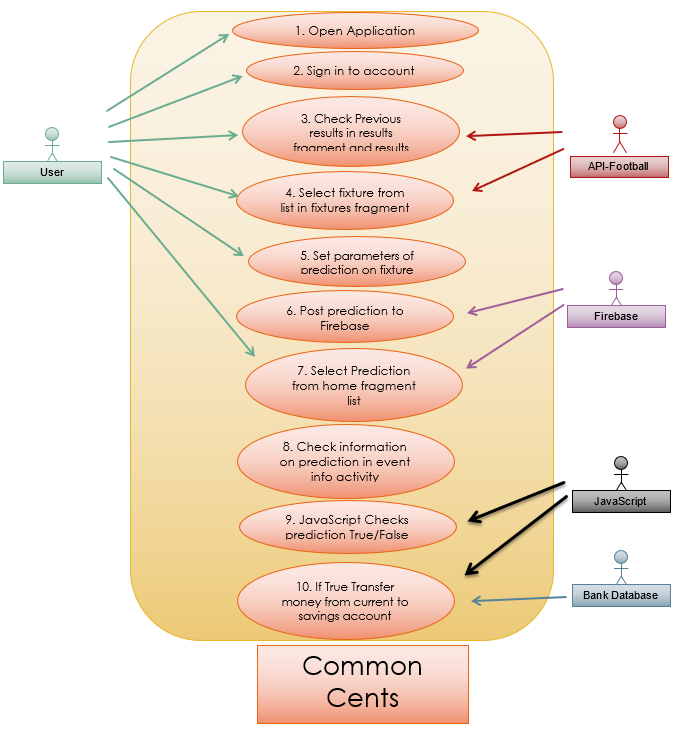


Figure 29: Use Case 2 - Full Application

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case 2** | | Using the Application | |
| **Description of Goal in Context** | | This is the general use of the application and a broad description of each activity and fragment. | |
| **Preconditions** | | User Has successfully created an account on the database | |
| **Post conditions, Success End Condition** | | User will successfully have utilized the functionality of the application | |
| **DESCRIPTION** | | When the user has logged in they have access to the 4 fragments. They’ll mainly use the results to check the previous matches, the fixtures for the coming matches and the home fragment to see the predictions they set. They set the prediction after selecting a fixture in the fixture list then sets the prediction in a new activity. This prediction now appears in the list at home to which they can click on and see more details about that prediction. | |
| **Main Flow** | | | |
| **Step** | **Action** | | **Alternate** |
| 1.1 | Open Application | |  |
| 1.2 | Log in to account | | Register new account |
| 1.3 | Home screen is opened | |  |
| 1.4 | User can navigate results fragment | | User can select another fragment |
| 1.5 | User can select a result and see more info on it in a new activity | | User can select another fragment |
| 1.6 | User can navigate to fixtures list and select one. | | User can select another fragment |
| 1.7 | New activity is opened, and they can set their prediction in this activity | | User can select another fragment |
| 1.8 | User presses button to post the prediction to the database | |  |
| 1.9 | This prediction will now show up on the home page. | | User can select another fragment |
| 1.10 | User can select this prediction | | User can select another fragment |
| 1.11 | More info on this prediction will be displayed. E.g. Machine learning predictions | | User Can Delete prediction |
| 1.12 | JavaScript Checks if prediction true | | prediction may not have occurred |
| 1.14 | If true, the given amount from the prediction is sent from the current account to the savings account | | If false, no money is transferred, prediction is deleted |
| 1.15 | **End Use Case** | |  |
|  | | | |
| **EXCEPTIONS ERRORS** | | | |
| **Step** | | **Branching Action** | **Alternate** |
| 1.4.1 | | -information doesn’t load  -error handling the user can change fragment to refresh |  |
| 1.6.1 | | -fixtures and results can’t load?  - will continue to try | -Application may reload |
| 1.12.1 | | - cannot post new info to database?  - error- cannot connect. |  |
|  | | | |
| **ALTERNATIVE VARIATIONS** | | | |
| **Step** | | **Branching Action** | **Alternate** |
|  | | No alternate branches |  |

|  |  |  |
| --- | --- | --- |
| **RELATED INFORMATION** | Use Case 2 | Using Application |
| **Priority:** | Very high | |
| **Performance** | 5 minutes to navigate entire app and look at information you want | |
| **Frequency** | Every use | |
| **OPEN ISSUES** | Not connecting to the database or API or payments | |
| **Due Date** | Release 1.01 | |

### Machine Learning

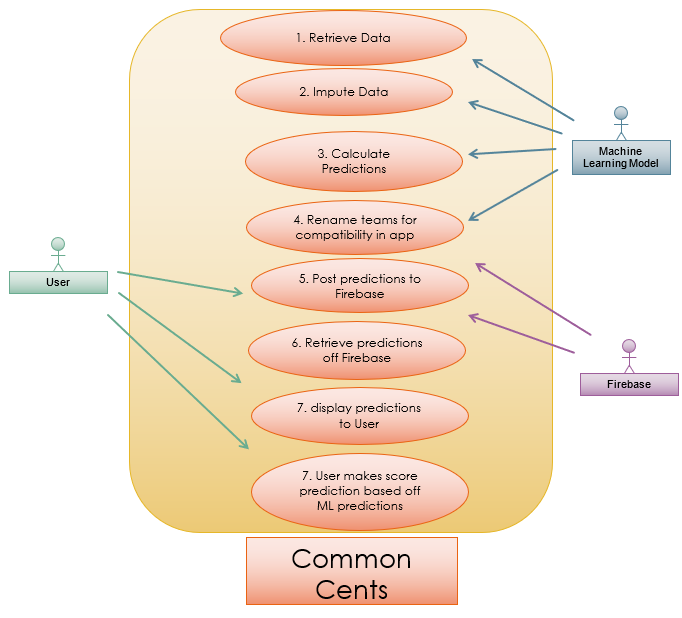


Figure 30: Use Case 3 - Machine Learning

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case 1** | | Machine Learning | |
| **Description of Goal in Context** | | This is the Machine Learning Model Use Case and what steps it takes for the predictions to get to the user | |
| **Preconditions** | | There is Data is the Dataset | |
| **Post conditions, Success End Condition** | | User has the predictions displayed to them and uses them to make the prediction events of their own to save money. | |
| **DESCRIPTION** | | The Machine Learning Algorithm takes the data from the dataset and imputes unwanted data, then creates the predictions using Poisson regression. Next it renames the teams, so they match with the application and posts the data to firebase. From here it’s then displayed to the user in the app to be used to inform the user. | |
| **Main Flow** | | | |
| **Step** | **Action** | | **Alternate** |
| 1.1 | Python gets the data from the online source and puts it into a dataset | |  |
| 1.2 | The data that is unwanted is dropped from the dataset. | |  |
| 1.3 | The prediction is made using Poisson Regression | |  |
| 1.4 | The team names in the dataset are changed so they can be compatible with the application and the API team names. | |  |
| 1.5 | The predictions are posted to Firebase | |  |
| 1.6 | The predictions are retrieved off firebase for use in the client application | |  |
| 1.7 | The application displays the predictions to the user | |  |
| 1.8 | The user makes a guess on the outcome of the match based on the machine learning predictions | | The user doesn’t use the predictions to make a guess on the outcome |
| 1.9 | **End Use Case** | |  |
|  | | | |
|  | | | |
| **EXCEPTIONS ERRORS** | | | |
| **Step** | | **Branching Action** | **Alternate** |
| 1.211 | | -Cannot retrieve data from online source  - Must try again as it is needed |  |
| 1.5.1 | | -data cannot be posted to Firebase  - model must try again |  |
| 1.6.1 | | -data be retrieved from firebase  -this is needed |  |
|  | | | |
| **ALTERNATIVE VARIATIONS** | | | |
| **Step** | | **Branching Action** | **Alternate** |
|  | | No alternate branches |  |

|  |  |  |
| --- | --- | --- |
| **RELATED INFORMATION** | Use Case 3 | Machine Learning |
| **Priority:** | High | |
| **Performance** | 2 minutes to run full application | |
| **Frequency** | Run once every few matches to update predictions | |
| **OPEN ISSUES** | - Internet connection errors will make app fail | |
| **Due Date** | Release 1.01 | |

## Flow Chart: Processes in Application



Figure 31: Full Application Flow Chart

This is a simple Flow chart depicting the overall usability and functionality of the application. It starts, like with most applications, with the user. Does the user have an account? If not, create an account and loop back to has account. If the user has an account, then they sign in to the application. Here the user can decide to make an event. If the user wants to make an event, they can check the fixtures of the coming matches. Then they can select one of the fixtures that are coming up within the given time period. Then they set up the event and the payment, fir if event is true, the if the event occurs the transfer will be made to the other account. If the event does not occur, then the money will not occur also. If the user doesn’t want to make an event, they get the option to check open events. They can check an event, or they can sign out of the application.

## Sequence Diagram

### Main Application Sequences

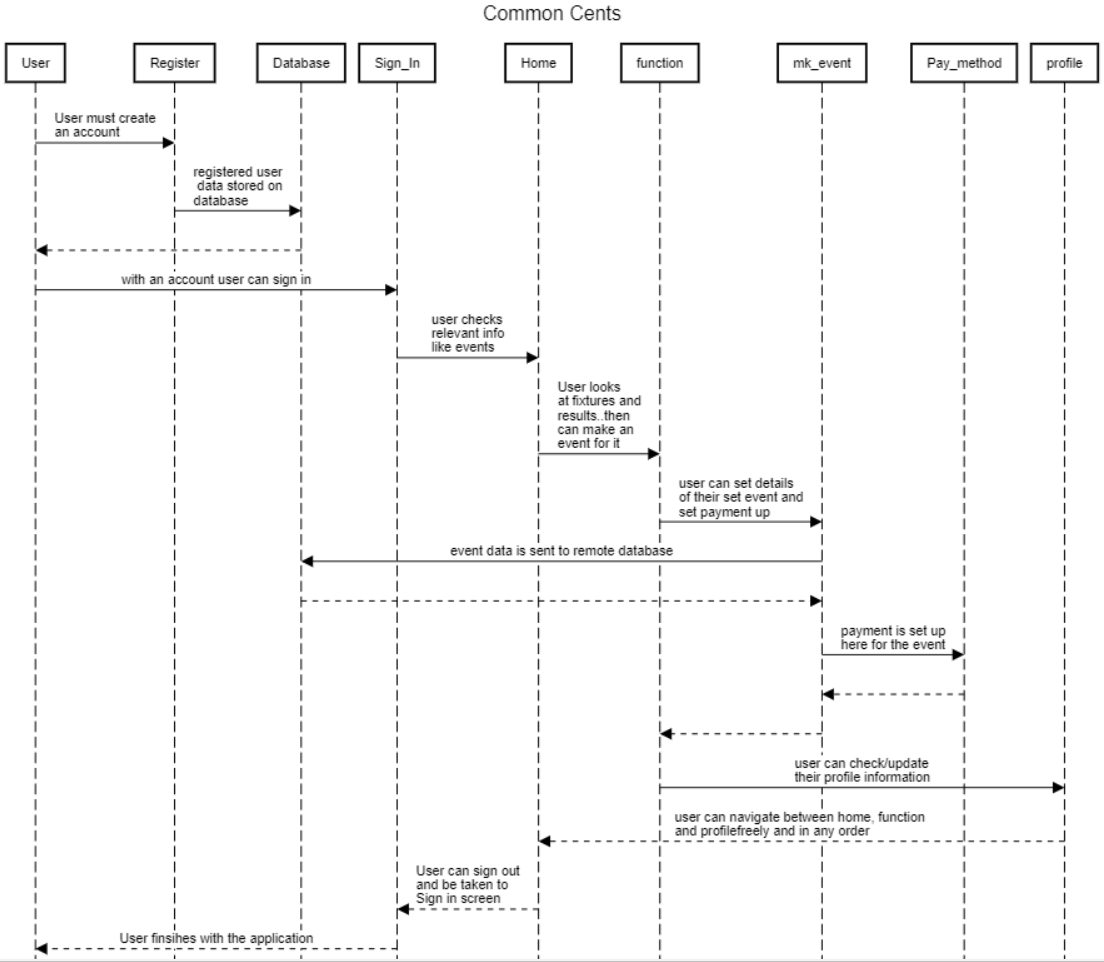


Figure 32: Full Application Sequence Diagram

This is a simple sequence diagram explaining the different processes inside the application and explains the different steps that the user must do in order to each bit of functionality of the application such as a payment or checking an event on the Home screen. It also explains the processes of for example the register process and it linking to the database where it saves your information. Then to sign in it gets your entered details off the screen and cross references it with the database. If the credentials match, then access will be granted to the rest of the application.

### JavaScript Prediction Checking

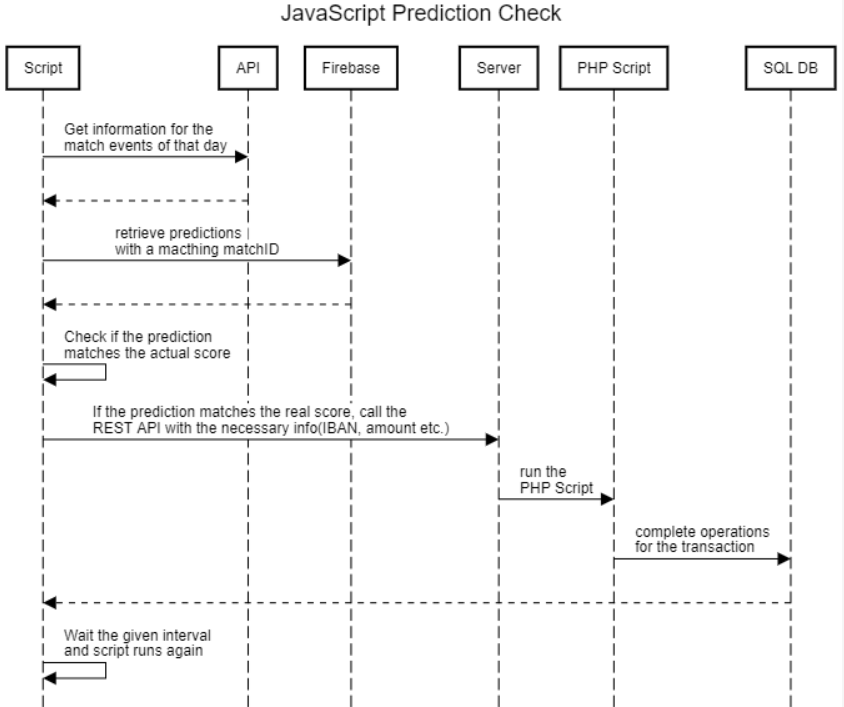


Figure 33: JavaScript Check Script Sequence Diagram

This sequence diagram represents the JavaScript function that checks the user’s predictions with the actual scores from the API. If the predictions are true then the JS script will contact the Heroku server, contacting the PHP scripts that conduct the transaction on the SQL Bank Database. This script is programmed to run once a day and can be changed to run more than that if need be. The top headings are all the entities involved in the checking and updating accounts process.

# Implementation & Development

## Overview

This Section will cover all the main areas of how each area in the project was programmed and implemented, providing a detailed description of each of the important sections. All the main areas are covered in this section. The different sections are the android application itself, the Machine Learning in Python, the Databases, the REST APIs and servers.

## Databases

### Bank Database

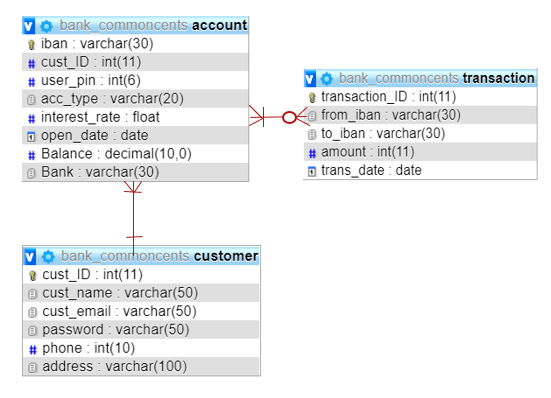


Figure 34: Bank Database ERD

This is the design of the ERD model for the mock bank example. I have the necessary components for bank transactions. The tables are for customer, account and transaction. They are all connect differently as they are all related to each other in a different fashion.

The customer table has a “one to many” relationship. This means that there must be one customer for every bank account, but also that one customer can have multiple accounts. In this instance a customer cannot have zero accounts as they are needed to be a customer.

The Account and transactions tables have a “many to many” relationship. This means there can be numerous transactions for each account, but also that there are many accounts per every transaction. It is this way due to an account will have to make many transactions throughout its existence with the bank, but a bank account can also have no transactions. Also, every transaction must have two accounts related to it due to every transaction needing both a receiving account and a sending account. In the case of this it is mainly current and savings accounts.

Each table has the necessary information needed. The customer table holds the personal information for each customer, such as; email, name phone etc. and is uniquely identified by the cust\_ID. The transaction table hold data such as each transaction amount, the receiving and sending accounts and the date, this table is uniquely identified by transaction\_ID. Finally, the account table hold information such as the customer ID of who owns the account, the Iban of the account, the PIN, the type of account, the balance etc. and is uniquely identified by Iban as each account has a unique Iban.

### Firebase Database

The Firebase database Stores data for each user. This would be the equivalent to a table In an SQL database. This JSON object stores the user information such as the username the date of birth, the userID, their 2 Ibans for their bank accounts etc. The userID will be used throughout the application to check other parts of the database such as the events array. It looks like this:

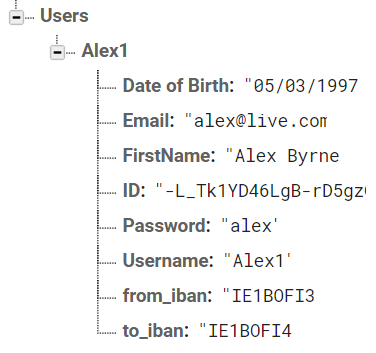


Figure 35: Firebase - User Info Storage

The event JSON Array stores all the event predictions that the user has set. Each object in this array stores the scores of each team playing, the names of the teams, the match date and time, the to and from Ibans to which the money is going to be withdrawn and deposited into. Each object in the array has a unique identifier that is the macthID received from the API. Also, each of those is identified by the userID from the user JSON array. It looks like this:

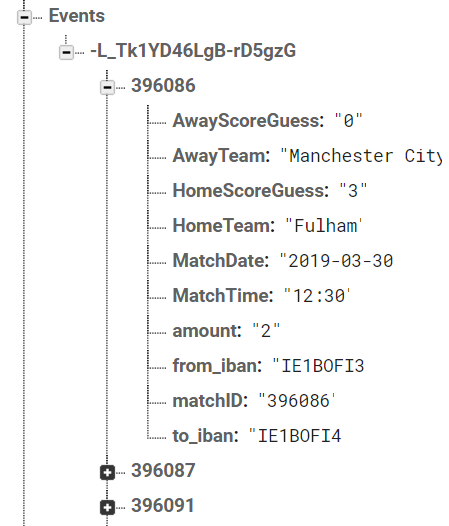


Figure 36: Firebase - Event Storage

The next JSON array in Firebase for this project is for the Machine Learning predictions. Under the predictions key there are all the predictions for all the matches that year whether they have occurred or not. Every time the Machine learning program is run these get updated based on the new data from the program. Each Object in this array contains both the away and the home teams name, it also contains the probability of each of the outcomes. This way all the data can be called from the client application if it is called and displayed in the app. To get a percentage of the outcomes, the app just multiplies the probability by 100 and rounds to the nearest integer. That JSON array looks like the following:

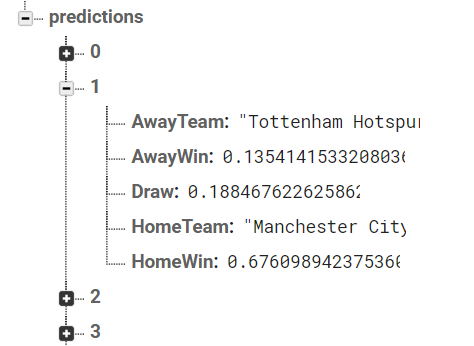


Figure 37: Firebase - ML Predictions Storage

The Final JSON array in the Firebase database is the crest array. This is for use in the application for displaying the teams crest in the application. This array holds objects that contain the key of the Teams name e.g. arsenal, and a URL as the value. This URL is a link to a .png file that is stored on the internet. Each of the .png files holds the corresponding teams crest with no background. This ensures the User Interface looks as nice as possible. That part of the database looks like the following



Figure 38: Firebase - Team Crest storage

## Servers

### Heroku & PHP Scripts

Heroku is used as the server for this project. This is a Remote server, it stores all the PHP files that are needed to make operations on the Bank Database. These files are kept on the Heroku Domain. The v1 folder contains the two PHP scripts that call the classes inside the includes folder. The transaction script is the script that is called by the JavaScript check script that checks the users guesses. The verify script is called by volley in the android application itself, it is used to verify the users PIN in the application.

The transaction script has four parameters passed into it when the “post()” method is called from the JavaScript. These parameters are, the two Ibans involved in the transaction, the amount the transaction is and the date. Once the transaction.php is called, if the parameters are set it calls the “createTransaction” method from the DbOperations file in the “includes” folder. It also calls update the accounts methods to ensure that both accounts are updated correctly, the current account has money withdrawn and the savings account has money deposited in. The DbOperations methods execute an SQL query to update the tables and they all must complete, if one fails they all fail. The parameters are passed into the SQL query and the tables are updated.

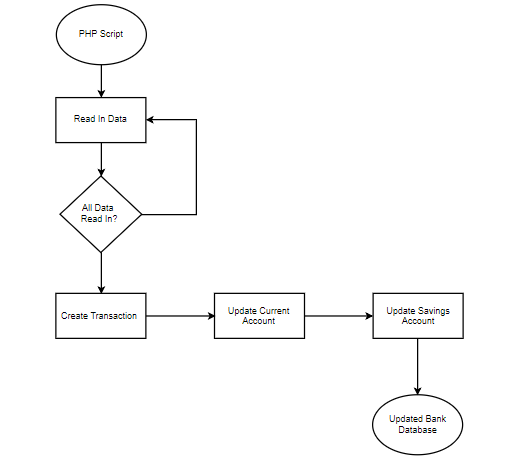


Figure 39: Transaction.php Flow Chart

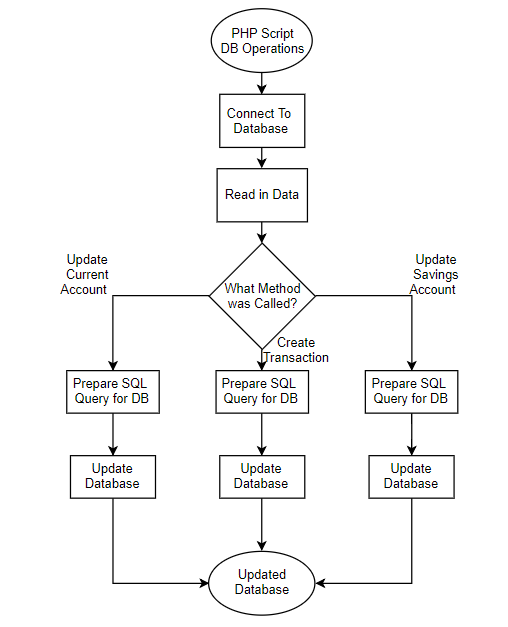


Figure 40: DBOperations.php Flow Chart

The verify script is called from the Android application in a Volley request. This calls the “GET” method from the server passing in the Iban set in the application. This returns the PIN from that set Iban account. The user must have the correct pin to be allowed access to this PIN. The PHP script reads the Iban and gets the corresponding PIN and returns it in the volley request.

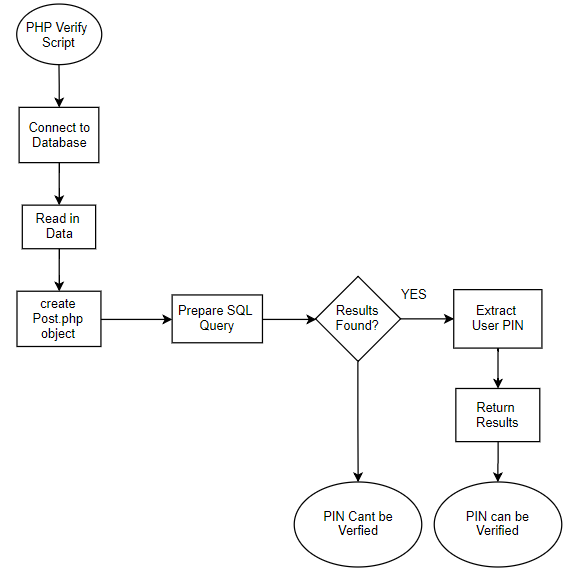


Figure 41: Verify.php Flow Chart

### JavaScript

The JavaScript file is the checking if the user’s predictions are true aspect of the project. Here quite a few things happen. Firstly, a connection with Firebase was made to ensure the users predictions could be used in the script. The second step in the script is to retrieve todays date using the built in JavaScript library “today.getDate()”, this gets an instance of today’s date and stores it in a date object. The date is then formatted to the correct format for the API call. Like the image below.

Next there is a function that is run a certain amount of times a day. At the moment, the function is run four times a day, represented by the “21600000” value in the “setInterval()” method. In this method a for loop retrieving all the data off Firebase in the Events array, which then calls the “compare()” method, passing in all the relevant data from Firebase, such as “guessMatchID”, “guessHomeScore”, “guessAwayScore”, the Ibans etc. The image below.

In the “compare()” method, the API calls the fetch request calling for all the matches that occurred today, using the date from above. The data received here is then parsed into variables such as the match ID and the scores. Then an IF statement checking the “MatchID” is called and if thy match then the guessed scores are checked against the actual scores in another IF statement.

If the scores match, then two things happen. First the “updateDB()” method is called passing in the savings account Iban and the current account Iban and the amount that the transaction should be. Then the firebase record is deleted as it won’t be used again. If the scores don’t match, then the record is still deleted as the match has occurred and lastly if the “matchID” doesn’t match the match result is pending and will be checked later.

The “updateDB()” method as stated in the “Server” section calls the “$.post()” method containing the transaction.php http request. It passes in the Iban that the money is leaving and the Iban that the money is going into, along with the amount the transaction is worth and lastly the date. All used in the PHP script. Like the image below.



Figure 42: JavaScript Checker Flow Chart

## Android Application

### Classes

The Common Cents mobile application is a congregation of 21 Classes in Android studio. Each Class has a unique function in the application and they are all a mixture or activities, fragments, asynchronous tasks, adapters, objects etc. These are all the logical classes that are used in the front end of the application. All the Activity, Object and Fragment classes all have a corresponding XML file that is used to style and layout the class, as these classes will be displayed to the user in the Graphical User Interface, so will need a design. A good example of this is the main Activity. This activity is where the user can Log in to the application or begin the registration process. This Class has an XML file in the layout directory. In the XML file it is specified what “TextViews” and “ImageViews” etc are uses in that activity. The functionality of that activity will be conducted in the java Class. A list of Java Classes and XML files are listed below.

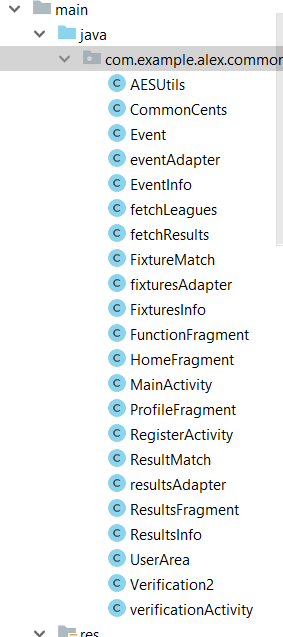
 

Figure 43: Java Class List Figure 44: XML Layouts list

### List Activities & Adapters

This section will revolve around the three main fragments in the application that house “ListViews” to which the user selects an item in the list. These “ListViews” involve a few steps to set up. The steps include creating a custom list item in XML that will be displayed in the list over and over, the object involved in housing the data for each item and the adapter that prepares the object to be put into the list. This List layout is used in the “UserArea” of the application 3 times under different fragments. It is used in the home fragment (“HomeFragment”), the fixtures fragment (“FunctionFragment”) and the results fragment (“ResultsFragment”). Essentially, they are all the same principle in the creation with minor alterations to each to fit the specific list.

The First step was to create the “ListView” in the Fragment in the xml layout and designing it like you normally would. The next step was to create the custom row for the list that would be displayed in each item in the list. You create this is XML similarly to an activity or fragment. You design it as you wish. For example, the “result\_row” has a “TextView” for the home team on the left side then one for the “v” and then one for the away team on the right side. This is the top, then on the bottom there is a “TextView” for each team’s score and the date.

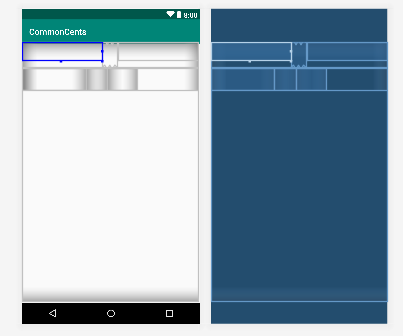


Figure 45: XML List Layout example

The next step is to create the object of the item that is in the list. For example, again using the results, there is “ResultMatch”. This is an object class that can be called. Once the class has the correct data assigned to its parameters, it can be used in the list. The class consists of parameters needed and a constructor.

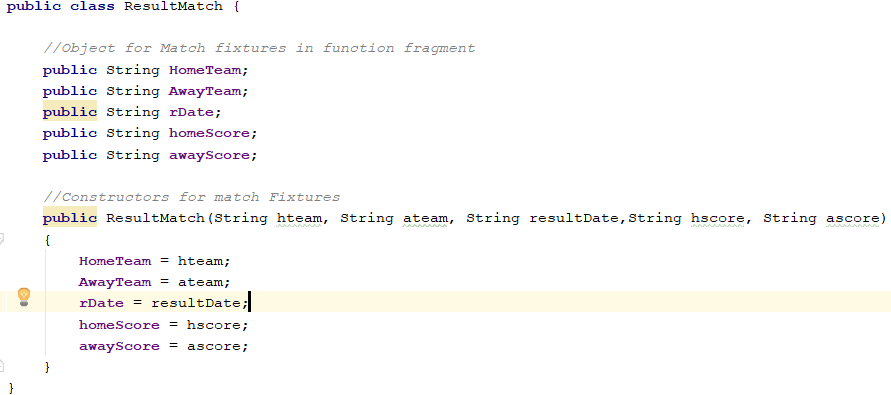


Figure 46: List Object Class

Once this is completed, the next stage is to get data and create the an “ArrayList” of objects to be assigned into the “ListView”. Any sort of data can be used for this, but in two of the three cases in this application I used an “AsyncTask” to retrieve data from the “APIFootball” database for fixtures and results for the coming and past two weeks. In the other “ListView”, a for Loop through Firebase is used to get the data and assign it. In this step, an “ArrayList” of type Object is needed. In this case the event, “fixtureMatch” and “resultMatch” objects were the objects in the array list. Once this is created, a loop through all the data you have is needed as you need to create an object for each of the entries in the dataset. For each of the entries, whether they are JSON objects it Firebase data snapshots, the object is called passing in the corresponding values. Once this is complete, this object is added to the “ArrayList”.



Figure 47: Array List for ListView

The next Step is the Adapter. This is called in the same class that the data is being fetched in, in this case the “AsyncTask” class “fetchResults”. Once it has fully executed, the class calls the adapter class, passing in the activity’s context and the “ArrayList” of objects. In the adapter class is where the data in the array list is prepared and set to the custom list layout that was set previously. Initially, there is a constructor, assigning the passed in values to the variables in the class itself. Next the custom row that was set is “inflated” so it can be used. Following this each of the values in the array lists object is assigned to the corresponding “TextView” in the custom row. Once this is done for each Object in the “ArrayList”, back in the “fetchResults” class, the adapter of the “ListView” is set to the custom adapter that has been created. Now the list will appear, and more functionality can be added to the “ListView” such as an “OnClickListener”.

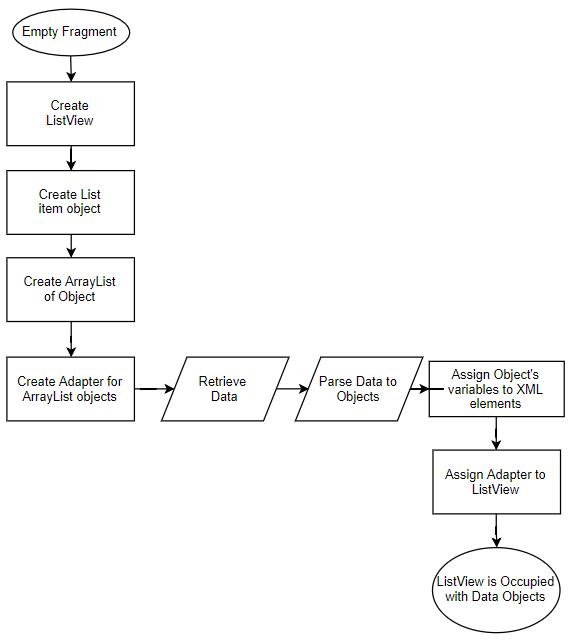


Figure 48: List View Implementation Process Flow Chart

### User Area Activity

This is the activity that houses all four of the fragments. Each fragment is its own class, but it is controlled in the User Area Class. The User Area has a bottom navigation bar that holds an option for each activity. To select which fragment is which a switch statement is used

The numbers in each condition represent which fragment is displayed, and what animation should be animated when clicking a new option in the nav bar resulting in a fragment change. The animations are created in XML, and they are utilized in the java class. If the next fragment is to the right and a higher number, a right to left exit is used and a right to left entrance is used and left to right exit and entrance is used if the fragment is a lower number and to the left. This gives a nice feel to the user interface.

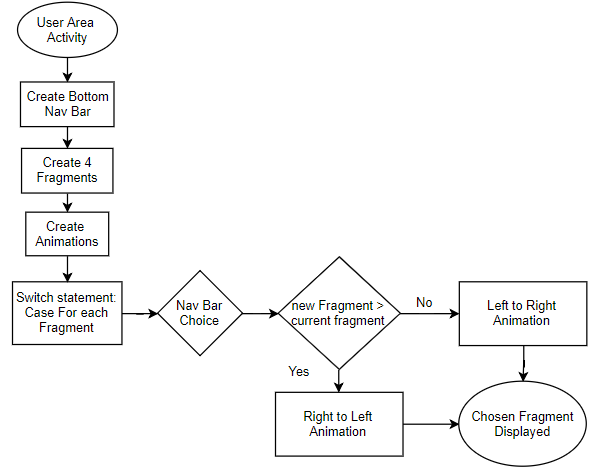


Figure 49:User Area Class Implementation Flow Chart

### Verification Activities

Two activities involve verifying the users PIN from their bank account. These activities have a basic design but have the volley request that calls the Heroku server to complete the “verify.php” script which returns the encrypted PIN from the Bank Database. The entered PIN is then encrypted and then is checked of it matches the encrypted PIN from the Database. If the user fails to verify their accounts, they won’t get to create the Common Cents account or change the Ibans. The user will only receive 5 tries at the PIN guess before they cannot try again, and that activity terminates. If the verification is the Register verification and the verity is successful, then the user’s data will post to firebase and a new user is created. Whereas if they user is just updating their Ibans, then the records on firebase will be updated, including all the Ibans in the events that are connected to that user.

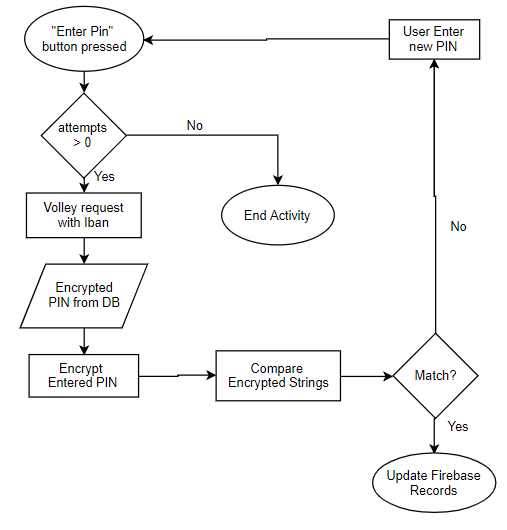


Figure 50: Verification Class Implementation Flow Chart

### Register Activity

The Register activity consists of many “EditTexts”, to which the user enters their personal information to create an account with the application. The “EditTexts” are declared at the beginning of the activity and they are linked with the specific XML object from the register layout. This is done with the “findViewByID()” method and inputting the relevant ID for that object.

Next is the If statements that ensure that all the data inputted is valid correct as error checking. The first if statement is to check if all the fields are filled out. Then there are error checks in place to ensure the Date of Birth is a valid date, along with the email regular expression that ensures the email is a correct email. These checks are done using the “.matches()” method.

There are also error checks to ensure the password entered matches the confirm password. These are in place to ensure the information posted to firebase is correct. If fields are incorrect or not filled out, the user is notified, and incorrect fields are marked in red. These nested if statements look complicated but are great for error checking as you can single out which aspects of the form aren’t valid.

The passwords to be posted to Firebase are Encrypted using the AES encryption algorithm. This is in the “AESEncryption” class in the class list. Once all fields are valid the verification page is loaded and if the user correctly verifies then the user’s data and the encrypted password will be posted to Firebase. This encryption adds a layer of security for the password safety so if the database is breached the accounts and passwords will still be private.

If the checks don’t give errors, the application opens “openVerification()” method. Here, the password is encrypted in this method then attached to the intent to be allowed to be used in the verification activity.

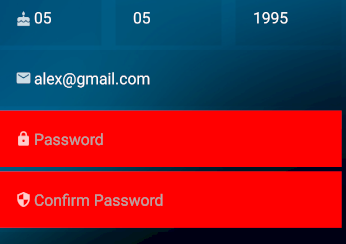


Figure 51: Incorrect Info Example

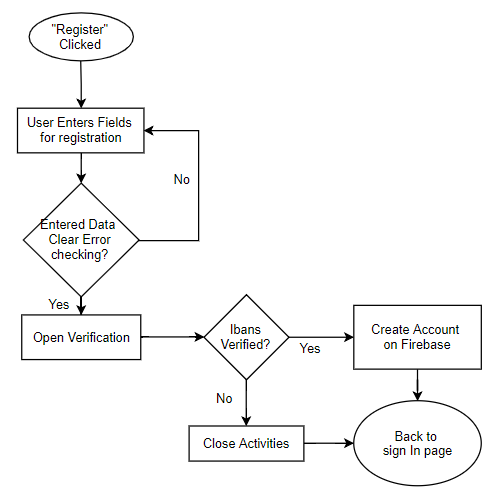


Figure 52: Register Class implementation Flow Chart

### Event Information

The event information activity holds information on the specific event chosen from the list on the Home fragment. This activity displays a Pie chart for the user that shows a visual representation of the machine learning prediction from Firebase. A visual of crests can also be seen on the page along with the prediction that the user has set so they can see more in detail what their prediction is. There is also a button at the bottom of the page to which the user can use to delete their prediction if they choose.

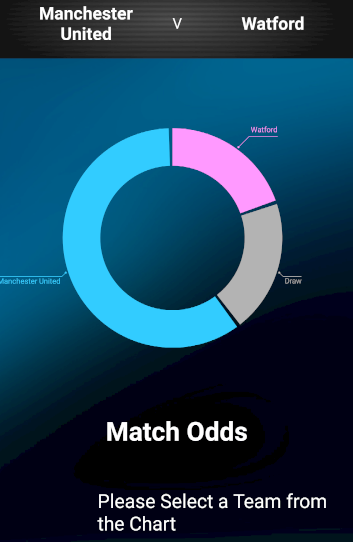
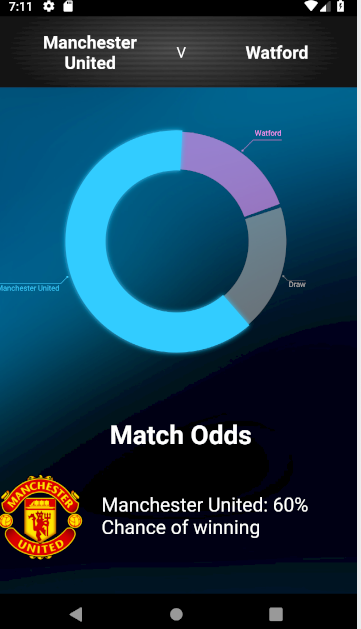
 

Figure 53: Event Info example 1 Figure 54: Event Info example 2

The Pie chart is a library that must be implemented and synced in the app level gradle file. This Pi chart is interactive. If one of the outcomes on the chart is selected, the section is highlighted, and also the “TextView” below is altered so it displays the crest of the team selected and the Integer percentage of the likely hood of that outcome.

Below the predictions area showing the Pie chart and chance percentages of the different outcomes is the Live scores. Here is the area which shows the current scores of each time and the time in the match, whilst the match is in play. This is done through a callback function that is called when the activity opens and is called again every 30 seconds to ensure the scores are up to date.

When the activity ends so does the callback function. The callback function makes an API call that retrieves the lives scores. The JSON array that is returned is parsed into strings, so the scores and time can be displayed in the “TextViews”. An extra feature, that adds to the user experience, is how if the match is in play and the users score guesses match the scores in the match, their prediction text is made green to imply that specific score is correct. If it is wrong, then the text is made red. If the match is not in play, the live scores area will be shown as “Match Not Available”.

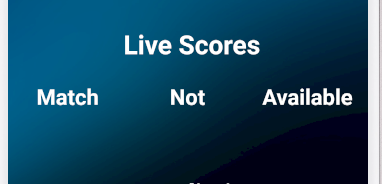
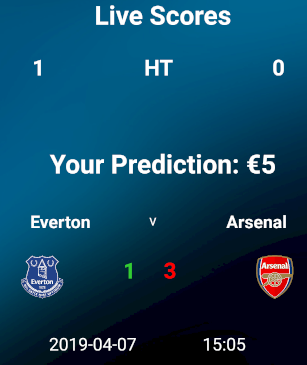
 

Figure 55: Live Score example 1 Figure 56: Live Scores example 2

Underneath is the prediction the user has set. Here all the information is displayed. The information displayed is the amount set by the user, the date of the match, the scores of the teams and both team’s crests. Below that again, the Delete button is seen. This button is used to delete the record of that prediction set by the user. This is there to allow a user to go back on their prediction for any given reason. Once this button is pressed a notification will appear asking the user if they are sure they want to delete that prediction to which they selected. If yes is selected, then it will be removed from firebase and if no is selected then it will return back to the activity.

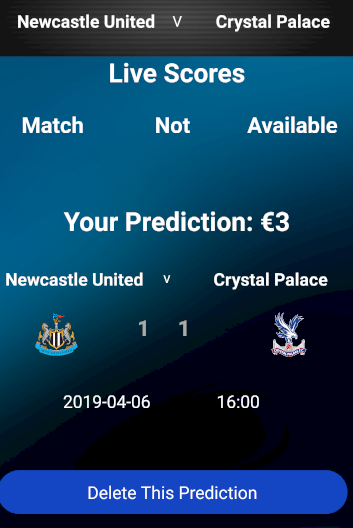
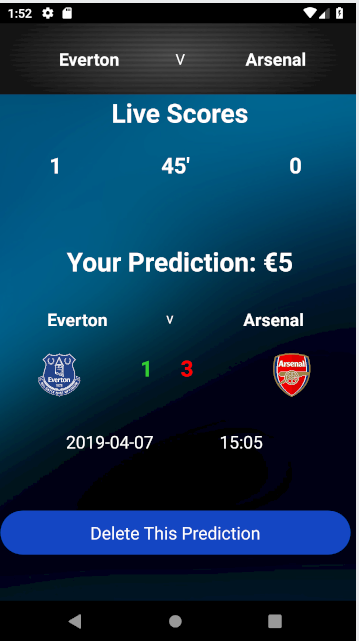
 

Figure 57: Prediction example 1 Figure 58: Prediction example 2

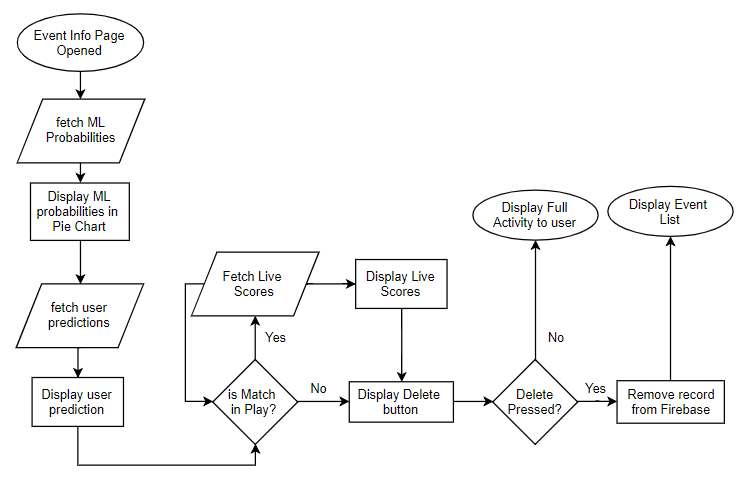


Figure 59: Event Info Processes Flow Chart

### Fixture Information

This Activity is where the user first sees the machine learning algorithm in play, and where they make predictions on the coming matches. Once fixture is selected from the fixture list this activity will appear. The user will be shown the teams playing in the match along with their crests, the percentage that each outcome will happen (Home win, Draw, away win) and an area where they can set their predictions using “EditTexts” and “TextViews”.

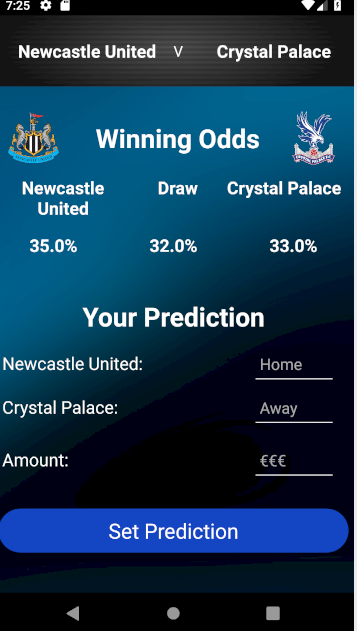
 

Figure 60: Fixture Info example 1 Figure 61: Fixture Info example 2

All elements from the Layout are declared and linked to each variable as normal. The crests are displayed using the Picasso library with the URL to the crest inputted to it. The machine learning probabilities are taken from firebase and displayed as percentages to the user. This is to help the user decide what their prediction should be but as it is a percentage they will have to make their own decision on the final score. This is to reduce any liability in the application if the prediction is wrong.

Below the machine learning outcomes, the space to set the prediction is shown. Here the user must set 3 parameters. The parameters being the home team score (1st), the away team score (2nd) and the amount they’re willing to put on it. Once they have decided with the prediction they press the set prediction button. Once that button is pressed then all that information is stored in the events array on Firebase.

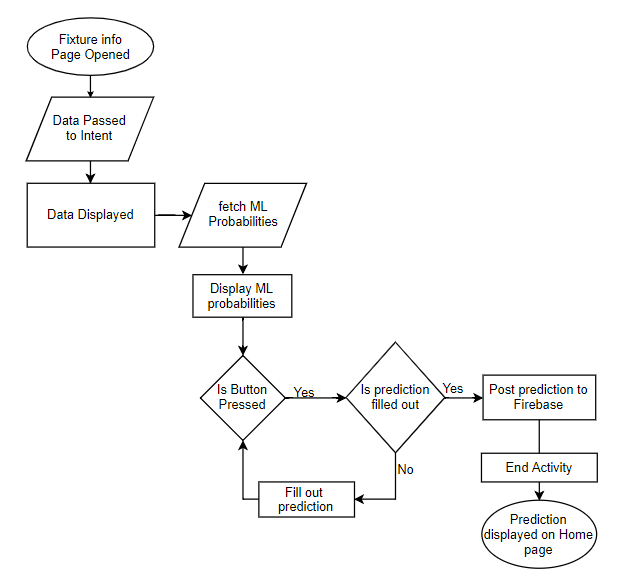


Figure 62: Fixture Info Flow Chart

### Result Information

This activity is where the user can see an in-depth statistics report on a selected match. They are faced with both team names, both crests and a whole variety of match facts that have been retrieved from the API. They are given a helpful Graphical interface that can really show if a match was one sided or relatively level. Numerous progress bars are displayed to the user, representing each match fact. The percentage of the progress bar is calculated by the data taken from the API and put into an integer to set that progress of the bar. This is done by getting a percentage of each team’s statistic and making it into a percentage. These visuals will help the user get a quick idea of how the match went.

Figure 63: Result Info example 1 Figure 64: Result Info example 2

### Profile Fragment

This fragment in the user area activity is where the user can view and change any information regarding their account with the system. There are three sections on this fragment. The sections are for the user’s personal information, their Ibans linked to the account, and their password. To change their personal information, such as email or username, they will alter the text in the “TextView” and press the “change info” button which will make an alert pop up to which the user must agree to, to change their info. If they select no the info won’t change. There is error checking in place here also to ensure no duplicate usernames in the system and make sure all information is valid e.g. email.

The section below that is to do with the changing of accounts and updating the Iban of the user to change where the money is coming into and out of. Similarly, to the section above, if a change in the user’s data is detected after the button is clicked, then the alert will appear ensuring the user is wanting to change their info. If yes is selected, then the verification of Ibans activity will be displayed to where the user must again verify their bank accounts with their PIN and have it matching the one in the Database. If the PINs match the Firebase is updated with the new accounts in both the user information and in the previous events they have set, to ensure money leaves from the right account. In this “OnClickListener”, after the Error checks Check out and the data is correct, the data is stored in the new activity that appears by adding extra data to the Intent. That means when the Pins are verified in the next activity they can be posted to firebase from that activity.

Lastly is the Passwords section of the profile fragment. Here the user can change the password that is related to their account. This is possible for security reasons. Again, to verify the user, they must enter their old password in the first “EditText” which must match the password in the database. Below that they enter their new password and a confirm of that password to ensure the password is what they initially wanted. Once again, if the user presses the submit button then the alert will appear again to ensure the actions of the user are correct. If the response on this is positive, then the account password will be updated with the new password to allow the user entrance to the application.

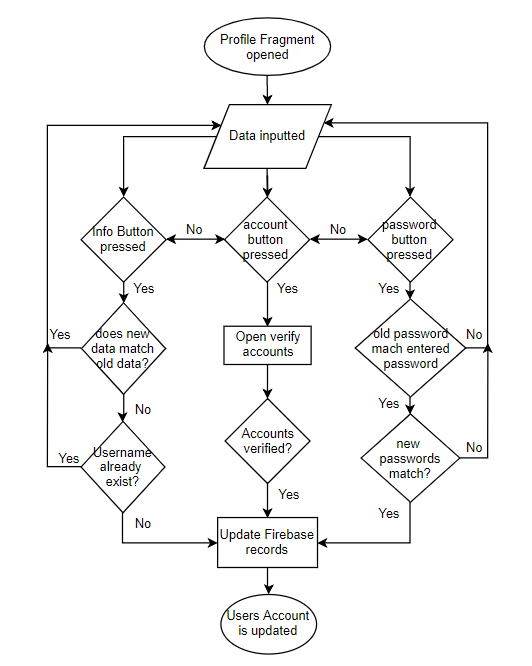
The old password is pulled off Firebase and decrypted using the AES algorithm in the “AESEncryption” class once this is decrypted, it is checked that it matches the entered old password. If it matches, then the two new entered passwords are decrypted. if they match as a string in the if statement, then that encrypted password will be posted to firebase as the new password.

Figure 65: Profile Fragment Processes Flow Chart

### AESEncryption Class

This Java class is where the encryption and decryption functions are called on the passwords as they come into the application from Firebase and get Posted to Firebase. The methods in here are based of the AES encryption algorithm.

The encryption process involves getting a string passed into the encrypt method. This method then gets a “rawKey” from the “getRawKey()” method which retrieves a secret key from the AES cryptography library imported to the class using a predetermined “keyValue” byte array. Next it calls the “encryptBytes()” method passing in the key and the bytes of the String that was initially passed into the class. This method then gets another secret key using the “rawKey”. Then the method gets an instance of the AES cipher. The cipher is initialized in “ENCRYPT\_MODE” using the “secretKey” retrieved previously in that method. The bytes of the initial string that are then passed into this method and are then encrypted using the new secret key. This encrypted byte array is passed back to the original encrypt method. This byte array is then converted to Hexadecimal String, and this hexadecimal String is passed back to the original class that called the method. The passed in string has been converted to a hexadecimal String much longer than the original and in this case the passwords will be then stored on Firebase.

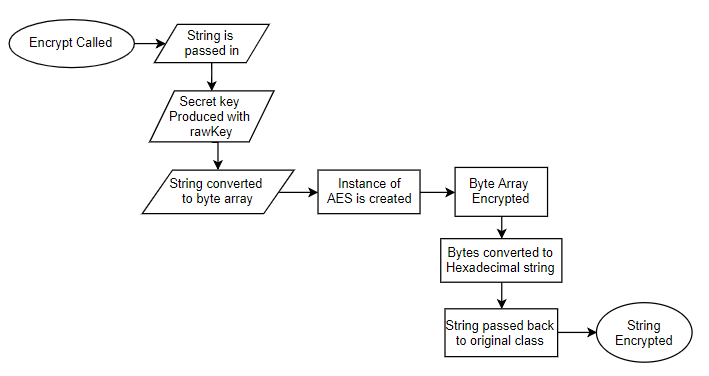


Figure 66: AES Encryption Flow Chart

The decryption process is the opposite of the encryption process. The “decrypt()” method is called from an outside class, passing in a String that has been Encrypted by this particular algorithm. The String will be some sort of Hexadecimal String. Once this string is passed in to the method, the “toByte()” method is called passing in the hexadecimal string. This converts the Hexadecimal to an array of type bytes. Once the string has been converted back to a byte array, the “decryptBytes()” method is called to decrypt the bytes. Here much like the “encryptBytes()” method, a “secretKey” for the decryption is retrieved from the cryptography library, using a predetermined “keyValue” Byte array and also choosing the AES algorithm. Again, an instance of a cipher is created using the AES algorithm. The cipher is initialized in “DECRYPT\_MODE” using the “secretKey” retrieved previously in that method.

The encrypted array of bytes is then decrypted using the “cipher.doFinal()” method and that decrypted array of bytes is assigned to a variable called result in the original decrypt method. Lastly, the result byte array is converted back into a String and returned to the original class that called the decryption method. In this case the password will be decrypted to what the user originally entered. Due to security reasons pointed out by the project supervisor, the decryption methods were not utilised as it was informed that for better security nothing was decrypted in password check, that the entered password was encrypted and then the string check is done on the two encrypted Strings. For this reason, the methods are not used but for the project, the methods are still within the code.

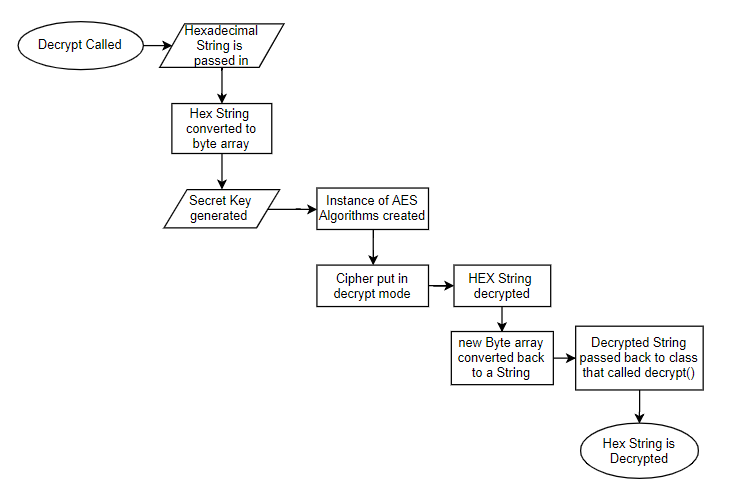


Figure 67: AES Decryption Flow Chart

## Python Machine Learning

### Libraries

The Library imports of the machine learning project are of huge importance. Firstly, there is Pandas and Numpy. These two libraries are mainly for conducting certain operation on the dataset and on the numbers that are in the dataset. E.g. imputing rows or columns in a dataset using Pandas or calculating array operations using Numpy. Next is the JSON library. This is used in conjunction with firebase to allow the program to upload a JSON array directly. Next are the libraries that are used for the machine Learning model itself. Here the Poisson Regression algorithm is imported alone with two other models that help in the creation of the model. Lastly are the Firebase libraries. These are what allow the program to connect and upload directly to Firebase, so the data can be used elsewhere.

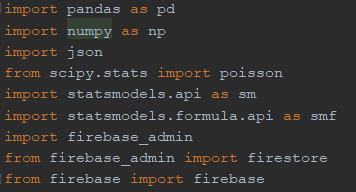


Figure 68: Python Imports

### Data Management

The next aspect of this program is the data management. The first part is just initializing the connection to the Firebase Database. Next the Dataset is retrieved off the online resource at football-data.co.uk and it is assigned to a dataset variable. Next the columns are dropped that aren’t needed, or in this case they are all dropped but the ones that are needed. These being, the team names and the full-time goals. These are the only columns needed for this program. The columns are then renamed. Next There is an array created with all the teams in the Premier League this season to be used later in the program. If the mean of the home goals versus the mean of the away goals is taken at any time, the mean of home Goals is always greater than the away goals average. This is known as the Home Team advantage. With this Model, the Home goal advantage is considered due to the method of how the predictions are received. The home advantage usually equates to around 20-25% extra goals per match, as seen below.



Figure 69: Home Team advantage example

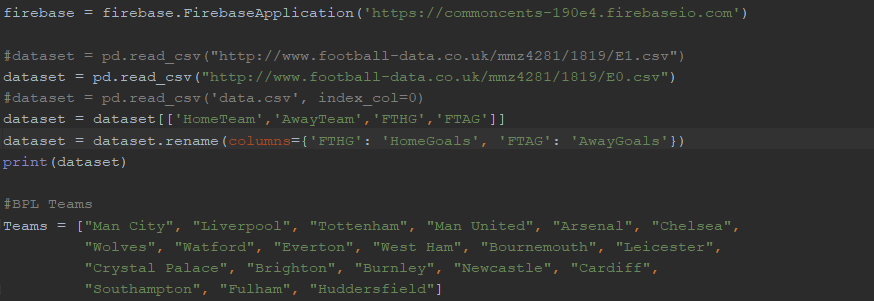


Figure 70: Python Data Management

### The Model

Due to the size of the dataset, especially at the beginning of the season, one match, good or bad, can greatly affect the model and its predictions. This can be a down side to the model as it can cause inaccuracies. As the season goes on the more accurate the predictions should be. The previous seasons statistics can be used but due to the summer break being 3 months and the transfer window being open and new players joining different clubs and also the relegated teams and promoted teams per season, it is felt that using previous season data isn’t going to improve and accuracies. Only using this season’s data improves the accuracy towards a team’s form during the season although the overall accuracy won’t be the best at the beginning of the season.

For the model creating itself involves creating 2 new datasets based off the data that we already have. The 2 new datasets are taking both teams and firstly assigning the away team as the “opponent” with the home team as “Team” then secondly the home team as the “opponent” with the away team as “Team”. The goals the “Team” scores are the goals in the dataset. There are only 3 columns in each dataset. This is done because each team’s goals are treated independently. These Datasets are then concatenated into one dataset. This “dataModel” is then used in the “poissonModel”.

The formula is declared and initialized before being used in the “smf.glm()” method. The formula in this method is set to the formula, the data is set to the “dataModel” and the family is set to the necessary Poisson exponential family for this algorithm. This is all fit to a Generalized Linear Model. This model will then output a few different values such as the coefficient, the standard error and the confidence interval etc. The value that is in the coefficient column is what’s used in this case. The Closer to 1 the value is the more likely the team is to score more goals whereas closer to 0 means more likely to be a draw. If a Team has a high coefficient and their “opponent” has a minus coefficient, then that team with the higher coefficient tends to be a lot better than their opponent and are far more likely to score more goals against that team.

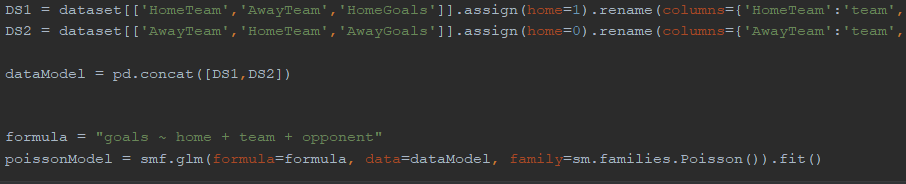


Figure 71:Python Prediction Model

### Predictions Function

The next step in the machine learning program is the function that will be called on each match to predict the outcome. It takes in the model going to be used, in this case the “poissonModel”, the home team, the away team and the max number of goals in the match. The function calls the .predict method on the data first set as the home Team as the “team” and the away team as the opponent, and secondly the other way around. Next the predictions matrix is created. This prediction matrix represents the possible outcome of the match with the two teams. The diagonal line through the matrix represents the probabilities of the match being a draw based on the amount of goals scored in the game. Adding up these diagonal values on the matrix gives the probability of the match being a draw. Any value above this diagonal line represents the home team probabilities based on the goals scored in the match and below is the away team probability of winning based on the goals scored. Summing up above the diagonal line represents the probability of the home team winning and the same goes for below the line and the away team winning.

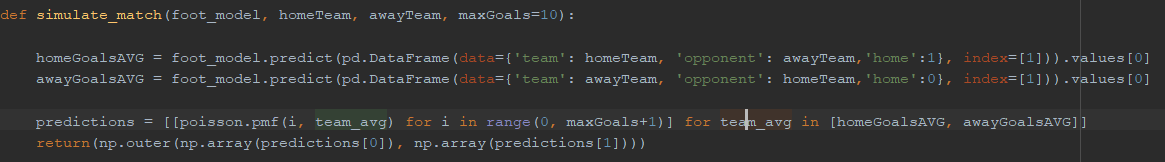


Figure 72: Python Prediction Function

### Function Call and Matrix Operations

The next area in the program is how the function is called and how the data is sorted afterwards. Nested While loops are in place to ensure iteration of every team in the league playing every team and not playing each other. Inside the While loops the function to get the prediction matrix is called with the current “i” and “j” teams along with the “poissonModel” and max Goals 10. The matrix is returned with the prediction on how the match is going to go. Next the 3 possible outcomes in the matrix are added up to get the final probabilities on each outcome regardless of the goals scored. The output is 3 values. The home team probability of winning, the probability of a draw and the away team probability of winning.

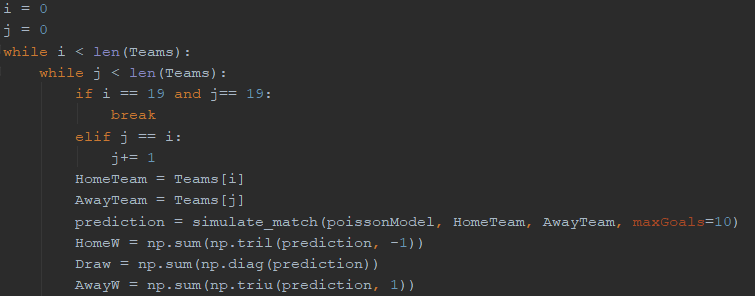


Figure 73: Python While Loops and Function Calls

### Firebase Data Preparation and Posting

The last area in this program is regarding the data that has been produced by the model being posted on Firebase. There is a large if statement that changes the team names form what they are in the dataset retrieved from football-data.co.uk to what they are in the “APIFootball” database. If the data is posted in the same format and using the same team names as the API call, the Android Application calls to firebase regarding football teams will be easier to understand as the team names won’t change in the Android Code. This goes for both home and away teams. Lastly in the nested while loops, the prediction is put into a JSON object and appended to the predictions JSON array. The team names and all the 3 outcomes are all inside a JSON array. Each possible match is given all the values and assigned to a JSON object and appended to the JSON array. Outside the While loops, the last operation of the program is posting the full JSON array to firebase in one go. The Predictions are now posted to Firebase and are ready to be used in the Android Application. The results are also printed to the console of the Python Program.

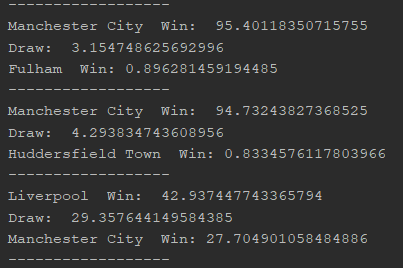


Figure 74: Python Predictions example

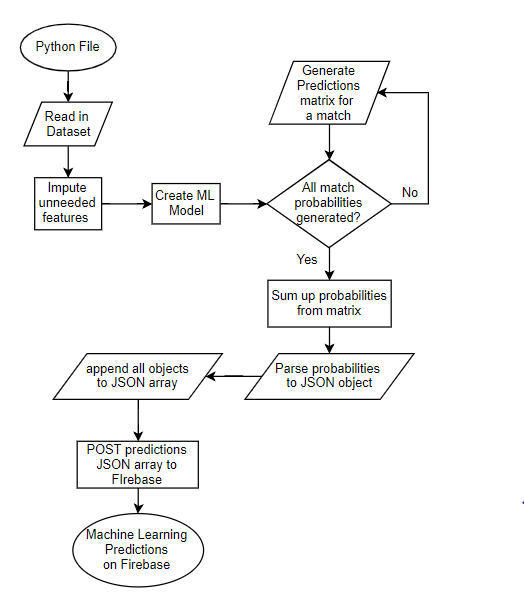


Figure 75: Machine Learning Processes Flow Chart

# Testing

## Overview

This section describes the processes that this project went through in terms of testing. Two forms of testing were conducted, “Black Box” and “White Box” testing[53]. For the Black box testing, the main goal was the functional testing. This is the testing and verifying of each function of the software and how it operates in relation to the whole application[54]. Some tests in this area include logging in, the user creating their prediction and the user updating their account. The white box testing for this application revolved around the non-functional testing as its focused more so on the usability, performance, debugging etc.

## Black Box Testing

Black Box testing was to test the overall usability of the project. The main areas focused on were checking for interface errors, missing functions and behavioral errors[53]. To test for these issues, the application was tested by five different users who installed the application on their phone and used it for a period. Then a small survey was asked of them for feedback on the application. This testing is great as it gets the users feedback on the application, but they are unaware of the functionality of it. “Testers only know the set if inputs and predictable outputs.”

### Q1. Were you able to register?

Three out of the five users said that they were able to register for an account easily and without any issues. The last two users were able to register for an account but said that the registration process wasn’t as fluid as other registration processes. They both put it down to the additional Iban input and verification process that must be done for the application to ensure the user has their accounts set up.

### Q2. Were you able to Log in immediately and consistently?

### 

All users reported back positively. All 5 were able to log into their account immediately and whenever they wanted to log in they were able to, no one had any problems in this area.

### Q3. Were you able to change profile information?

Four of the five users were able to change any information they wanted whenever they tried to change information. Three of these four users commented that the process of changing any of the information wasn’t as intuitive as they would have liked but still didn’t have any issues. One user was not able to change their Iban accounts information when they tried. This was an issue with the SQL mock Bank Database that has been fixed. This user was able to change other information easily.

### Q4. Were you able to create and delete your predictions?

All users reported that they could make the predictions and set the amount of money as they wanted with no issues. All Users also reported that they were able to delete their predictions whenever they tried to. When the predictions were created they showed up on the home fragment and when they were deleted, they would appear there. Two users reported that they could delete predictions off their home fragment if their prediction came true and have no money transferred. This can be addressed by updating the JavaScript Checker to occur more times throughout the day.

### Q5. Did the Live Scores work during matches?

All users reported that the Live Scores worked through the different matches and the information was updated very quickly. Three of the five users commented on the user interface turning the prediction numbers green and red based on whether their prediction was on track or not.

### Q6. How did you find the match’s outcome probabilities (ML)?

All users said they percentages of the outcomes that were displayed were helpful in the decision of their own predictions. All five users said when creating the predictions that the percentages of outcomes looked right in terms of how the league was going, so the better teams having a better chance and with home team advantage etc. two of the five users claimed that the percentages ended up being inaccurate, but that it could be put down to a poor performance for the teams in question.

## Black Box Testing Conclusion

Based off the surveys filled out by testers of the application, the application seems to be in good working order with not major bugs to be sorted out. This is very positive news for the project and its overall usability. Aside from this there were some issues that had to be addressed and were. One big issue was the ability to delete a prediction after the outcome had already come true. This is a big concern as it means the user can go back on their prediction if it comes true. This was addressed by reducing the match checker to 4 times a day, rather than the once. One issue was with the Bank Database that has now been fixed, the user was unable to change their accounts in the profile section due to the verification page. Another big issue is the usability of the application and how intuitive it is. With the extra information of the Ibans, the register is a bit unnatural, but it must be this case in order for the accounts to be linked to the right Bank accounts.

## White Box Testing

To test for any bugs or any logical errors in the applications code, White box testing was used. This testing is used to find any “typological errors, and uncovering incorrect programming assumptions”[53]. This testing was done throughout the implementation of the project in the non-functional areas. There were three primary types of white box testing conducted in the project. They were, “Code Walkthrough”, “Loop Testing” and “Path Testing”.

### Code Walkthrough

This testing usually involves a third party to the source code, but in this case, it wasn’t needed. This is where the source code is followed line by line with a set of test cases to ensure the applications function works properly[53]. For each of the classes in the Android application, whilst it was in development, function would be written, and test cases would be applied to ensure the functions work. An example of this can be seen with the “AESEnctyption” class. A “walkthrough” line by line was conducted on all the code. This gave clarification on the different keys and different byte arrays and how they were correct or incorrect and if they were incorrect which they were twice, they can be adjusted

### Loop Testing

“Errors often occur near the beginning and end of loops”, so because of this the testing of the preciseness of the loops was very important. Any incorrect loops could result in a failed function or the crashing of the application[53]. To ensure this didn’t happen, individual tests on nested or even non-nested loops often occurred in the implementation of this application. A primary example of this can be seen all over the application as loops are a huge factor in the application.

One main example is in the JavaScript Checker script. Here there are numerous loops for both the API call to parse the data and also in the Firebase call to also parse the data. Because these loops are also checked off each other, the loops must be accurate and error-free. If the loops were checking the wrong data off the wrong data then the user’s predictions could be falsely analyzed, then the incorrect transfer of money from current accounts to saving accounts could be undertaken and resulting in a major flaw in the application. All the loops in the project were tested mainly using alerts such as “System.out.println()” for the console and “Toast” for inside the app itself. The values of different variables would be put inside these functions to be displayed. In seeing the contents of specific variables, it can be determined whether the correct data is there or not, and if not then the issue with the loop can be altered as necessary. That allows the development to proceed if the values in the variables is correct and if incorrect the Loop will be revised and rewritten.

e.g. in the Machine learning there is a nested while loop. If a nested for loop was used in this situation, the incorrect order in the loops is an issue, some date posted to Firebase will be redundant and won’t be able to be used. This can be seen if a prediction where both teams are the same, e.g. Arsenal, the prediction will still be taken and posted to Firebase but can never be utilized in the application and is therefore redundant and solely there taking up memory.

### Path Testing

“Path Testing” is the testing of every possible executable approach to the application. A path is determined based of the calling of specific functions in a given order[53]. The path testing involved in this project was mainly based off the control flow and can involve different areas and can revolve around a “happy path”, “bad paths” or “Alternate Paths”. All the functions in the application can be represented in a flow chart.

The Happy path is in relation to the “shortest path to happiness”[55]. An example of a Maze can be used in this case. From point A to point B there is only one path that is the fastest to get to point B. An example in the project is when the user is registering. One test was for the regular expression on the email box. The test would be if a string with the correct email format was entered that would be a success. Everything. All the correct information or data is inputted with no exceptional or error exceptions. This is known as the happy path.

The bad path methodology is regarding a path that involves error exceptions. Error checks and solutions must be in place for this sort of path testing[53]. Taking the maze example again, if on the way to point B a wrong turn is taken, and a dead end is reached, an error has occurred. The maze is not over, you go back to a different turn and try again. This is error exception handling. In the application, an example if this can be seen in every class. Error exception handling can be seen in Volley requests, converting a string to a JSON object, connecting to Firebase and in many other situations. Taking the Firebase as an example, if data isn’t posted to firebase and there is no error exception handling, then the application will crash. Exception handling as simple as a Toast to the user saying “Failed to connect to Firebase” will suffice in some cases, other cases need more detailed handling. The bad path involved what happens in the application if a user comes across error exception handling. The email example again, if the entered string does not match an email format, then the “EditText” will be cleared and marked in red to display to the user that there is an issue. This testing ensures all the error exception handling is working and no crashes occur upon bad information being entered.

The Alternate path methodology is regarding the different ways to get to the same place as a happy path flow[53]. Error exceptions are also needed for this type of path testing. Again, the maze. There can be many ways to get from point A to point B. the “happy path” is the fastest way whereas the alternate routes are the “alternates paths”. Regarding the application, taking the end goal to have a prediction on the home screen, you can do it straight away as the happy path or you can visit other screens of the application to get a better understanding of the matches and to get a more informed prediction. For example, the user can visit the results page and view their favourite teams match facts for their last game to get a read on their form and see how they played. Then they can go to the fixtures list, pick the coming match and make a prediction based off the last match and still resulting with a prediction on the home page. This is an alternate path and “alternate path testing”

## White Box Testing Conclusion

The “White Box” testing proved to be very useful throughout the implementation process. This in conjunction with the approach methodology of an iterative waterfall model, ensured each section of the application was tested accurately to ensure the best result. For example, once a loop was made, it was tested to ensure it was correct, it not the loop would be fixed and tested again until it was correct and working correctly. Much like the loop testing, the code walkthrough would also occur whilst implementing the code. This ensured that the code was correct and again if errors were spotted, then a change would be made and adjusted.

These two testing methods showed to be very good as once it came to any sort of path testing there seemed to be very little errors and bugs. The path testing would occur when a large section was completed, and a few different things needed to be checked together if they all worked in unison. For example, if a fragment had a new feature added the path testing would operate on that once the implantation is complete and then a larger scale path test would take place and so on. This also showed to work very well for the project as errors and bugs would be discovered in this scenario when they wouldn’t have been before the path testing.

# Project Plan

## Overview

This section outlines and examines the plan for the project as it analyses what in the project worked, what didn’t work, what features were added and what features were taken away. A GANTT chat is also displayed here outlining the sections that were worked on and what amount of time was allotted to each section throughout the project’s implementation. Lastly, in this section, the strengths, weaknesses and challenges overcome will be outlined.

## Project Plan

From starting this project to the finishing of it, many ideas that could be implemented were initially thought of and brought to light during the development of the project. Some initial ideas for the project were scrapped completely due to the time constraints mainly but mostly they were implemented fully.

### Team Goals Trigger

One idea that was removed was the ability to set up with your account so that every time your favorite team scored a goal you would transfer €1 from your current account to your savings account. Unfortunately, as it is still felt that this is a good feature, the time constraints on the project were closing. This feature was in competition with the Machine learning to be implemented to the application. It was felt that the machine learning predictions on the odds of an outcome to the match would be of greater importance to the user and it would add a great degree of complexity to the project as it would be something new and modern added to the application rather than a variant of the JavaScript check Script. This would be a great feature to add in in that future and if more time was allocated, then it would be a great feature to have.

### Machine Learning

The machine Learning aspect of the project was initially meant to only be considered towards the end of the project. Due to its complexity and it being a great feature to be added to the application is was swiftly moved up the priority list in February when most of the initial design and fragments and activities were completed. A place to display the outcomes of the prediction model was needed to be considered. The best place was thought to be where the user sets their own prediction, so they can have help in making their decisions.

### Payment System

The payment System was a big concern for a long time. There were many different areas considered when developing it. Part of a payment system would be developed and then an issue with it would be discovered and the project would be reverted to where it was previously. In the end, the best sort of payment system was set up for the application as if it was working in unison with a bank. Any other system had too many issues in terms of developing it, such as the likes of Stripe and VISA charging per transaction or Dwolla not having an updated documentation. In the end, it is felt that the best one was implemented to the application due to it being effortless for the user and it being all developed through the project with no third parties.

## GANTT Chart

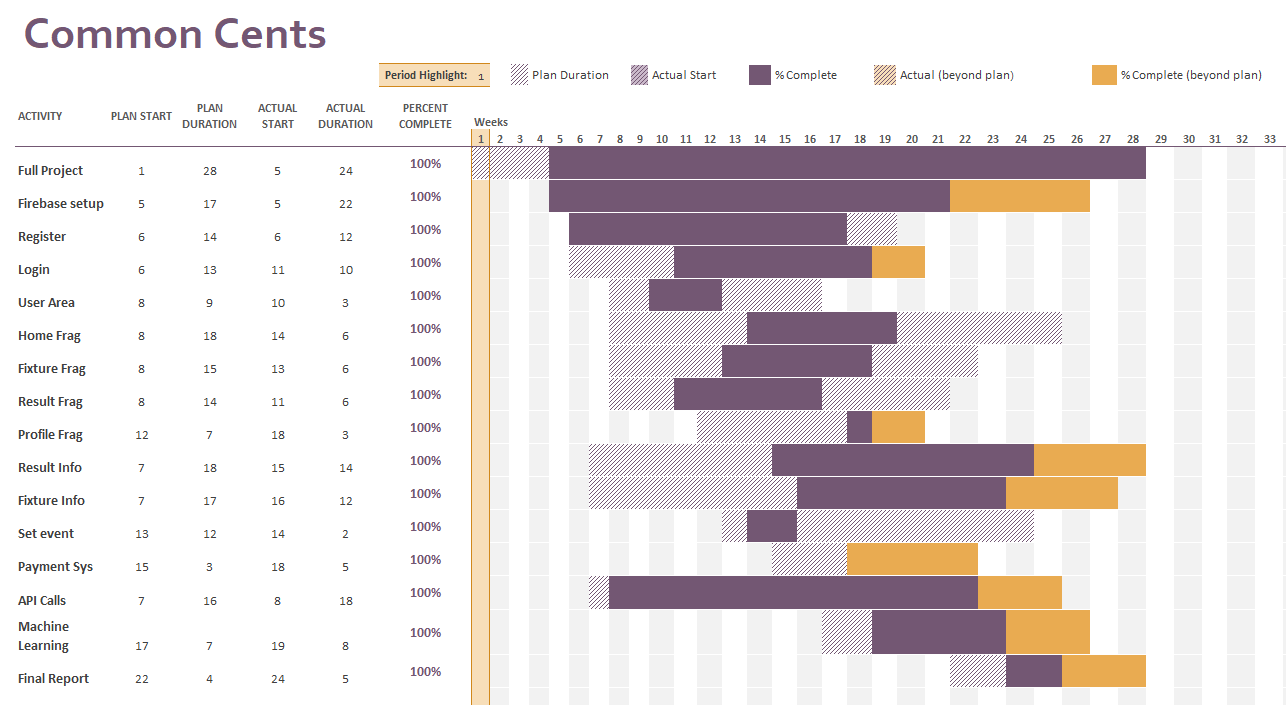


Figure 76: Project Plan GANTT Chart

This Gantt chart represents the work that has been completed on this project throughout the year. Each row in the chart represents a certain aspect of the application whether its machine learning, app activities or databases. One example, the Firebase setup is regarding the database that stores all the user data and events, crests and predictions. This was getting altered throughout the duration of the project, so it wasn’t fully completed until 2 weeks before the submission as the crests were only added around then.

In the chart, several things were started late. This was due to the design aspect of the project. The early stages of design on the application took up more time than expected. It can also be seen that many things were finished late, this was the case due to the number of things that were finished early such as the fragments. This allow extra time to be allocated to the other areas of the project, and thus more time was taken of them.

## Strengths

### Machine Learning

One good strength in the project I feel is the Machine Learning in the project. This isn’t saying its without flaws but having it with only the current seasons matches as a dataset means it only takes the teams form for this season. Previous seasons a team may have players of a lower quality and this season changes to players staff and management may result in the team playing much better in the given season. For this reason, I felt it was best to use the dataset for only the current season

### Overall Application

It is felt that due to the whole application working properly and working in unison, that this is a huge strength for the application. I can be very hard to fit all aspects of a project together but in this case, it has been done and all areas can be used by the user without fail and with a limited number of bugs. At the beginning of the project it was unsure how deep each aspect would go, but the level of depth that all the sections go to are certainly a strength, as it really does fit together very well and is very near a finished product for initial launch.

## Weaknesses

### Machine Learning

Although its mentioned in Strengths, the machine learning algorithm has its flaws. The issue with predictions is to do with what the predictions will be like at the beginning of the season. Due to the dataset only being the current season, the predictions at the beginning of a season may be inaccurate due to an unexpected start to a season. As the season progresses the predictions will get more accurate and will fit the teams better due to the better data.

### Real Life Banks

In this project a mock bank example is used, a real bank would have a much more complicated database. They would also be very sceptical of who they allow access to that database due to Security being their highest priority. Due to this, to launch this application in a real-life situation, the verification process would have to be revised. The process that transfers of money from account to account would also have to be revised due to them having a much more secure way of transferring money. There would be various changes that would need to be made for a real launch of the application in terms of the payments and verification of accounts.

# Conclusion

In terms of the overall project, it is felt that it was a success. Although, there are issues and draw backs with the project, it was concluded that due to the well function mobile application, the quality of the Machine Learning predictions, the well-functioning REST APIs and the good performance of the JavaScript Checker, that the project was a great success. The achievement of planning the time allocated to the project well enough to get a large majority of the functionality projected from the beginning is also something to be pleased with. The task of having a full application to develop with the functionality intended is not easy.

In saying that, there are still areas of the project that fell short. Due to time constraints, some extra money management ideas for the application would not be implemented, this was also due to the priorities of different aspects of the application. In this case mainly the machine Learning. It was felt that to greater the complexity of the application, some predictive analytics were needed.

Overall, the machine learning feature was a good accomplishment, but that is not to say it doesn’t have its flaws. Due to the lack of data in the beginning of a season, the predictions in the early stages of the season will be less accurate then they will be towards the end of a season. Regardless, the fact the predictions take the home team advantage into consideration to give a solid probability is very positive.

The Overall aim of the project was to create an application that could help the user save money whilst having fun and getting a bit extra out of their weekly football enjoyment. This main goal has been reached and ever surpassed.

Other Goals such as ease of use for the user have also been reached in terms of the user not having to initiate any payments, they are all automated and free flowing for the user, optimizing the user experience.

The design of the interface is also a goal reached. Due to the user feedback received in the “Black Box Testing” phase, it is known that the interface is intuitive and not overwhelming for a user. This was of great importance as having a complex interface can sway users to other methods of saving money.

Due to all this, there is a firm belief that Common Cents, with the right assistance, could be a fully working application to be used on the market with a large userbase. Some assistances needed would be with the banks in terms of the Payment System, as this is the biggest concern. Banks are very select to who can use their systems in this way, therefore a strong case would need to be provided to them to get them on board with the idea. If this can’t be done, then some form of custom Payment System would need to be that solution.

The whole process in the project’s development was very enjoyable and certainly beneficial. I very much look forward to working on this project in the future and seeing the outcome that it holds and watching how the project grows.

# Figure Table

[Figure 1: ML Classifier Example Diagram 22](#_Toc5902942)

[Figure 2: K-NN Algorithm Diagram 23](#_Toc5902943)

[Figure 3: Decision Tree Algorithm Diagram 24](#_Toc5902944)

[Figure 4: Linear Regression Algorithm Diagram 25](#_Toc5902945)

[Figure 5: Poisson Regression Algorithm Diagram 26](#_Toc5902946)

[Figure 6: Agile Crystal-Clear Methodology 33](#_Toc5902947)

[Figure 7: Waterfall Model Diagram 34](#_Toc5902948)

[Figure 8: Iterative Waterfall Model Diagram 35](#_Toc5902949)

[Figure 9: Application screen Flow 39](#_Toc5902950)

[Figure 10: Home Activity / Figure 11: Register Activity Figure / 12: Verification Activity 41](#_Toc5902951)

[Figure 13: Home Fragment / Figure 14: Fixtures Fragment / Figure 15: Results Fragment 42](#_Toc5902952)

[Figure 16: Results info example 1 / Figure 17: example 2 42](#_Toc5902953)

[Figure 18: Fixture Info example 1 / Figure 19: example 2 43](#_Toc5902954)

[Figure 20: Event Info example 1 / Figure 21: example 2 44](#_Toc5902955)

[Figure 22: Profile example 1 / Figure 23: example 2 / Figure 24: example 3 45](#_Toc5902956)

[Figure 25: Pipe - Filter Arhitecture 47](#_Toc5902957)

[Figure 26: Client - Server Architecture 48](#_Toc5902958)

[Figure 27: Common Cents System Architecture 50](#_Toc5902959)

[Figure 28: Use Case 1 - Register 51](#_Toc5902960)

[Figure 29: Use Case 2 - Full Application 54](#_Toc5902961)

[Figure 30: Use Case 3 - Machine Learning 57](#_Toc5902962)

[Figure 31: Full Application Flow Chart 60](#_Toc5902963)

[Figure 32: Full Application Sequence Diagram 61](#_Toc5902964)

[Figure 33: JavaScript Check Script Sequence Diagram 62](#_Toc5902965)

[Figure 34: Bank Database ERD 63](#_Toc5902966)

[Figure 35: Firebase - User Info Storage 64](#_Toc5902967)

[Figure 36: Firebase - Event Storage 65](#_Toc5902968)

[Figure 37: Firebase - ML Predictions Storage 65](#_Toc5902969)

[Figure 38: Firebase - Team Crest storage 66](#_Toc5902970)

[Figure 39: Transaction.php Flow Chart 67](#_Toc5902971)

[Figure 40: DBOperations.php Flow Chart 67](#_Toc5902972)

[Figure 41: Verify.php Flow Chart 68](#_Toc5902973)

[Figure 42: JavaScript Checker Flow Chart 69](#_Toc5902974)

[Figure 43: Java Class List / Figure 44: XML Layouts list 70](#_Toc5902975)

[Figure 45: XML List Layout example 71](#_Toc5902976)

[Figure 46: List Object Class 72](#_Toc5902977)

[Figure 47: Array List for ListView 72](#_Toc5902978)

[Figure 48: List View Implementation Process Flow Chart 73](#_Toc5902979)

[Figure 49:User Area Class Implementation Flow Chart 74](#_Toc5902980)

[Figure 50: Verification Class Implementation Flow Chart 75](#_Toc5902981)

[Figure 51: Incorrect Info Example 76](#_Toc5902982)

[Figure 52: Register Class immplementation Flow Chart 76](#_Toc5902983)

[Figure 53: Event Info example 1 / Figure 54: example 2 77](#_Toc5902984)

[Figure 55: Live Score example 1 / Figure 56: example 2 78](#_Toc5902985)

# Figure Table

[Figure 57: Prediction example 1 / Figure 58: example 2 78](#_Toc5902986)

[Figure 59: Event Info Processes Flow Chart 79](#_Toc5902987)

[Figure 60: Fixture Info example 1 / Figure 61: example 2 79](#_Toc5902988)

[Figure 62: Fixture Info Flow Chart 80](#_Toc5902989)

[Figure 63: Result Info example 1 / Figure 64: example 2 81](#_Toc5902990)

[Figure 65: Profile Fragment Processes Flow Chart 82](#_Toc5902991)

[Figure 66: AES Encryption Flow Chart 83](#_Toc5902992)

[Figure 67: AES Decryption Flow Chart 84](#_Toc5902993)

[Figure 68: Python Imports 85](#_Toc5902994)

[Figure 69: Home Team advantage example 86](#_Toc5902995)

[Figure 70: Python Data Management 86](#_Toc5902996)

[Figure 71:Python Prediction Model 87](#_Toc5902997)

[Figure 72: Python Prediction Function 88](#_Toc5902998)

[Figure 73: Python While Loops and Function Calls 88](#_Toc5902999)

[Figure 74: Python Predictions example 89](#_Toc5903000)

[Figure 75: Machine Learning Processes Flow Chart 90](#_Toc5903001)

[Figure 76: Project Plan GANTT Chart 98](#_Toc5903002)

# Resources

[1] Apple. “Swift - Apple Developer.” Available:https://developer.apple.com/swift/.

[2] M. Rebouças, G. Pinto, F. Ebert, W. Torres, A. Serebrenik, and F. Castor, “An Empirical Study on the Usage of the Swift Programming Language,” in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2016, vol. 1, pp. 634–638.

[3] Rick Walter. “Do I Need a Mac to Make iPhone Apps?” April 27th 2015. *Made up by People*. Available: http://www.madeupbypeople.com/blog/2015/4/23/do-i-need-a-mac-to-make-iphone-apps. [Accessed: 13-Nov-2018].

[4] Microsoft. “Xamarin App Development with Visual Studio,” *Visual Studio*. [Online]. Available: https://visualstudio.microsoft.com/xamarin/. [Accessed: 13-Nov-2018].

[5] Jared Dickson, “Xamarin Mobile Development,”2013. p. 18.

[6] Global Stats. “Mobile Operating System Market Share Worldwide,” *StatCounter Global Stats*. [Online]. Available: http://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed: 24-Mar-2019].

[7] Android Developers. “Download Android Studio and SDK tools,” [Online]. Available: https://developer.android.com/studio/. [Accessed: 13-Nov-2018].

[8] *Android Developers*. “Write your app,” [Online]. Available: https://developer.android.com/studio/write/. [Accessed: 13-Nov-2018].

[9] *JetBrains*. “IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains,” [Online]. Available: https://www.jetbrains.com/idea/. [Accessed: 15-Mar-2019].

[10] Mehul Rajput. “Why Android Studio Is Better For Android Developers Instead Of Eclipse - DZone Mobile.” 20th May 2015. [Online]. Available: https://dzone.com/articles/why-android-studio-better. [Accessed: 15-Mar-2019].

[11] Gradle. “Gradle Build Tool,” *Gradle*. [Online]. Available: https://gradle.org/. [Accessed: 15-Mar-2019].

[12] Amazon. “AWS Documentation.” [Online]. Available: https://docs.aws.amazon.com/index.html#lang/en\_us. [Accessed: 13-Nov-2018].

[13] Bob Violino, “How to choose the right NoSQL database,” *InfoWorld*, 09-Mar-2018. [Online]. Available: https://www.infoworld.com/article/3260184/how-to-choose-the-right-nosql-database.html. [Accessed: 24-Mar-2019].

[14] W. Li, C. Yen, Y. Lin, S. Tung, and S. Huang, “JustIoT Internet of Things based on the Firebase real-time database,” 2018.

[15] *Firebase*. “Firebase API Reference.”23rd Feb 2019 [Online]. Available: https://firebase.google.com/docs/reference/. [Accessed: 13-Nov-2018].

[16] Revolut. “Round up your spare change and save with Vaults.” *Revolut Blog*, 17-Apr-2018. [Online]. Available: https://blog.revolut.com/round-up-your-spare-change-and-save-with-vaults/. [Accessed: 13-Nov-2018].

[17] Michael Moroney, “The Myth of Working Hard vs. Working Smart,” *Entrepreneur*, 27-Dec-2013. [Online]. Available: https://www.entrepreneur.com/article/230527. [Accessed: 12-Apr-2019].

[18] Megan Stonecipher. “8 apps that can help save you money - SpunOut.ie - Ireland’s Youth Information Website.” 12th July 2018. [Online]. Available: https://spunout.ie/life/article/8-apps-that-can-help-save-you-money. [Accessed: 13-Nov-2018].

[19] VISA. “Visa B2B Connect | Visa Developer Center.” [Online]. Available: https://developer.visa.com/capabilities/vba/docs. [Accessed: 13-Nov-2018].

[20] Donal O,Mahony, Michael Peirce, Hitesh Tewari “mmn7.pdf,”. [Online]. Available: http://www.medien.ifi.lmu.de/lehre/ws0910/mmn/mmn7.pdf. [Accessed: 19-Mar-2019].

[21] PayPal. “Sandbox Overview - PayPal Developer.” [Online]. Available: https://developer.paypal.com/docs/classic/lifecycle/sb\_overview/. [Accessed: 13-Nov-2018].

[22] Braebot. “paypal/PayPal-Android-SDK: Accept PayPal and credit cards in your Android app.” 25th January 2017. [Online]. Available: https://github.com/paypal/PayPal-Android-SDK. [Accessed: 19-Mar-2019].

[23] Dwolla inc. “Guides | Dwolla API.” [Online]. Available: https://developers.dwolla.com/guides/. [Accessed: 13-Nov-2018].

[24] Stripe. “Stripe API Reference.” [Online]. Available: https://stripe.com/docs/api/authentication. [Accessed: 19-Mar-2019].

[25] W. Chan Kim & Renée Mauborgne “What is Blue Ocean Strategy | About Blue Ocean Strategy,” *Blue Ocean Strategy*

[26] Manali Oak. “Advantages of Relational Databases,” 17th Feb 2018. *Techspirited*. [Online]. Available: https://techspirited.com/advantages-of-relational-databases. [Accessed: 19-Mar-2019].

[27] Ruben Verborgh *et al.*, “Survey of Semantic Description of REST APIs,” in *REST: Advanced Research Topics and Practical Applications*, C. Pautasso, E. Wilde, and R. Alarcon, Eds. New York, NY: Springer New York, 2014, pp. 69–89.

[28] Margaret Rouse. “What is RESTful API? - Definition from WhatIs.com,” March 2019 *SearchMicroservices*. [Online]. Available: https://searchmicroservices.techtarget.com/definition/RESTful-API. [Accessed: 19-Mar-2019].

[29] Natalia Chrzanowska. “Node.js vs. PHP: Which Environment To Choose For Your Next Project? | Netguru Blog on Node.js.” 7th Sept 2017 [Online]. Available: https://www.netguru.com/blog/nodejs-vs-php. [Accessed: 19-Mar-2019].

[30] Rajat Harlalka, “Choosing the Right Machine Learning Algorithm,” *Hacker Noon*, 18-Jun-2018. [Online]. Available: https://hackernoon.com/choosing-the-right-machine-learning-algorithm-68126944ce1f. [Accessed: 20-Mar-2019].

[31] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “KNN Model-Based Approach in Classification,” in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, 2003, pp. 986–996.

[32] Avinash Navlani. “KNN Classification using Scikit-learn,” *DataCamp Community*, 02-Aug-2018. [Online]. Available: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn. [Accessed: 20-Mar-2019].

[33] Philip H. Swain and Hans Hauska, “The decision tree classifier: Design and potential,” *IEEE Trans. Geosci. Electron.*, vol. 15, no. 3, pp. 142–147, Jul. 1977.

[34] Smart Draw. “Decision Tree Maker | Free Online App, Templates & Download.” [Online]. Available: https://www.smartdraw.com/decision-tree/decision-tree-maker.htm. [Accessed: 20-Mar-2019].

[35] “Linear Regression.” [Online]. Available: http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm. [Accessed: 20-Mar-2019].

[36] Peter Flom. “The Disadvantages of Linear Regression,” *Sciencing*. [Online]. Available: https://sciencing.com/disadvantages-linear-regression-8562780.html. [Accessed: 20-Mar-2019].

[37] [Gujarati, Damodar N.](https://en.wikipedia.org/wiki/Damodar_N._Gujarati); Porter, Dawn C. (2009). Basic Econometrics (Fifth ed.). New York: McGraw-Hill/Irwin. pp. 73–78

[38] Julie Legler, Paul Roback. "Broadening your statistical horizons, Generalized Lineear Models and multilevel models" 29th Jan 2019.

[39] Alexander Katz, Andy Hayes, Tejas Suresh. "Poisson Distrobution". Brilliant.org. 4th Feb 2019.Available: "<https://brilliant.org/wiki/poisson-distribution/>"

[40] “Poisson Exp.” [Online]. Available: https://www.statstodo.com/Poisson\_Exp.php. [Accessed: 25-Mar-2019].

[41] Pennsylvania State University“Lesson 9: Poisson Regression | STAT 504.” [Online]. Available: https://newonlinecourses.science.psu.edu/stat504/node/165/. [Accessed: 20-Mar-2019].

[42] XMLSoccer. “XMLSOCCER.COM - Stable and affordable XML feed.” [Online]. Available: http://xmlsoccer.com/Default.aspx. [Accessed: 13-Nov-2018].

[43] ApiFootball. “APIfootball - About Us.” [Online]. Available: https://apifootball.com/about-us/. [Accessed: 13-Nov-2018].

[44] Heroku. “What is Heroku | Heroku.” [Online]. Available: https://www.heroku.com/what. [Accessed: 11-Apr-2019].

[45] “Software Development Methodologies.” [Online]. Available: http://www.itinfo.am/eng/software-development-methodologies/. [Accessed: 21-Mar-2019].

[46] Management. “Crystal Clear Methodology,” 07-Feb-2018. [Online]. Available: https://newline.tech/blog/crystal-clear-methodology/. [Accessed: 11-Apr-2019].

[47] “12 Best Software Development Methodologies with Pros & Cons,” *Web Solutions Blog*, 31-May-2018..

[48] Jamsheer K“What is iterative development? - Definition from WhatIs.com,” 1st June 2018. *SearchSoftwareQuality*. [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/iterative-development. [Accessed: 21-Mar-2019].

[49] “Pipe-And-Filter.” [Online]. Available: http://www.dossier-andreas.net/software\_architecture/pipe\_and\_filter.html. [Accessed: 02-Apr-2019].

[50] Jorge L. Ortega-Arjona. “Architectural Patterns for Parallel Programming.” 2005. [Online]. Available: http://lya.fciencias.unam.mx/jloa/PPF/EuroPLoP2005.html. [Accessed: 02-Apr-2019].

[51] “Client-Server Architectures.” [Online]. Available: http://www.cs.sjsu.edu/~pearce/oom/ood/distArch/server.htm. [Accessed: 02-Apr-2019].

[52] V. Mallawaarachchi, “10 Common Software Architectural Patterns in a nutshell,” *Towards Data Science*, 04-Sep-2017. [Online]. Available: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013. [Accessed: 20-Nov-2018].

[53] S. Nidhra, “Black Box and White Box Testing Techniques - A Literature Review,” *Int. J. Embed. Syst. Appl.*, vol. 2, no. 2, pp. 29–50, Jun. 2012.

[54] Guru99. “Functional Testing Vs Non-Functional Testing: What’s the Difference?” [Online]. Available: https://www.guru99.com/functional-testing-vs-non-functional-testing.html. [Accessed: 08-Apr-2019].

[55] D. Quercia, R. Schifanella, and L. M. Aiello, “The shortest path to happiness: recommending beautiful, quiet, and happy routes in the city,” in *Proceedings of the 25th ACM conference on Hypertext and social media - HT ’14*, Santiago, Chile, 2014, pp. 116–125.