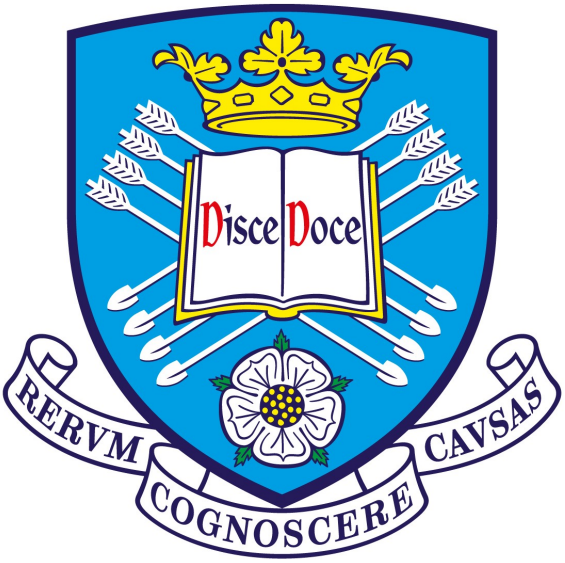
University of Sheffield



Department of Computer Science

Personal Money Manager using A.I.

Alexander Morris
Supervisor: Olakunle Olayinka

May 12, 2020

This report is submitted in partial fulfilment of the requirement for the degree of MComp Computer Science

Declaration

"All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole."
- Alexander Morris.

Abstract

When students first start university many of them have very little experience in managing their money. They have financial responsibility thrust upon them and are suddenly balancing paying for food, rent and social activities while staying afloat between loan payments. Nowadays most students have access to and regularly use smartphones, allowing mobile applications to be used anywhere and making them powerful tools for improving many aspects of life.

The aim of this project was to create a mobile application to aid students in managing their personal finances. This report details an exploration of mobile apps and their development, reviews current solutions to the problem and the prospect of using A.I to solve it. It then details the implementation of a new application, which is evaluated and a number of areas of improvement are identified. These could be used in further development of the application or to aid in the development of other solutions.

A demo of the finished application can be found at https://www.youtube.com/watch?v=Aae-jQg6bjgfeature=youtu.be

COVID 19 Impact statement

The lockdown imposed because of COVID-19 caused additional challenges for the completion of this project. In the second semester of the project, the university switched to online delivery of all teaching, and university buildings were closed. All project meetings were shifted to email correspondence and video meetings.

In addition, my project plan was revised because I could no longer conduct the in-person user testing that as was originally planned.

Acknowledgements

I would first of all like to thank my supervisor Olakunle Olayinka for the excellent support and feedback given throughout the project. I would also like to thank friends and family for their support throughout my time at university. Finally I would like to thank all the participants who took part in the testing and evaluation stages of the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

When first coming to university most students have financial responsibility thrust on them for the first time in their lives. Many don't have the financial skills necessary to face this, with studies putting student financial literacy at around 30% [1]. A lack of financial literacy results in poor financial wellbeing [2] with students struggling to manage their finances effectively and ending up in debt; a large 2019 survey by saveTheStudent.org [3] found that 49% of students end up using an overdraft to cover their costs.

Financial difficulties and the stress related to them have an extremely adverse effect on the physical and mental wellbeing of students. Physical health suffers as paying for healthy food and activities becomes a struggle [4], many students also say that worrying about money disrupts their sleep [3]. Because of sleep deprivation and through feeling distracted and hopeless, a student's motivation and productivity can be affected and their grades suffer as a result. Students' can struggle to pay for social activities so they can become isolated and their relationships with friends and partners can be impacted. This can all trigger or contribute to mental health conditions such as anxiety and depression. [5]

1.2 Objectives

The main aim of this project is the creation of a tool to aid students in managing their personal finance.

Developing a mobile app was chosen as a solution to the problem. The app will feature tools that let the user generate a personal budget and help them stick to it. Creating a budget is an essential step in taking control of personal finances [6], but despite this only 1/6 students claim to have used one before [3]. It is hoped that the app will make budgeting easy and simple, encouraging more students to try it. The possibility of using A.I techniques to aid in personal finance management will also be explored.

1.3 Report Overview

Chapter 2 outlines research done into the development of mobile applications, reviews existing personal finance apps, explores the use of A.I in finance and discusses the results of a pre-requirements survey.

In **Chapter 3** the requirements of the application are analysed.

Chapter 4 describes the process of designing the app.

Chapter 5 presents the technical details of how the application was implemented.

Chapter 6 discusses the evaluation and final results of the project, as well as outlining any further work that could be done.

Chapter 7 summarises the report and highlights the main achievements of the project.

Chapter 2

Literature Report

This chapter looks into mobile apps and some of the details and challenges involved in their development. An analysis of existing personal finance apps is also conducted as is research into the use of A.I in finance. Finally, the results of a pre-requirements survey is presented.

2.1 Mobile Applications

From their first conception in 1992 smartphones have come a long way, in the past few decades they have transformed from large and impractical luxury items to small and powerful devices that most people can't live without [7]. As smartphone technology has evolved it has become more advanced and powerful but also cheaper and more accessible. As of 2019 the majority of young adults, and therefore students, own a smartphone, with studies showing that 96% of people in the 16-34 age bracket own a smartphone. [8].

A mobile application, commonly referred to as an app, is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Mobile applications aim to provide users with similar services to those accessed on personal computers [9]. On early smartphones mobile apps filled simple functions such as writing notes, playing music and sending emails. They were developed by the phone's manufacturer and came preinstalled, or were downloaded with difficulty using wireless application protocol (WAP). [10]. The introduction of the iPhone was a major step in realising the potential of mobile apps, allowing users to easily download apps off of its 'App Store'. It enabled 3rd party app development, resulting in a surge of new applications and an explosion of the mobile development industry [11]. As of 2019 there are a combined 4.27 million apps available on the App Store and its largest competitor Google Play [12]. Improved hardware including better processing, access to the internet and improved displays have allowed for more powerful and useful apps that can assist with all aspects of life; apps that help with social media, navigation, entertainment and transport apps are amongst the most popular on today's market [13].

The functionality and power that modern apps possess, their ease of use and the prevalence of smartphones amongst students makes them an attractive choice for creating a tool that tackles the personal finance problem.

2.2 Developing a Mobile Application

This section discusses some of the different technology and design choices that need to be taken when developing a mobile application, covering the advantages and

disadvantages of different options.

2.2.1 Mobile Platforms

Currently there are 2 major platforms for developing applications for mobile devices; Android and iOS, who as of October 2019 own 99.61% of the UK mobile market between them [14].

The iOS operating system was designed by Apple for use with their mobile devices such as iPhone, iPod and iPad. It currently holds 49.2% of the UK market share. iOS applications are developed in the Xcode development environment, which requires Mac OSX operating system [15]. Advantages of iOS development include faster development and a more affluent audience and hence higher revenue. [16]

Android is a platform developed by the Open Handset Alliance, a consortium of firms looking to create open standards for mobile devices, with the main contributor and marketer being Google [17]. It is currently the market leader when it comes to mobile development sliding in just ahead of iOS with a market share of 50.41%. Applications are usually developed using the Android SDK [**17**] which can be run with any desktop operating system. Android is open source and so has more flexibility with regards to customizing your app and building the features and functions [16]. Getting an application approved for the Android app store; the 'Play Store', is also a much easier process than for iOS and they also allow apps to be privately hosted without any approval from Google. [18]

2.2.2 Development Approach

When developing an application for a mobile device there are three possible approaches; creating a web-based application, a native application or a hybrid of the two. [19]

Web apps are built in JavaScript, CSS, and HTML5 and run inside a web browser. They allow a wide user base to be reached as they do not depend on the operating system of the mobile device. As they require a web browser to run them the device has to be connected to the internet and are slower and less responsive as they need to load images and functions through the network. They also have less access to the devices features and hardware. [20]

Native apps are built for a specific platform or operating system and are written using the appropriate software development kit (SDK) and language for that platform. The application will only be available on the selected platform and therefore a reduced audience is reached. Despite this native applications are the most common due to several advantages; they provide the best performance, are available when the device is offline and receive guidelines, marketing and support from the app stores related to the platform.[21]

The final approach is a combination of the two; a hybrid app is a web app wrapped in a native shell. It allows the application to work over multiple platforms but is dependent on 3rd party platform to deploy the application. They are still slower than native applications and only have some functionality offline but do allow the access to the devices' hardware.

2.2.3 Programming Language for Android Development

If the chosen platform for an app is Android and a native approach for developing is chosen, the next step is choosing what programming language to write the program in.

When choosing which language to write an Android application with there are two main contenders to consider; Java and Kotlin are both the official languages of android development. There are other options, but these require development outside Android Studio. [22]

Java is the original and most popular language for Android development, because of this there is plenty of support and resources for developing applications using it. Java has a large number of free libraries to complement your code and many online tutorials to help learn the language and solve common problems. Java's main issue is its complexity, it isn't very readable due to large amounts of boilerplate code [23] and has complications that can make development tedious. [22]

The up-and-coming alternative Kotlin is Androids secondary official language since 2007, although as of 2019 Google has said that it is now their preferred language and that "Android development will become increasingly Kotlin-first" [23]. Similar to Java, Kotlin runs on the Java Virtual Machine (JVM). It aims to streamline code, improve coding experience and is simple to pick up and use. It also enables mixed language projects as it interpolates with Java and can compile to JavaScript and Native. This however comes at a slight cost as Kotlin programs generally achieve slightly worse results on performance benchmarks when compared to Java [24]

2.2.4 Security

Applications that handle personal information and sensitive financial data are required to follow strong security standards to protect that data, in this section I will describe some of the principles of information security and techniques that are commonly used to do this.

Information security is usually defined as the confidentiality(C), integrity(I) and availability(A) of information, referred to as the CIA triad. Where confidentiality means keeping information from being accessed by unauthorized parties. Integrity involves stopping information being modified by unauthorized parties and availability means ensuring that information is available to authorized parties.

The identify-authenticate-authorize(IAA) sequence is one of the main controls aimed at protecting the CIA triad. First the identity of the user is discovered and undergoes authentication. Commonly done by requiring the user to provide a username and password combination [25]. Using the authenticated identity access control can be used to limit their actions and the data that they have access to.[26].

The weak link in the chain of the IAA sequence is the password strength. If the account username is known an attack can be performed by recursively trying to guess the password using dictionaries of frequently used words or probabilistic models. If they are successful then they will gain unauthorized access to information and gain privileges to tamper with it [27].

Passwords based on words or phrases are easy to guess, but also easy to remember so are preferred by most users. To force users to create strong passwords a set of rules can be enforced requiring them to make their passwords be above a minimum length and include capitalisation, numbers and symbols.

Communicating information with an online server also poses a risk to the triad by opening up the application to 'man-in-the-middle' attacks; where an attacker attempts to intercept information as it passes between the client and the server. To protect against it should be ensured that any communications is done through an encrypted channel. This can be done using the HTTPS protocol, data sent using this protocol is protected with TLS which provides 3 layers of protection – Encryption

and integrity; data cannot be modified during a transfer without detection, and authentication; proves that your users communicate with the intended website. [28]

There is also the possibility that the contents of an applications database to be leaked or obtained by unauthorised parties. To ensure that this this information cannot be easily used the data should be stored in an encrypted form. Handling confidential financial data requires that the strength of encryption be high, to ensure that it is sufficient an industry-tested and accepted algorithm such as RSA(at 2048 bits or higher) or AES(128 bit or higher) should be used [29].

2.3 Open Banking

Over the past few decades banks have been moving from paper statements to internet banking; hosting the customer data online and allowing customers to instantly view an updated list of their spending and finances using a computer or mobile device.

Open banking is a banking practice that provides third-party financial service providers open access to consumer banking, transaction, and other financial data from banks [30]. It takes advantage of internet banking to allow financial information to be securely shared with approved providers [31]. Open banking has opened an opportunity for new services to be developed that can use customers financial data to provide helpful tools for managing finances [32]; it has opened the door for mobile applications that can access account data directly.

To become an approved provider and use Open Banking a developer must first gain regulatory permissions from the Financial Conduct Authority or similar financial regulatory body and then apply to enrol to the Open Banking Directory [33]. This is process is virtually impossible for an amateur developer, however there is an alternative in the form of Open Banking compliant application programming interfaces. TrueLayer is one of these interfaces, allowing developers to bypass regulation and providing easy access to open banking. [34]

2.4 Review of Personal Finance Applications

This section introduces and analyses some of the most popular personal finance applications currently available with the aim of trying to identify useful features for the development of a new solution and what functionality is currently lacking amongst current solutions.

2.4.1 Emma - Budget Planner and Money Management

Launched in 2018 by Emma Technologies LTD, Emma is a personal finance manager that takes advantage of open banking to track a user's spending and provides a selection of features to aid with saving money.

The app allows multiple bank accounts to be tracked through open banking, allowing the user to conveniently view all their finances in one place. The financial data is analysed, identifying regular payments/subscriptions and automatically allocating spending into categories so that the user can see where their money is being spent. These can be then be manually adjusted so that unknown transactions can be assigned a category and missed subscriptions can be added. Some graphs and charts can be viewed to better understand the data and see past spending and income.

A budget can be manually created for a set time period, this can then be allocated into categories. The app will notify the user if they have gone over budget in a specific category.

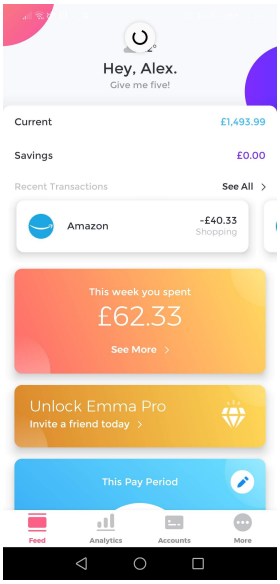


Figure 2.1: Screenshot of Emma's home screen

Emma is an easy to use solution for improving personal finance, the ability to clearly see spending and budget around that is extremely useful, especially to students who often struggle to prioritise their spending. There are however some essential features missing that would greatly benefit students; there is no way of adding future income or transactions, useful for budgeting around student finance or setting aside money. Although the app recommends a starting budget this is based on your average monthly spend and doesn't consider saving goals, income or future payments and so a proper budget has to be calculated manually, something many students struggle with. It also doesn't allow the manual input of financial data, so if your bank does not support open banking or you have cash savings then there is no way of tracking these. [35]

2.4.2 Money Dashboard Budget Planner

Developed in 2009 Money Dashboard was the UK's first personal finance application. Despite being such an old application, it has kept up with the times and now uses open banking to allow bank accounts to be tracked in real time. It provides users with the ability to view all of their online financial accounts in one place and categorises and analyses all of their transactions so they can understand how they use money.

The app displays overview of your account spending by category as a simple to understand bar chart and allows the creation of budgets based on category. It also has a planner feature, which shows the history of your bank balance and allows future payments and income to be added.

Similar to Emma a good knowledge of your spending and saving is required to make use of the budget feature. The app also does not allow the manual input of bank accounts or financial data. It does allow the input of upcoming transactions and displays a future balances based on this, however this feature has no integration with the rest of the app. [36]

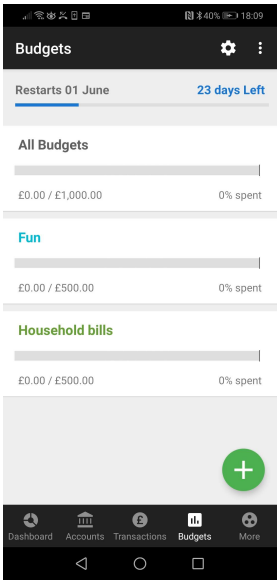


Figure 2.2: Screenshot of Money Dashboard's budget screen

2.4.3 YNAB

You Need A Budget (YNAB) is a budgeting app that provides a straightforward system to help you understand your income and outgoings and plan ahead. It allows account linking in the US and Canada, although this feature is not available in the UK. It also allows past and future transactions to be manually entered. The user can create a monthly budget and allocate this into categories, it will then tell the user if they've overspent in a category and allow them to reallocate their budget to mitigate the overspending. YNAB also provides a host of features to help people budget such as savings goals, an overview of their net worth and an 'age of money' metric that helps show how well you're saving. The basic features of YNAB and its

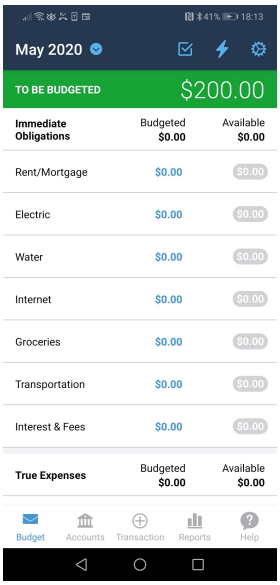


Figure 2.3: Screenshot of YNAB's budget screen

approach to budgeting work really well, the idea of reallocating money from other parts of your budget is a great way of prioritising your spending and the ability to set saving goals/upcoming payments is a huge help. The ability to manually add data is

really useful and quite rare since the open banking revolution. One problem with the application is its complexity, the U.I is fairly clunky and hard to navigate and some of the features are really complicated to use; so much so that the creators provide a huge amount of resources, including a book, that explain the use of the application. Most students will not need a large amount of the functionality available in YNAB and its complexity is definitely a daunting thing for the less financially literate. [37]

2.4.4 Cleo

Cleo is an A.I powered chatbot that helps analyse account data and provide financial advice. Making use of open banking to view user's account data it allows the user to query their financial data by typing questions to the chatbot. Cleo provides useful graphs and analytics on request to help the user understand their finances and allows the user to set a budget, providing advice on whether the user can afford things based on their remaining budget.

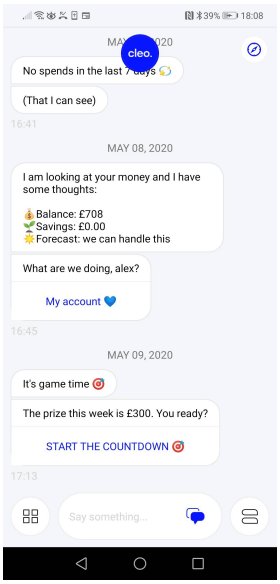


Figure 2.4: Screenshot of Cleo's chatbot

The chatbot is a novel feature, which appeals to some people. It does however makes viewing information slower and less intuitive than a well-made interface. Cleo allows a long-term budget to be set, which is useful for students trying to save throughout the term, and also tries to give a recommendation based on your average spending and bills, making it easier for people who are new to budgeting. The prediction is basic, but still useful for giving users an idea of how they are performing against their budget and the spending advice is also useful. The lack of a manual input option is frustrating, as some banks do not allow linking with the application. [38]

2.5 Artificial Intelligence

This sections explores literature on uses of A.I. techniques in finance, it was decided to focus on machine learning techniques as these are well suited to the problem.

2.5.1 Machine Learning

Machine learning is a subset of Artificial Intelligence that involves using computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment [39]. There are two main categories that machine learning algorithms fall under: supervised and unsupervised learning. [40]

In unsupervised learning there is no training data set and the outcomes are unknown, it aims to find the natural structures present within the data. There are two types of algorithms for handling different types of output; Clustering aims to order data into discrete groups. Dimensionality reduction involves learning the relationships between data points so that we can reduce unimportant ones and represent the data with less features.

Supervised learning involves the use of a 'ground truth'; we have prior knowledge of what the output samples should be. It aims to learn a function that best approximates the relationship between input and output observable in the sample of data. A training data set is used to 'teach' the algorithm, the quality of the model relies heavily on this data and noisy or incorrect data will lead to a less effective function being learned. There are two main problems that can be solved with supervised learning algorithms. Classification, which involves mapping the input data to a discrete output or 'Labels' and regression; mapping the input to a continuous output.

2.5.2 Supervised Machine Learning Algorithms in Finance

Given the amount of data available and the quantitative nature of the financial world it is well suited to the use of machine learning, in particular supervised learning techniques.

In J. De Spiegeleer et al's 2018 paper [41], they demonstrate the use of Gaussian process regression (GPR) to solve classical quantitative finance problems; problems involving the mathematical modelling of financial markets. GPR is a machine learning algorithm based that uses a Bayesian inference approach. Imagine the problem as a data set of observations, vectors x_i each with a corresponding output/answer y_i the goal is to learn the function f where:

$$y_i = f(x_i) + \varepsilon_i$$

ε_i represents the noise in the data.

Using a Bayesian approach, f is created by specifying a prior distribution and combining it with the observed datapoints based on the observed data (the training set) using Bayes Rule, $P(A \mid B) = \frac{P(B|A)\,P(A)}{P(B)}$, to condition a joint distribution on the observations. GPR is a nonparametric (i.e. not limited by a functional form) take on the approach where $f(x)$ is a possible infinite collection of random variables, any finite subset of it having a joint Gaussian distribution. J. De Spiegeleer et al showed that the technique can be used effectively to solve curve fitting, derivative pricing and hedging problems at a much greater speed and with relatively small errors compared to calculating the values or using traditional simulations/models.

Machine learning can be used to aid with portfolio management, this involves analysing the stock market and making decisions on whether to buy company stocks. Huseyin Ince and Theodore B. Trafalis [42] investigated the use of support vector machines (SVMs) to give recommendations on whether or not to buy a company's

stock. SVMs are a supervised learning algorithms that given a training set consisting of vector inputs x_i with N dimensions/features and outputs y_i, tries to find the hyperplane in N-dimensional space that distinctly classifies the points and has the largest margin; the maximum distance between data points of both classes. This is then used to classify real data.

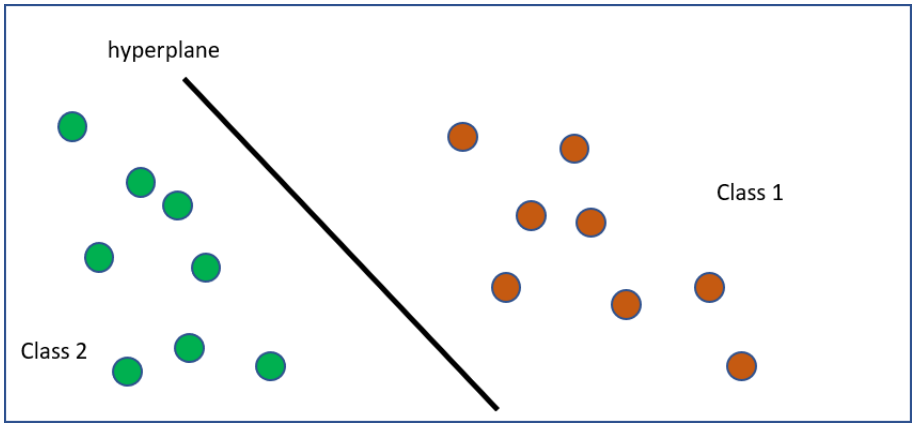


Figure 2.5: 2D SVM hyperplane

Huseyin Ince and Theodore B. Trafalis results showed that the SVM method heavily depends on which features were chosen for differentiating the data and that it works well for identifying which stock should be traded provided the correct parameters are chosen.

Artificial Neural Networks (ANNs) is a process designed to mirror the human brain, layers of highly interconnected processing elements or units communicate with each other similar to neurons in the brain, making a series of transformations on data to generate an understanding of it. Generally, an ANN can be envisioned as 3 main layers; an input layer that receives the data; the hidden layer that learns the mapping between the input and output in the training data through series of mathematical transformations; and the output layer, that is responsible for providing the output. All the units in each layer are connected to the units in each adjacent layer, This is visualised in figure 2.6

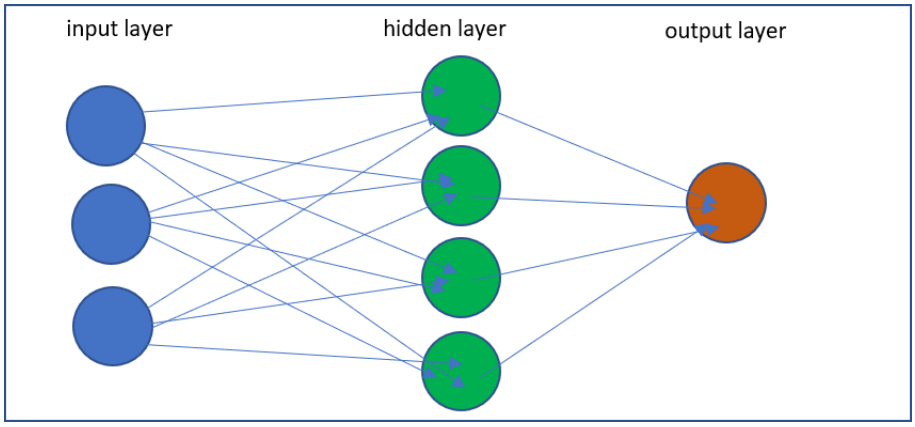


Figure 2.6: Interconnected layers in an artificial neural network

Kyoung-jae Kim [43] demonstrated the use of ANNs for financial forecasting by predicting stock market prices, the focus of his study was experimenting with different instance selection methods, proving that ANNs performs well on the problem

when combined with genetic based learning algorithms.

There is not much academic research into using A.I. to solve personal finance problems, however there is definitely scope for the above approaches to be applied to the area. This has been explored by some commercial services; with 'A.I personal assistants' becoming more and more prevalent [44].

2.6 Pre-Requirements Survey

Before finalising the requirements of the application a survey was conducted to provide a clearer picture of the financial situation of most students, discern demand for the app and to help decide what features should be included. A copy of the questions can be found in Appendix C.

Conducting the survey was approved by the University of Sheffield Ethics Approval Procedure, a copy of the official approval document can be found in Appendix A. Volunteers were provided with an information sheet that explained the process and were required to sign a consent form before participating, these can be found in Appendix B. Thiry-nine volunteers completed the survey.

The survey affirmed worries that some students struggle with financial matters with over 40% agreeing that it was a source of worry for them. The survey found that an astounding 66% of participants do not budget and a further 18% struggle to stick to one.

The demand for a mobile budgeting application seemed fairly high, with 69% of participants voting 4/5 or 5/5 when asked how helpful they would find one. 40% of current budgeters stated that they currently use pen and paper techniques or a spreadsheet to manage their finances, despite this most agreed that they would still find a budget app helpful.

A draft set of requirements was drawn up based on the research earlier in the chapter, volunteers were then asked to rate how helpful they would find each one. Participants were also given the opportunity to add any extra features they would like to see, notable suggestions include; allowing regular expenditure to be defined and graphically representing weekly spending.

Other useful insights gained from the survey were that 59% of volunteers use an Android device and that an almost equal amount of people prefer budgeting around both weekly or monthly periods, so an ideal solution would allow both.

Chapter 3

Requirements and Analysis

This chapter covers the problem that is to be solved by my application, discussing the aims and objectives, identifying the stakeholders and presenting a list of functional and non-functional requirements. It will also cover some of the technology that will be used to implement the application and explain the testing and evaluation process.

3.1 Aims

This project will aim to create an application for Android devices that can aid students with personal finance management. The application should facilitate saving without requiring large amounts of financial knowledge, enabling users to understand their finances and easily create a budget. User's performance against their budget will be tracked and they will be notified if they overspend or if they are predicted to go over budget; calculated by analysing their spending from previous periods and their current spending. The application will allow bank account and spending information to be tracked using both open-banking and manual input.

3.2 Stakeholders and Target Audience

The application will be designed with students as the primary audience and the features of the application will aim to reflect that; catering for the student financial lifestyle and not including features that would be unnecessary and overly complicated. Despite this the application will still be useful for anyone trying to improve their personal finances especially young adults who are looking to master the basics of budgeting and saving.

A stakeholder is an individual or group who will be affected by the project, with some vested interest in the quality of the final product. In this project there are only two main stakeholders; the user of the product and the developer. The users will benefit from a well-made product that can help them manage their finances. The developer of the product has an interest in its marketability and also the security of the application, as any breach in information could lead to legal action against them. Finally, the banks who participate in open banking are a potential secondary stakeholders as the app will be requesting account information from them.

3.3 Requirements

The functional and non-functional requirements of the application have been developed based on insights gained during the literature review and pre-requirements survey. They are presented as a set of user stories. Each user story is given a priority based on how critical they are for the application. Stories that are critically important for the final product that will definitely be implemented are labelled Must Have(M), requirements that are useful for the product and will be implemented if development goes to plan are labelled Should Have(S) and optional requirements that will be implemented if development is ahead of schedule are labelled Could Have(C). The priority of each requirement was informed based on votes received during the survey.

3.3.1 Non-Functional Requirements

The non-functional requirements in table 3.1 describe the qualities of the application, representing a set of standards to judge the performance of the system.

User Story	**Priority**
As a user I want to be able to run the product on all android devices so that I can use the application on my device.	M
As a user I want my data to be stored securely so that it is protected.	M
As a developer I want my app to be responsive and fit to the device size so that the user experience is enjoyable.	S
As a developer I want my app to be thoroughly tested and bug free so that the user experience is enjoyable.	S
As a developer I want my app to run quickly on the intended hardware so that the users stay engaged.	S
As a user I want to be able to easily navigate and understand the application so that I can use it without prior information.	S

Table 3.1: Non-functional requirements

3.3.2 Functional Requirements

The functional requirements in table 3.2 outline the desired features for the application. They define the components of the system and describe the functions the software must perform. Notes have been added on some of the stories to clarify their purpose or add detail.

User Story	Notes	Priority
As a user I want to be able to manually add financial data so that I can keep track of my money.	Allow bank statements to be input into the program using csv files. Allow individual transactions to be added to the program using a form.	M
As a user I want to be able to link my bank accounts with the app so that I can track them automatically.	Allow the app to be linked to bank accounts using open banking.	M
As a user I want to be able to view my spending in categories so that I can understand where my money is going.	Automatically allocate the user's spending into categories based on information stored in a database, if the transactions/merchant isn't recognised then the user should be able to manually categorise it.	S
As a user I want to be able to view my account information and balances so that I know how much money I currently have.	Display all the user's financial data in an easy to read and intuitive format, showing total net worth and previous transactions.	M
As a user I want to be able to see my bills and subscriptions so that I can manage them.	Analyse the users bank data to identify subscriptions and bills and display them.	C
As a user I want to be able to set saving goals so that I can budget around them.		S
As a user I want to be able to input upcoming payments and bills so that I can budget around them.	Allow the user to manually input upcoming payments. One off payments and regular payments.	S
As a user I want to be able to input future income so that I can budget around it.	Allow the user to manually input future income so that they can be managed around. Regular payments and one-off payments.	S
As a user I want to be able to create and edit my budget so that I can save money.	Allow the user to manually create set a budget.	M
As a user I want to be able to generate a budget based on my goals so that I can save money.	Automatically generate a budget using the users saving goals and taking into consideration future income/payments.	S
As a user I want to be able to distribute my budget into categories, so that I can prioritise my spending.		M
As a user I want to be able to see my performance against my budget so I can adjust my spending.	Display the amount of the users budget they still have left to spend in the current period and the amount of days left.	M
As I user I want to be able to see my past spending and budget performance so that I can understand how my spending is changing.	Save budget and spending information and allow past information to be viewed.	C
As a user I want to be notified if I'm predicted to go over budget so I can adjust my spending.	Use machine learning techniques on past spending and analyse current monthly or weekly spend to predict if the user is going to go over budget and notify them if so.	S
As a user I want to be notified if I've gone over budget so that I can reallocate funds between categories and mitigate the damage.	Send a notification in the app and a push notification if the user goes over budget in a specific category, then allow them to rectify the overspend by allocating money from other parts of their budget.	S
As a user I want to be given advice on how to save money so that I can make better saving decisions.	A.I. financial assistant that gives advice on how to save money	C

Table 3.2: Functional requirements

3.4 Technology

It was decided to develop a native application to allow the application to be fully functional offline and because the application will involve processing and storing large amounts of data. A native approach will offer the best performance and result in a faster and more responsive application. Android is the chosen platform, due to the results of requirements survey and as developing an iOS application required access to a Mac OSX machine. Kotlin was chosen as the programming language as it is designed to make the coding experience more intuitive which is believed will help speed up the development process.

As it is difficult to acquire FCA approval to use Open Banking directly the TrueLayer API will be used to allow the linking of bank accounts to the app. TrueLayer has a free development package that allows apps to be developed and tested in a sandbox environment, allowing the app to be tested using mock account data.

3.5 Testing Plan and Evaluation

To ensure that the final application is of a high quality and bug free a range of testing measures will be undertaken throughout the development process. Further testing will be completed at the end of the process to evaluate whether the finished product has met the aims of the project.

- **Unit Testing** will ensure that individual components of the software perform as designed by testing them in isolation
- **Integration Testing** will used to expose any defects in the interaction between components.
- **System Testing** will be taken to verify that all the requirements have been met and function correctly.
- **Compatibility Testing** will aim to ensure the final product works correctly across all Android devices.
- **Performance Testing** will be undertaken to evaluate the responsiveness of the application and how much of the device's resources it requires.
- Finally, to evaluate the success of the project volunteers will be recruited from amongst fellow students to test and provide feedback on the final application. **Usability Testing** will be completed; where users are observed attempting to complete tasks with the application to test its ease of use and find flaws in the UI. Volunteers will also be asked to download the application on to their devices and test some of the features, they will be asked to compete a questionnaire based on their experience and will be given the option to provide feedback and suggest possible changes.

3.6 Risk Analysis

As this is a complex project with a long development process it is important to be aware of the risks that might affect its outcome and mitigate them if possible. To this effect table 3.3 below has been constructed to identify some of the major risks associated with the project, analyses their likelihood and severity and states any measures that will followed to avoid them.

Description	**Likelihood**	**Severity**	**Risk Level**	**Mitigation**
Lack of development time or experience resulting in the application not being finished.	Likely	Extremely Severe	High Risk	Follow a structured development plan and prioritise essential features.
Lack of volunteer users resulting in the project evaluation not being thorough.	Likely	Severe	High Risk	Leave plenty of time for user testing and explore many different channels to recruit subjects.
Application crashing.	Likely	Severe	Medium Risk	Write extensive tests for the application.
Non-intuitive interface resulting in a hard to use app.	Unlikely	Severe	Medium Risk	Aim to keep the UI as simple as possible and undergo usability testing to find any flaws.
Loss of data	Unlikely	Severe	Medium Risk	Use version control and store a backup of project data.

Table 3.3: Risk Analysis Table

Chapter 4

Design

In this chapter the focus will be to explain the design choices made for creating the Android application. It will detail aspects of the system design including the architecture of the application and the database design, and it will show the design of the user interface using a series of mock screens.

4.1 Architecture

The recommended architecture for android applications written using Kotlin is based on the Android Jetpack suite; a collection of libraries, tools and guidance that help developers produce high quality code that follow best practices. [45]

Mobile applications comprise of multiple different components that must survive and adapt to complex user-workflows, normally involving cycling between multiple different apps in a short amount of time. As mobile devices are resource constrained at any time, the operating system might kill some of an app's processes to free up resources. This leads to an environment where lots of factors are out of developer control; app components can be launched or destroyed by the system, or the user, at any time and in any order. Because of these issues best practice states that you should only store app or state data in persistent components and that the app components should not depend on each other. Figure 4.1 shows the recommended structure of components.

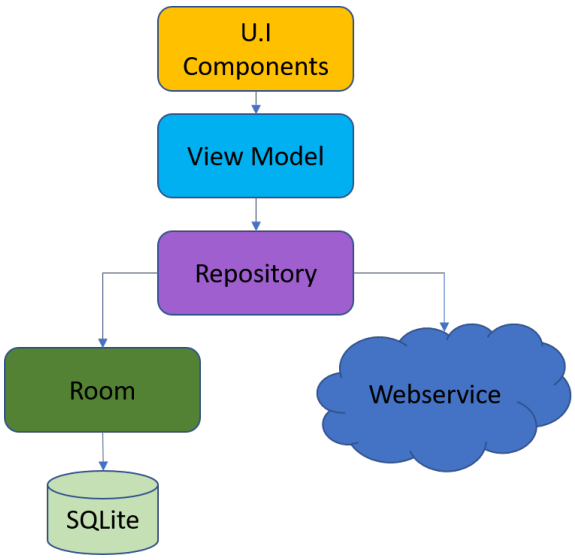


Figure 4.1: Android component architecture

4.1.1 U.I. Components

Activities and Fragments are the two main components that manage the interface in Android applications. An activity is an entry point into the application; when an app is launched or linked from an app an activity will be invoked. Each activity provides a window in which the app draws its U.I and can be seen as a 'screen' of the app. Generally, apps contain multiple activities that are loosely bound, but ideally not dependant, and combine to form a cohesive experience. Fragments are modular sections of an activity that have their own lifecycles and input events.

These components will be used to handle U.I and operating system interactions only, they should not handle any data or database interactions as they can be destroyed by the operating system at any time.

4.1.2 ViewModel

ViewModel's are persistent components that are responsible for handling the data that drive the U.I classes. By storing data in a model many lifecycle related problems are avoided. They are bound to an activity but survive if the activity is paused by the operating system. The data contained in the model is also shared to all the activity's fragments, reducing the amount of calls to the database or webservice needed to get the data for each component.

4.1.3 LiveData

The LiveData class is a useful tool for easily linking an app's data sources to its interface components. It is a data holder class that is linked with an observer and lifecycle owner (fragment or activity component) pair, the observer will be notified when the data is changed, allowing the component to stay consistent with the data source and consistent across your app. LiveData will only notify the observer if it's paired component is in an active state, making efficient use of resources and preventing memory leaks.

4.1.4 Repository

Repositories act as mediators between data sources, providing the view model with data obtained from several different sources and implementations depending on the situation. For example if the app normally retrieves data from a webservice, but is currently offline; the view model will call the repository to retrieve the data, the repository will attempt to retrieve the online data and when that fails it will provide local data instead. This separation of concerns allows for a cleaner API and easier testing.

4.1.5 Room Persistance and SQLite

To handle the local storage for my application an SQLite database will be used in conjunction with the Room persistence library, which provides an abstraction layer above SQLite that allows fluent database access. Room provides a number of advantages over default SQLite; it prevents database injection under intended use [46], it simplifies debugging with compile time verification and reduces the amount of boilerplate code needed to convert between queries and objects [47]

There are 3 major components in Room; the Database which serves as the main access point to your app's relational data, Entities which represent tables within the database and the DAO which contains the methods used for accessing the database.

4.1.6 Database Design

Figure 4.2: Database structure

The database design consists of seven tables, with the User entity acting as a parent to all the other entities, and the Transaction entity being a child of both the Category and Account entities.

User – Stores the account data and current allowance for the app's users, there is a one-to-many relationship between the User and all the other entities.

Account – Represents a bank account or pot of money created by the user or linked using open banking, there is a one-to-many relationship between the Account and Transaction entities.

Category – Spending categories, either default or created by the user. There is a one-to-many relationship between the Category and Transaction entities.

Transaction – Represents transactions made by the user's accounts.

SavingGoal – Represents a User's saving target by a particular date, used for generating a period allowance.

Income – Represents a User's expected income over a number of different dates, used for generating a period allowance.

Bill – Represents a User's expected bills/regular outgoings over a number of different

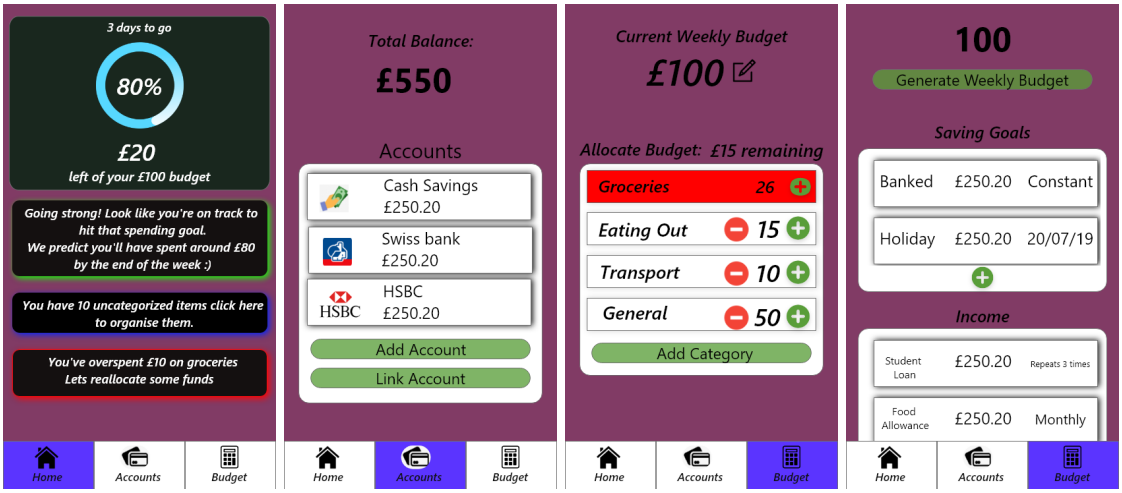
dates, used for generating a period allowance.

4.2 User Interface and Mockups

An important early step in the creation the application was to create a series of UI mockups. These helped the design and implementation process by providing a good view on how the final application would look like, showing what information would need to be displayed and giving an idea of how some of the different features could be presented and interacted with and how they would interact with each other.

4.2.1 Main Screens

The design is centred around 4 main screens, as displayed in figure 4.3



(a) Home screen mockup (b) Account screen mockup (c) Budget screen mockup (d) EditBudget screen mockup

Figure 4.3: Mockup of the Main Screens of the Application

The Home screen gives users an overview of their spending and their predicted spend. It also shows a feed of notifications that provide important updates and guide the user to other parts of the app. A machine learning algorithm will analyse the users spending data, if it is predicted that they are going to exceed their budget then they will be notified here.

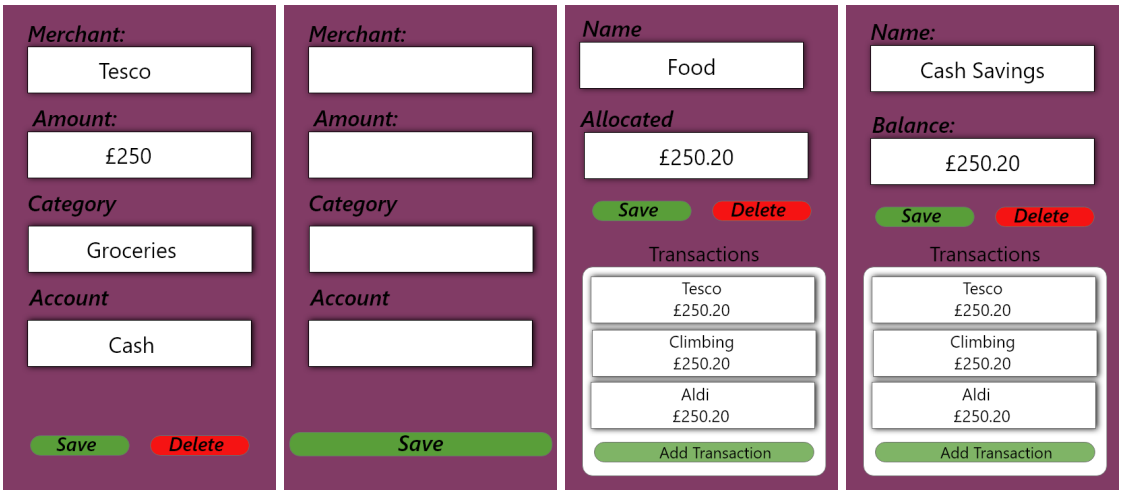
In the Accounts screen the user is presented with their total balance and an overview of any accounts they have created or linked to the app. Selecting an existing account from the list will send the user to the EditAccount screen. A form for creating a new account and linking an account through open banking is done by pressing the respective buttons.

The Budget screen allows the user to allocate their allowance between a set of categories, displaying any categories where they have spent over budget. From this screen users can open a create category form or select an existing category to navigate to the EditCategory screen. Selecting the edit icon beside the current budget opens the EditBudget screen, figure 4.3d. Navigation between the other main screens is performed using the bottom bar which clearly displays the current selection - as shown below in figure 4.4.

Figure 4.4: Nav bar mockup

In the EditBudget screen, figure 4.3d, a list of saving goals and their expected income and bills are shown, by clicking the '+' buttons below each list more entries can be added. Entries can also be clicked to open a form to modify them. This screen will be scrollable so that all 3 lists can be seen. By pressing the 'Generate Budget' button an allowance will be generated for that period, used to manage their spending. The budget will be generated by calculating the amount of money that needs to be saved each week to reach their saving goals and meet all their bills based on their current total balance and expected income.

4.2.2 Input Screens



(a) Edit from mockup (b) Add form mockup (c) EditCategory screen mockup (d) EditAccount screen mockup

Figure 4.5: Mockups of the input forms

Figures 4.5b and 4.5a show an example of the simple forms that will be used to edit items in the database or to create new entries. They will consist of a set of input boxes, where user's can enter the details of each object, and buttons to save or delete the entry. When editing an entry the boxes will be pre-filled with its current details.

The EditCategory and EditTransaction screens also show a list of the selected entity's child transactions. From here a new transaction can be created by clicking the 'Add Transaction' button or a transaction from the list can be selected to open a form to modify it.

Linking an account will redirect the user to an online form handled by TrueLayer.

4.2.3 Log-In and Register Screens

When the app is started the user will be presented with a Log-in screen that takes a username and password input, this will prevent unauthorised persons from viewing the stored data for each user. It will also allow several user accounts to be created

for each device. If the user does not have an account on the device, or wishes to start a new one, then they can navigate to the register screen where new user accounts can be created.

Figure 4.6: Mockup of the Log-in screen

Figure 4.7: Mockup of the register screen

Chapter 5

Implementation and Testing

This chapter details the implementation of the application, following on from the requirements outlined in chapter 3 and based on the design in chapter 4. An overview of the app will be given through a series of screenshots from the mobile application paired with details on how important elements were implemented. The structure of the application and database implementation will be explored in depth as will the TrueLayer integration and A.I. prediction algorithm. To finish, the methods of testing will be shown and discussed.

5.1 Components and Structure

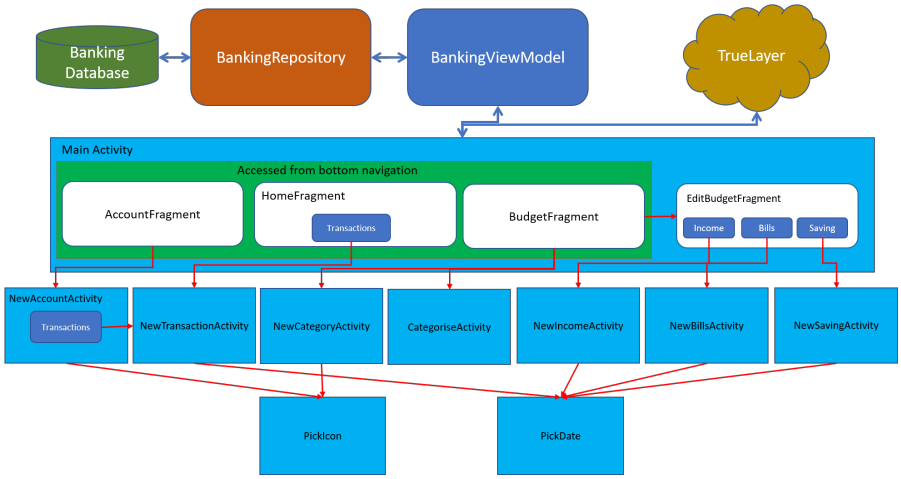


Figure 5.1: Visualisation of the Application - The structure of U.I components and how these interact with the ViewModel, Repository, Database and TrueLayer webservice.

5.1.1 Activities, Fragments and Navigation

The activity structure of the app was as follows; the MainActivity activity which hosts all of the 4 main screens, and 9 activities for the various input forms. The MainActivity contains a bottom navigation menu and a fragment container. This container is filled with one of 4 fragments that implement the 4 main screens of the app. This is done using the Jetpack Navigation component [48], the empty fragment container is called as a NavHostFragment object, which is then paired with a NavController to allow the fragment within the container to be switched between destinations asserted in the NavGraph. The bottom Navigation menu is

linked to the NavController so that it can be used to switch between the Home, Account and Budget screens. The EditBudget screen is displayed from a button click which invokes the NavController. Views can be observed for click events using the SetOnClickListener function, which allows an on-click callback to be specified.

```
val navView: BottomNavigationView = findViewById(R.id.nav_view)

val navController = findNavController(R.id.nav_host_fragment)
// Passing each menu ID as a set of Ids because each
// menu should be considered as top level destinations.
val appBarConfiguration = AppBarConfiguration(
    setOf(
        R.id.Transactions, R.id.Accounts, R.id.Budget
    )
)
setupActionBarWithNavController(navController, appBarConfiguration)
navView.setupWithNavController(navController)
```

Figure 5.2: Implenting the NavHostFragment and NavController

Navigation between activities is done using instances of the Intent class [49]. These are messaging objects which can be used to request actions from other app components and facilitate the communication of data between them. To launch an activity an Intent that describes it is passed to the startActivityForResult() function.

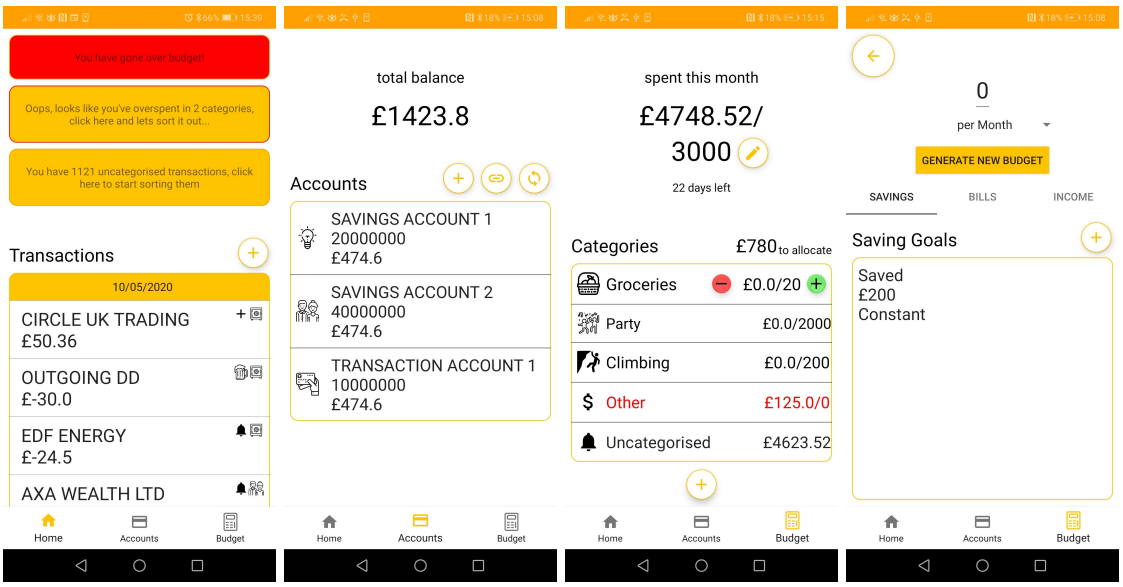
5.1.2 ViewModel and Repository

The application uses the BankingViewMode class to retrieve and manage all the data stored in the Database. A single instance of the class is created for the MainActivity, which means it can be accessed by all of the fragments that implement the main screens; making switching between each one smoother as no new data is needed. Instances of the class have a set of LiveData attributes that hold data retrieved from the database, these are observed in U.I. components to display up-to-date data. On initialisation, the LiveData attributes are retrieved from the repository. Calculations are made using the attributes and stored in MediatorLiveData objects [50]. These allow multiple sources to be combined into one object so they can be observed by a single observer.

A set of methods handle modifying data in the database. These take Entity objects as arguments and then request the appropriate action from the repository. Database requests can take a long time to run, freezing up the main thread and causing the U.I. to react slowly. This issue is mitigated by running the repository requests in a Coroutine; Kotlin's solution for asynchronous programming, a coroutine is a lightweight thread that can run concurrent operations with the main thread [51].

As my application does not store any data in an online database, and as I opted for letting the user manually sync their open bank data, the repository simply passes any database requests straight from the ViewModel to the DAO. This makes the repository almost redundant in my implementation, but it was left in to easily allow any expansion of the application.

5.2 Overview



(a) Home screen (b) Account screen (c) Budget screen (d) EditBudget screen

Figure 5.3: The Main Screens of the Application

5.2.1 Aesthetics and Logo

As seen in figure 5.3, the visual design of the app has changed a fair amount since the design stage. The colour scheme has been changed so that background of the app screens is now simply a block white colour and the items in the screen are highlighted with a subtle yellow or orange colour. This was done for two reasons; Firstly it was decided that the original colour scheme was too similar to those of Cleo and Emma and a different primary colour would help the app become more recognisable. Secondly, a lighter colour scheme made text stand out and makes it easier to direct the user's attention to important information.

In an another effort to make the app more recognisable a unique logo was created that sports the colour scheme of the app, this is displayed in the app's icon.

Figure 5.4: qwibBank logo

5.2.2 Implementing Lists

Most of the screens in the app involved displaying a list of data or entities from the database into a container. These were implemented using RecyclerView objects [52], scrollable containers populated from a dynamic set of data. The content in each RecyclerView is controlled by a custom adaptor, created by extending the RecyclerView.Adapter class.

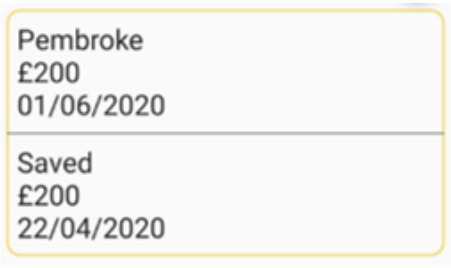


Figure 5.5: RecyclerView list for the SavingGoal entity

The data for each list is retrieved and kept updated by calling the appropriate function from the ViewModel and observing the LiveData object that the function returns. After retrieving the data it is passed to the appropriate adapter so that it can populate the RecyclerView. If the LiveData object is empty then the RecyclerView will be replaced with a message asking the user to enter items.

```
bankingViewModel.allBills.observe(requireActivity(), Observer { bills ->
    // Update the cached copy of the bills in the adapter.
    bills?.let { billAdapter.setBills(it) }
    if (bills.isNullOrEmpty()) {
        billRecyclerView.visibility = View.GONE
        emptyView.visibility = View.VISIBLE
        emptyView.text = "Please enter your Bills"
    } else {
        billRecyclerView.visibility = View.VISIBLE
        emptyView.visibility = View.GONE
    }
})
```

Figure 5.6: Retrieving the LiveData object and passing data to the RecyclerView adaptor

When one of the RecyclerView elements is clicked the application needs to navigate to the appropriate form to edit that data. This is done by attaching an OnClickListener to each view element as it is initialised.

5.2.3 Home Screen

Figure 5.3a shows the implementation of the home screen. The user's performance against their budget was omitted from the design as this was already being displayed on the Budget screen. Instead, the list of transactions was added so that the user can view all their incoming transactions and easily edit them or add more.

The notification feed is implemented by showing or hiding views based on a set of conditions. There are 6 possible notifications, some of which have an on click behaviour -

- The user will be warned if they are predicted to go over budget, based on the prediction algorithm.
- The user will be notified if they have gone over budget.

- A notification will be displayed if the user has not created a budget, clicking this will navigate to the EditBudget screen.
- A notification will be displayed if the user has not created or linked any accounts, clicking this will navigate to the Account screen.
- A notification will be displayed if the user has not created a budget, clicking this will navigate to the EditBudget screen.
- A notification will be displayed if the user has exceeded their allocated allowance in a category, clicking this notification will send the user to the Budget screen.
- If there are uncategorised transactions the user will be notified, clicking the notification will send the user to the Categorise screen.

The RecyclerView adaptor for the transactions list is a bit more advanced as it groups its elements based on date, with a dividing view containing the data between each group. This makes it easier for the user to see how much they've spent on any given day. The Transaction's Category and Account logos are displayed alongside the text information.

Logos are stored as URL path strings that either point to a vector image in the drawable folder, or in the case of linked accounts; a .svg file downloaded from the TrueLayer server. The SVGUtils class was written to help manipulate the vector logo images. The fetchImage method handles loading images into views. Vector images in the drawable file are easily loaded using Android's default methods for retrieving drawables. The .svg files are harder to process; an InputStream is opened to retrieve the data from the file, this is then loaded into the view using Sharp [53]. A library that facilitates loading vector graphics as drawables, which Android does not support by default.

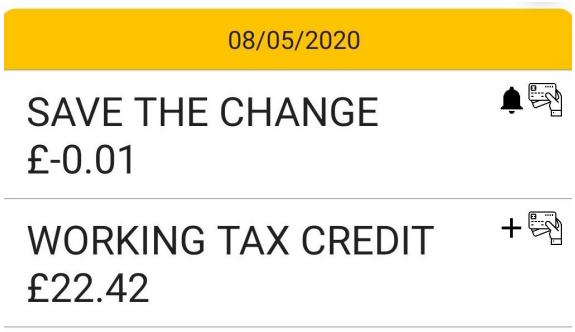


Figure 5.7: Transaction item group

5.2.4 Account Screen

The account screen, figure 5.3b, is mostly unchanged from the design, bar some minor changes. The add and link account buttons have been moved to the top of the container and use icons instead of labels inside the buttons. This allowed them to be neatly packed together. A "sync" button has been added which when pressed triggers the DataApi to retrieve any new transactions from open banking linked accounts.

The account recyclerview displays the name and balance of each element, as well as the account's logo.

5.2.5 Budget Screen

The budget screen received a few minor tweaks, the total amount spent was placed up against the budget so that user's could have a clearer picture of their spending. Similarly the amount spent in each category this period was also added up against the amount allocated in the list view.

The elements in the category list consist of a logo, the name of the category, the amount spent and amount allocated and buttons to allocate or remove funds from the category. To implement the click behaviour separate listeners are attached to different elements of the view so that the appropriate behaviour is called when an element is clicked. If the amount spent in a category for the current period exceeds the amount allocated then the colour of the text in the view is changed to red. The click behaviour on the minus button is also disabled and the button is greyed out slightly so that the user is forced to modify their allocated amount or increase their budget to deal with the overspend.

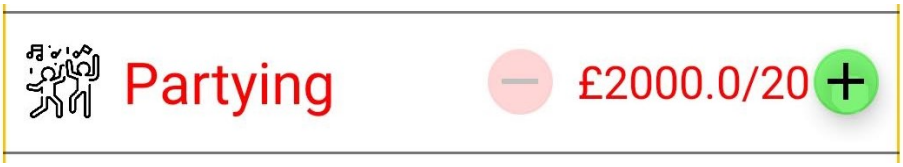


Figure 5.8: A category list item, with overspend behaviour

5.2.6 EditBudget Screen

The edit budget screen has undergone one major change from the design stage, the 3 lists have been split up into separate tabs. This was done in an attempt to make the screen less cluttered, while also allowing more elements of each list to be seen without the user having to scroll.

```
//Initialise the tablayout and viewpager
tabLayout = view.findViewById(R.id.tab_layout)
collectionAdapter =
    collectionAdapter( fragment: this)
viewPager = view.findViewById(R.id.pager)
viewPager.offscreenPageLimit = 3
viewPager.adapter = collectionAdapter
TabLayoutMediator(tabLayout, viewPager) { tab, position ->
    when (position) {
        0 -> tab.text = "Savings"
        1 -> tab.text = "Bills"
        2 -> tab.text = "Income"
        else -> tab.text = "Undefined"
    }
}.attach()
```

Figure 5.9: Initialising the viewpager

A ViewPager [54] was used to host the separate tabs and allow the user to switch between them by swiping horizontally, this is paired with a TabLayout [55] to display which tab is currently active and allow them to be switched by clicking the labels. Each tab was implemented as an independent fragment, each with their own logic to handle interactions with their elements. To manage the loading and switching of fragments the FragmentStateAdaptor class was extended so that it could be used

with several unique fragments and this adaptor was then attached to the ViewPager on initialisation.

The generated budget is calculated by first combining all the user's Income, Bills and Saving Goals into a single array. The budget period is then set as the latest date out of all these objects. Next, the array is iterated through and added or subtracted from the current total account balance, producing the total amount that can be spent that period without falling below the minimum accepted balance. This is divided by the amount of weeks or months in the period to produce an allowance. The minimum accepted balance is modified in the 'Saved' SavingGoal entity, allowing money to be set aside or negative balances to be considered. The dropdown menu above the button is used to select if the budget is per week or per month.

5.2.7 Input Forms

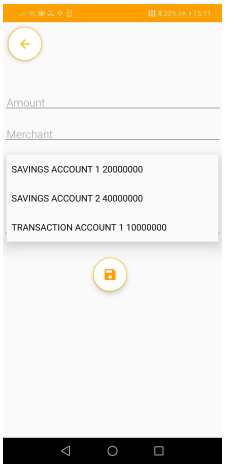


Figure 5.10: Select account spinner

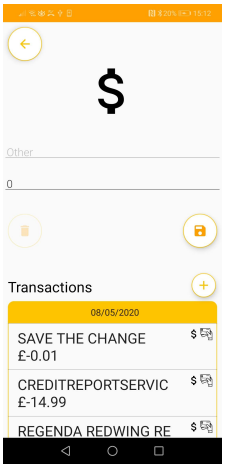


Figure 5.11: NewCategoryActivity with transactions list visible

Figure 5.12: PickIcon activity

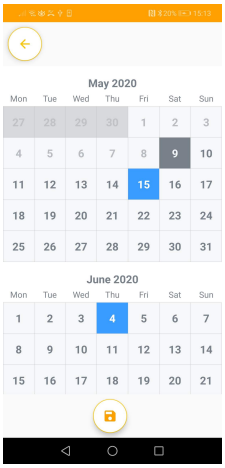


Figure 5.13: PickDate activity

To implement the input forms an activity was implemented for each entity. Selecting an object from a list to edit will pass it to an activity in an Intent. Input activities select the appropriate layout and pre-fill inputs based on the contents of their Intent.

When opened by selecting an entity, the NewCategoryActivity and NewAccountActivity will show the list of transactions linked to the selected entity. A major new addition to these screens is the image above the input boxes that displays the logo of the entity, clicking this will launch a PickIcon activity where the user can select a new image to replace the logo.

The PickIcon activity was implemented by using a recyclerview, whose dataset is simply a list of image paths. The EditCategory and EditAccount screens are implemented by the NewCategoryActivty and NewAccountActivity respectively, if these activities are not passed an entity object in the Intent then only the input boxes and logo will still be displayed.

In some of the forms a date input is needed. Clicking or focusing the input box will launch a PickDate activity that lets the user select dates from a calendar. As the Income and Bill objects require multiple dates to be selected the default Android calendar widget was unsuitable. To overcome this issue the TimeSquare library's [56] CalendarPickerView was used, this gives the option to allow multiple

date selection which is enabled when the activity is launched from the Income or Bill forms.

So that the user can only select existing Accounts or Categories when creating or editing transactions, dropdown menus were required. These were implemented using a spinner object [57] paired with an ArrayAdaptor. To populate the spinners, all the accounts/categories are retrieved from the database and passed to the adaptor which generates the views in the spinner.

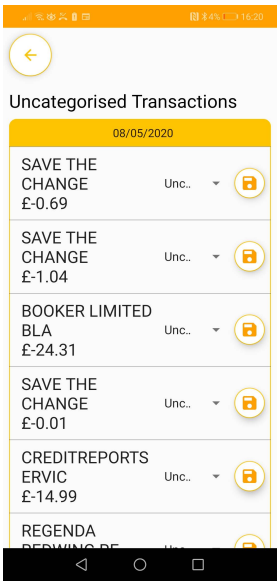


Figure 5.14: CategoriseActivity

To aid in assigning categories to imported transactions a new input screen was added, the Categorise screen. This screen shows a list containing all uncategorised transactions, each with a dropdown menu that lets the user easily select a new category for each item. This screen is accessed from a notification on the HomePage or by selecting 'Uncategorised' from the category list on the Budget page.

5.2.8 Log-in Screen

It was decided to let the operating system handle the log-in process for easier implementation and higher security at the cost of losing the ability to allow multiple user accounts. On launching the MainActivity of the application the user is sent via an Intent to the ConfirmDeviceCredential screen that is present on every device. This screen allows the use of fingerprint ID or a passcode to enable access to the application. Cancelling the log-in or failing to enter the correct information too many times will close the app.

5.3 Database

Data storage for the application is done with a local SQL database that is implemented and managed using the Room library. The structure of the database is identical with two exceptions, as I opted to forgo having separate user accounts on one device, the User entity is no longer a parent to all the other entities. User is now simply a container for storing the current allowance and time period chosen by the user. I also added an AccessToken entity to store TrueLayer api tokens. Entities are simply defined as normal Kotlin classes and then annotated with @Entity. All

```
@Parcelize
@Entity(tableName = "user_table")
class User(@PrimaryKey(autoGenerate = true) val id: Int?,
           val allowance: Int,
           val period: String
) : Parcelable
```

Figure 5.15: Implementation of the User entity

of the app's entity classes are also annotated with @Parcelize, this allows them to be bundled up and sent between activities using Intents.

The database's DAO is implemented as a interface and annotated with @Dao. It contains a set of methods that allow entities to be fetched, deleted or modified. Room allows simple update, insert and delete operations with the @Update, @Insert and @Delete annotations, while retrieving data and more complex operations are done with @Query that takes an SQL string query as an argument.

```
@Dao
interface BankingDao {

//Transaction queries
    @Query( value: "SELECT * from transaction_table ORDER BY id ASC")
    fun orderedTransactions(): LiveData<List<Transaction>>

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insertTransaction(transaction: Transaction)

    @Delete
    suspend fun deleteTransaction(transaction: Transaction)

    @Update(onConflict = OnConflictStrategy.IGNORE)
    suspend fun updateTransaction(transaction: Transaction)
```

Figure 5.16: Implementation of the DAO

As Room can only be used to store primitive types, a set of type converters were implemented in the Converters class. This enabled Dates and Lists of Dates to be stored in the database. These were annotated with the @TypeConverter annotation.

In the BankingRoomDatabase class, an extension of RoomDatabase, all these elements are combined to form the Database. It has a @Database annotation that lists all the entities, a @TypeConverters annotation to call the Converters class and contains an abstract method that returns the @Dao annotated class.

5.4 TrueLayer Integration

Accounts are linked to the app through open banking, using the TrueLayer API as a mediator between the app and user bank accounts. Retrieving open banking data using TrueLayer is done through a three step process;

- First a user must be authenticated, which will generate an Exchange Code.
- Next an Access Token must be obtained using the Exchange Code.
- Finally the token can be used to make requests to the api to retrieve data.

When a user clicks the button to link an account, the authentication process is started. A link generated from the TrueLayer website is parsed and attached to an Intent, when launched this opens the browser on the TrueLayer webpage for entering open banking credentials. Completing this form will redirect to an url that is Deep Linked back to the app [58]; Android will direct the user back to the application and retrieve the Exchange Code embedded in the url.

A request is then made to the server to exchange the Exchange Code for an Access Token. API requests were implemented using the Retrofit library [59]. Retrofit enabled the creation of interfaces that build the api requests and convert the JSON response into predefined objects. Two interfaces were built to handle the requests; a DataApi interface and AuthApi interface, to handle retrieving Data and exchanging tokens respectively. All API calls are made in a Coroutine so that the main thread remains unblocked. The request to exchange tokens returns an AccessToken object on success, this contains both the token for retrieving data and a refresh token used for retrieving a new AccessToken once the original one has expired. AccessTokens are stored in the database for use later when syncing new data from a linked account. Finally, the token is used to request the account and transaction data. A

```
interface DataApi {
    @GET( value: "/data/v1/accounts")
    suspend fun getAccounts(
        @Header( value: "Authorization" ) accessToken: String) : Response<AccountResults>

    @GET( value: "/data/v1/accounts/{account_id}/transactions")
    suspend fun getTransactions(
        @Header( value: "Authorization" ) accessToken: String,
        @Path( value: "account_id" ) accountId: String) : Response<TransactionResults>
```

Figure 5.17: The DataApi interface

request is made to list the user's accounts and then another request is then made to retrieve the transaction data for each account. The returned AccountResults and TransactionResults objects are processed so that a set of Account entities and child Transaction entities are created and inserted into the database through the BankingViewModel. Account logo files are downloaded by calling the saveSVG method of SVGUtils on logo uri strings.

To sync data their linked account data the user clicks the 'sync' button on the accounts page, this will trigger requests on every stored AccessToken. Instead of an exchange token, the refresh token is used to retrieve a new Access Token and this is then used to retrieve the account data - adding any new transactions since the last update.

5.5 Security

To ensure the security considerations outlined in section 2.2.4 the following measures have been put in place;

To ensure that any uploaded information isn't accessed or modified by unauthorised parties, the user is required to identify and authenticate themselves on opening

the app. This is done by requesting that they provide either their device's fingerprint or passcode. This process is handled by the device's operating system, so the information is stored securely and cannot be accessed or exploited through the app. The log-in process is also protected against brute force attack as it only allows a maximum number of attempts before locking the app for progressively longer periods of time.

The HTTPs protocol is used for any communication with TrueLayer, preventing any account information from being stolen or modified during communication with their server. The TrueLayer authentication and access token process also provides another layer of security that an attacker would need to breach to access any data. Even if an attacker were to overcome all this they would not be able to get hold of usable account credentials as these are never directly provided to the app and are also enciphered using AES-256 with a uniquely generated key as described in [60].

Information stored locally in the app's database needed to be encrypted in case the contents of it was leaked. This was done by encrypting the whole database using the SQLCipher library. SQLCipher allows the use of the standard SQLite API to create and manipulate a database while it handles security in the background. Making sure that data is encrypted and decrypted as it is written to and read from the internal storage. By default SQL-Cipher enciphers the database with 256-bit AES encryption, which is sufficiently high level for the financial data the app handles.

5.6 Spending Prediction

The prediction feature was implemented by training a multi-layer feed-forward neural network. When given the current day of the week and the total amount spent this week, the network attempts to predict the final spend by the end of the week. Which can then be used to check if the user is predicted to spend more than their budget by the end of the period.

To train the network a supervised learning algorithm was written in Python using the Tensor-flow platform; which provides a comprehensive set of tools, libraries and community resources that lets developers easily build and deploy ML powered applications [61]. As shown in figure 5.18, the neural network is constructed using keras; Tensorflow's implementation of the Keras API specification, a high-level API to build and train models. The network consists of 4 layers;

- An input layer of 2 neurons, which takes the current day (As a number) and current total spend as inputs.
- 2 hidden layers of 64 neurons, each with the Rectified Linear Unit(ReLU), $Relu(x) = max(0, x)$, activation function applied to their output.
- An output layer of 1 neuron which outputs the predicted final spend for the current week.

A year's worth of TrueLayer's mock transaction data was processed to create a dataset that could be used to train the network. Each row in the dataset consists of a number representing the current day, the running total for spending that week, and the final spend at the end of the week. The network is trained by feeding it the data in batches of 32 rows at a time, the day and running total are passed through the network to produce a predicted total spend. Once all the rows in the batch have been processed, an error signal is calculated using the Mean Squared Error(MSE) function, $MSE = \frac{1}{n}\sum_{i}^{n}(y_i - \widehat{y}_i)^2$; the average squared difference between the predicted total spend and the actual final spend for each row of data. This value is then

```
#Build neural network
def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=[len(dftrain.keys())]),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model

model = build_model()
```

Figure 5.18: Building the neural network

back-propagated through the network to generate a weight change for the connections between each neuron. One pass through the whole dataset is called an epoch, after an epoch the algorithm will check stopping conditions and decide if training should be stopped. A portion of the training data is held back as a 'validation set', the error on this set is checked and another epoch of training will be initialised as long as the error has not increased from the previous epoch. This ensures that the network does not become over-fitted to the training data. The training process was also implemented with keras, using the model.fit function.

Once the model was trained the next problem was how to use it to make predictions in the app. The chosen solution was to use TensorFlow Lite to deploy the model locally and allow on device inference. The model was saved as a compressed flat buffer .tflite file using the TensorFlow Lite Converter, it was then placed in the assets folder of the application to allow access from within the app. Methods for prediction are called from in the BankingViewModel class; The model is retrieved as a MappedByteBuffer which can then be loaded into the TensorFlow Lite Interpreter. The Interpreter can then be fed data which it will use to execute the model and provide predictions. A prediction is made using the current day and current period spend, using a MediatorLiveData object ensures that a new prediction is made if user or periodTransaction values are changed. The current predicted value is observed on the Home Screen, if this value is greater than the current allowance then the user will be notified that they are predicted to go over budget.

5.7 Testing

5.7.1 Unit Testing

Due to the scale of the application, and the time-frame of the project, it was not realistic to write unit tests to cover every class in the program. Instead it was decided to focus testing efforts on a few core components; the BankingViewModel and BankingDao classes.

Android unit testing is done via two approaches; Local tests are compiled and run locally to minimise execution time, as they are not run with an Android device libraries such as Robolectric and Mockk are used to simulate Android framework objects. The other approach is Instrument testing, where unit tests are run on

an Android device or emulator, these tests are much slower but allow more robust interaction with complex Android dependencies and components. Both types of tests have been written to test various aspects of the application.

5.7.2 Integration and System Testing

Integration and systems testing has been performed manually by running the application on a device and testing the behaviour across a range conditions.

During development features were regularly tested as new functionality was implemented, to ensure that everything was interacting as intended. Android's logging system LogCat was regularly used to track the values of various variables and states to check that they were correct.

System's testing was done by manually running through a set of tasks based on the User Stories presented in the function requirements table 3.2. Volunteers were also asked to install the app as part of the evaluation process, discussed in the next chapter, and complete a set of functional tasks. This provided a larger volume of testing so that it was more likely that problems would be picked up.

5.7.3 Compatibility Testing

To ensure that the application functioned correctly across the range of compatible devices it was tested on a selection of both physical and virtual devices. As the volunteers for the evaluation stage also installed the application on their own device this provided an even wider coverage of devices than included here.

Virtual devices were created with the Android Emulator, which is included in Android Studio. It allowed the app to be tested on a large range of devices, shown in table 5.1, including both phone's and tablets, with a range of different operating system versions.

Device Model	**Android Version**	**Type**
Huwawei P20 Lite	9.0	Physical Phone
Samsung Galaxy A40	9.0	Physical Phone
Pixel 3 XL API R	10.0	Emulated Phone
Nexus 7	7.0	Emulated Tablet
7" WSVGA	8.1	Emulated Tablet
Galaxy Nexus	7.1.1	Emulated Phone
Pixel 2	8.1	Emulated Phone

Table 5.1: Devices used during compatibility testing

The table also lists the physical devices that the app was tested on. To distribute the application on to physical devices it was converted into an Android Package(APK) file. When this file is opened on a compatible device it prompts the user to open and install the app using Android Package Manager. As the emulator cannot behave exactly like a real device, testing the app on physical devices ensured that no major compatibility issues slipped through testing.

5.7.4 Usability Testing

As mentioned in the COVID-19 impact section this could not be completed as intended, with users being directly observed using the application. Questions were however added to the evaluation questionnaire that asked user's to complete tasks and rate how easily they could complete them. This allowed some degree of useability testing of the app.

5.7.5 Performance Testing

The Android Profiler, a tool included in Android Studio, was used to measure how well the application used the devices resources. The profiler was used to record the CPU, Memory and Energy usage while a set of tasks were completed, that covered most of the apps functionality. During these tests the app was run on a physical Huawei P20 Lite, running Android OS version 9.0. This middle of the range and slightly dated smartphone represents a device that would be accessible to most of the app's target audience due to its reduced price tag, and so the app could be considered to be performing well if this phone could easily handle running it.

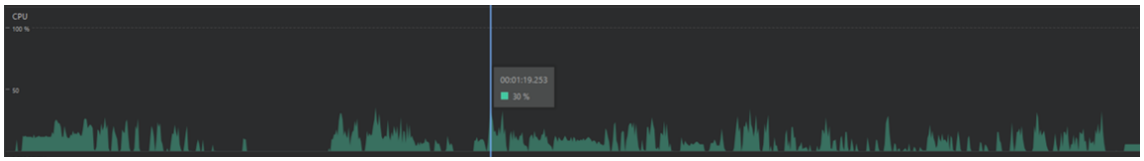


Figure 5.19: Application CPU usage while completing functional tasks

As shown in figure 5.19 the CPU was not running at capacity at any point, with usage peaking at 30% during the testing. This was an acceptable level although there is most likely some scope for improvement as minimising CPU usage was not a focus during development.

Figure 5.20: Application memory usage while completing functional tasks

Memory usage peaked at 158MB during testing, which is just less than 4% of the phone's 4GB of RAM and well within normal parameters. While most of the memory increase shown in the trace is a result of adding and loading more transactions and accounts over the course of the test, there was a gradual increase over time while the application was open. This is slightly worrying as it suggests the presence of some memory leak issues, where unused objects are not correctly released from the memory. Minimising the application seemed to clear these objects, as shown in the sharp dip on the graph, so it is not a major concern as the user would have to be using the app continuously for a large length of time for it to become an issue. It is however something to explore in any future testing and development.

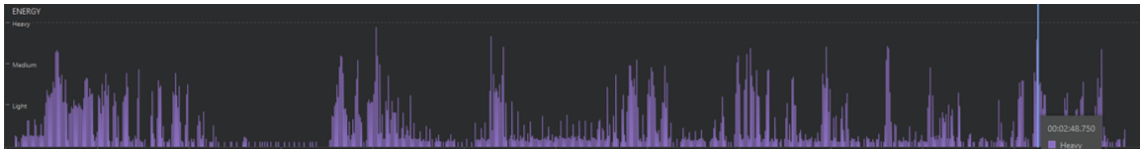


Figure 5.21: Application energy usage while completing functional tasks

5.21 shows the energy usage during the test, on the whole the energy usage was fairly high throughout, with the app mostly causing a medium impact to the battery with some short periods of heavy use. This was seen as acceptable due to the fact that the application consumed barely any of the battery while the app was running in the background and that under normal use case it would not be open for extended

periods. Again, there are likely improvements to be made in this area as it was not a focus during development.

Chapter 6

Evaluation

6.1 User Testing and Evaluation

To evaluate the success of the project, sixteen volunteers were recruited to test and provide feedback on the application. They were asked to download the application on their personal devices and complete a set of tasks, as listed in table 6.1. Following this they were then asked to complete a set of questions on their experience and given the chance to provide feedback or report any problems with the app. A full copy of the questionnaire can be found in Appendix D.

This method of evaluation was approved by the University of Sheffield ethics procedure and volunteers completed a consent form before participating. The approval document, information sheet and consent form are the same as in 2.6.

6.1.1 Useability

The volunteers were not given any instruction on how to complete each tasks, so they had to navigate the application independently. They were asked to rate how easily they found completing each task on a scale from 1 to 5 (Unable to complete the task to easily able to complete the task). This allowed the useability of the application to be tested and provided feedback on how intuitive the application is.

	Task	**Average Useability Score**
1	Manually add both a new account and a new transaction.	4.4/5
2	Add a linked account from the Account page.	4/5
3	Delete or edit one of the generated accounts.	4.2/5
4	Generate a budget.	3.7/5

Table 6.1: Evaluation tasks with their usability score

On the whole the application scored well, with most participants indicating that they could complete the majority of tasks easily. The Generate budget task seemed to be the most difficult with 30% of participants scoring it a 3 or lower, in the feedback section for the task it was suggested that this was down to the page for generating a budget not being completely obvious to find.

Only one person failed to complete a task; they were unable to add a linked account. In their feedback however it was made clear that this was not a useability issue, they were able to complete the process as intended but the accounts did not load.

6.1.2 Evaluation Score

Following the completion of every task, participants were asked set of questions that asked them to score various aspects of the application, again on a scale from 1-5. The results are shown in table 6.2. Most of the testers agreed that the application

	Question	Average Score
1	How responsive did you find the application?	4.4/5
2	How do you rate the general look and feel of the application?	4.2/5
3	How would you rate the overall quality of the application?	4.3/5
4	Would you be likely to use the application in its current form?	4/5

Table 6.2: Evaluation scores of various aspects of the application

ran well and was responsive to use; with every single participant giving the app a score of 4 or 5. Similarly every vote on the look and feel of the application was at 3 or above, with a couple of participants even praising the interface applications visual design.

Overall the application was very well received, the quality of the application was scored at 4.3/5 and a large amount of participants indicated they would be likely to use the app if it was released, with 37.5% of them providing a score of 5 for that question. Some participants who indicated they wouldn't be likely to use the app stated that this was due to them already budgeting in other ways.

Participants were also given an explanation on how the generated budget was calculated and asked if they would find it helpful. 87.5% indicated that they would find the feature useful, with a few others stating that they were unsure if it would be useful in their situation and that a more dynamic budget would be ideal.

6.1.3 Other Feedback and Issues

Participants were given the opportunity to provide feedback on each individual feature they tested, provide general feedback on any other aspect of the application and also report any crashes or issues they encountered. The insights gained from this feedback proved helpful in highlighting changes that could be made in future development and allowed for a more well-rounded evaluation.

As mentioned in the useability section, a couple of participants pointed out that the Edit Budget screen was difficult to find initially. This has since been remedied slightly by adding a notification that directs the user to the screen if they have not created a budget. Further evaluation is needed to discern if this has fixed the issue and potentially larger changes to the structure of the app may be needed.

Feedback was given on aspects of the input forms. Suggestions were made to make some selections more obvious to the user by adding labels next to input values and making it more obvious that the logos could be selected to change them. Similarly one participant stated that it was not clear enough that list elements could be selected to edit them.

Another participant suggested that the safety and security of linking an account through open banking needed to be made clear, so that users could easily know what they were signing up too. This could be solved by adding a page that explains the security details of linking an account through TruLayer, that the user would be shown prior to being directed to the TrueLayer log in form.

Some other feedback highlighted that some tasks, such as adding income and bills and categorising a long list of imported transactions, would take a long time to complete. Solutions for both of these issues, auto-categorising transactions and automatically detecting income and bills, were considered at points during the development process but due to time constraints were not added. Another feature that did not make the current version of the application and came up in feedback was displaying a history of their past spending and performance compared to their budget.

Four people experienced technical issues while using the application. A couple of them experienced problems with linking accounts; one with loading the TrueLayer webpage correctly, this was not able to be reproduced and is assumed to be a problem with their device or TrueLayer's sandbox server; the other noted that the linked accounts failed to load after completing the sign in, this has been reproduced and it is due to TrueLayer responding with an Internal Server error or a Bad Request error in rare cases. Further testing is required to discover the root of this issue. The third issue was that images were sometimes slow to load, changes to make the image loading process more efficient or run on separate threads could solve this issue. Finally one user had the app crash on start-up, more thorough testing is required to discover the cause of this issue.

6.2 Evaluating the Prediction Model

The effectiveness of the prediction model was evaluated using the mock transaction dataset mentioned in section 5.6. The model was trained and tested 10 times using this data, each time the dataset was split into 2 separate sets; a random 75% sample of the data was used to train the model while the remaining 25% was used to evaluate the performance of the model after training.

Once trained the model was used to make predictions on the evaluation dataset, the current day and total amount values were fed to the model to produce a predicted spend. Predicted values were compared against the expected values contained in the dataset to give an average error. Over the course of the testing the average error across all predictions was 36%, which on its own suggests that the model is performing decently.

However, as shown in 6.1 and 6.2, the spread of errors is quite large. With some predictions being as much as 90% wrong. Due to this the predictions cannot really be relied on in their current state. It is hypothesised that these extreme errors are due to predictions on the first day of the week being incredible difficult, more testing is required to explore this.

The reliability of these results for evaluating the model is also under question, as the TrueLayer dataset used does not reflect that accurately the spending habits of a real person. There is no apparent pattern to the transactions or weekly spending amounts of the mock account and so it's inherently hard for any model to reliably make predictions on them. There are no account statement or transaction datasets available in the public domain, and as collecting real financial data was outside of the scope of this project, this was the only option available. Ideally the model could be trained and tested on real data from several different people.

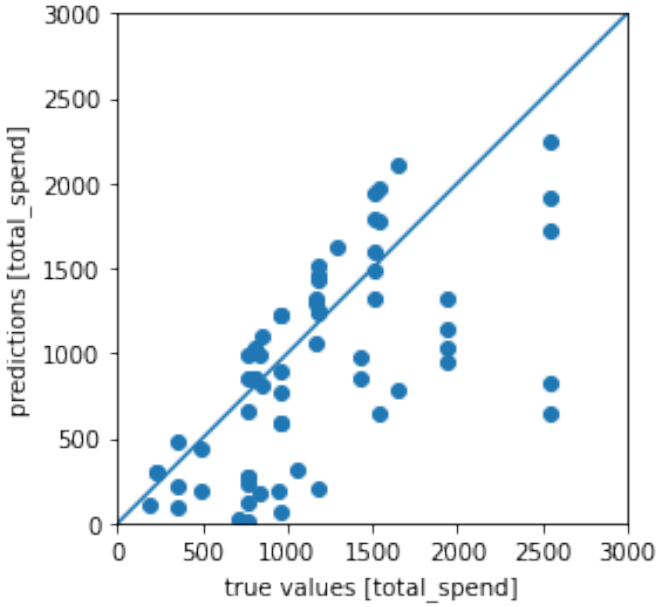


Figure 6.1: Comparison of produced values and expected values when predicting total spend using the model.

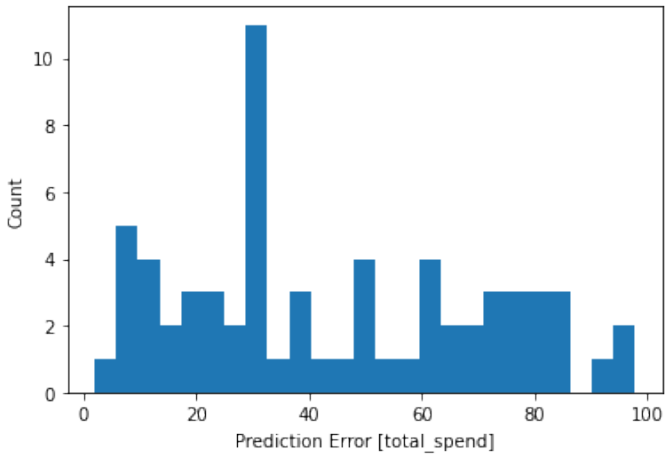


Figure 6.2: Absolute prediction error when predicting total spend using the model.

6.3 Requirements Analysis

The requirements outlined in section 3.3 were reviewed and analysed to determine whether they have been achieved. Tables 6.3 and 6.4 show the results of this analysis.

	User Story	**Priority**	**Status**
1	As a user I want to be able to manually add financial data so that I can keep track of my money.	M	Achieved
2	As a user I want to be able to link my bank accounts with the app so that I can track them automatically.	M	Achieved
3	As a user I want to be able to view my spending in categories so that I can understand where my money is going.	S	Partially Achieved
4	As a user I want to be able to view my account information and balances so that I know how much money I currently have.	M	Achieved
5	As a user I want to be able to see my bills and subscriptions so that I can manage them.	C	Not Achieved
6	As a user I want to be able to set saving goals so that I can budget around them.	S	Achieved
7	As a user I want to be able to input upcoming payments and bills so that I can budget around them.	S	Achieved
8	As a user I want to be able to input future income so that I can budget around it.	S	Achieved
9	As a user I want to be able to create and edit my budget so that I can save money.	M	Achieved
10	As a user I want to be able to generate a budget based on my goals so that I can save money.	S	Achieved
11	As a user I want to be able to distribute my budget into categories, so that I can prioritise my spending.	M	Achieved
12	As a user I want to be able to see my performance against my budget so I can adjust my spending.	M	Achieved
13	As I user I want to be able to see my past spending and budget performance so that I can understand how my spending is changing.	C	Not Achieved
14	As a user I want to be notified if I'm predicted to go over budget so I can adjust my spending.	S	Partially Achieved
15	As a user I want to be notified if I've gone over budget so that I can reallocate funds between categories and mitigate the damage.	S	Partially Achieved
16	As a user I want to be given advice on how to save money so that I can make better saving decisions.	C	Not Achieved

Table 6.3: Review of functional requirements

All of the highest priority functionality of the application have been completed as have many other features. Due to time constraints none of the extension 'Could Have(C)' features were implemented. Story 3 was only partially achieved as the functionality to automatically assign transactions to categories was not implemented. Push notifications were not implemented so story 15 is only partially complete.

The prediction algorithm can only really be said to be partially implemented in its current form as the model is not very accurate. The model is also a general one as it is trained using the TrueLayer dataset, it is not tailored to the spending habits of the user. Training the model on device is not possible, or at least very difficult, so to implement user tailored predictions a webservice would have to be created that receives transaction data from the app and returns trained .tflite models. Creating this service would have been too time consuming, so the feature was left partially implemented.

User Story	Priority	Status
As a user I want to be able to run the product on all android devices so that I can use the application on my device.	M	Achieved
As a user I want my data to be stored securely so that it is protected.	M	Achieved
As a developer I want my app to be responsive and fit to the device size so that the user experience is enjoyable.	S	Achieved
As a developer I want my app to be thoroughly tested and bug free so that the user experience is enjoyable.	S	Partially Achieved
As a developer I want my app to run quickly on the intended hardware so that the users stay engaged.	S	Achieved
As a user I want to be able to easily navigate and understand the application so that I can use it without prior information.	S	Achieved

Table 6.4: Review of non-functional requirements

All of the non-functional requirements are believed to have been achieved with the exception of ensuring the application is completely bug free, while extensive testing has been completed some issues slipped through and therefore more thorough testing is required.

6.4 Further Work

As touched upon in the previous sections there are a couple steps that still to be completed before the app could be considered release ready. The app should be more extensively tested to eliminate any remaining issues, and also tested from a security perspective to check for any vulnerabilities.

The prediction feature would also need to be fully implemented so that it is tailored to the user. A considerations with this is how predictions would be made for new users with very little data, potentially a generic student profile could be trained off a collection of student financial data and this could be used to initially make predictions until enough data is collected.

Further development could be done to complete the remaining features from the requirements stage and solve the issues highlighted in 6.1.

The prediction feature could be explored in further depth by collecting real user data to evaluate its performance. Different optimisations to the model or even completely different approaches to the problem could also be experimented with if a reliable data set was used.

One possible improvement to the model would be to factor in the amount spent in each category for each period, this might improve prediction as generally people spend a fairly consistent amount on some things; if a user has not already spent their usual amount on groceries yet they are likely to spend more before the end of the period.

Calculating a value for the confidence in each prediction could also go some distance towards solving the issue of earlier days being harder to predict. This value could be used to determine if the user is notified or not; if they are predicted to go over budget but the prediction has low confidence then they should not be notified.

Chapter 7

Conclusions

The project started with off with one question - How do we help students manage their finances? Mobile applications were chosen as a potential solution to the problem, and so their use and development was researched. Existing personal finance apps were analysed and the prospect of using A.I. to aid in personal finance management was explored. A pre-requirements survey was also conducted to help get a clearer picture of how student's currently handle their finances and what features they would find helpful in a personal finance application.

Insights and information gained during the research stage was used to develop a set of requirements for the application, which informed choices made during the design of the solution and aided the final evaluation of the success of the project.

A native Android application was developed that fulfilled all of the core functionality required. The implemented app allows user's to easily manage their personal finances by allowing them to track their bank accounts through open banking or manual input, allows them to generate a budget and hosts a set of tools to help them stick to it; including a neural network based prediction algorithm that attempts to predict their future spending. The implementation and technical details of these features were discussed in detail as were testing methods that ensured the app was compatible with a range of devices, performed well on target hardware and aimed to eliminate crashes and bugs.

The application was then evaluated, firstly by performing user testing and gathering feedback from participants. The application was well received with the quality, use-ability and design scoring highly and many participants indicating that they would use the application should it be released. Next the prediction model was evaluated, this didn't perform exceptionally well but results of the evaluation were deemed inconclusive as access to a reliable testing dataset could not be achieved. Finally, the requirements were reviewed with all essential requirements being deemed complete and therefore the objective of the project achieved.

The evaluation highlighted some areas of further work, namely; developing extra features for the application, more extensive testing, improvements to the prediction algorithm and further exploration of A.I. techniques for aiding in personal finance.

To conclude, the goals of the project have been met and a personal finance application for Android has been developed. The implementation of the app is seen to be a success, however there is still room for further work on the solution.

Bibliography

[1] Pru Marriott. "An Analysis of First Experience Students' Financial Awareness and Attitude to Debt in a Post-1992 UK University". In: *Higher Education Quarterly* 61.4 (2007), pp. 498–519.

[2] Marzieh Kalantarie Taft et al. "The relation between financial literacy, financial wellbeing and financial concerns". In: *International Journal of Business and Management* 8.11 (2013), p. 63.

[3] Ruth Bushi. *Student Money Survey 2019 – Results.* URL: https://www.savethestudent.org/money/student-money-survey-2019.html. (last accessed: 22.04.2020).

[4] Unite. *UNITE STUDENTS INSIGHT REPORT.* 2016. URL: https://www.unitestudents.com/about-us/insightreport/2016-full-report. (last accessed: 22.04.2020).

[5] healthassured.org. *Why Is Financial Wellbeing So Important?* 2019. URL: https://www.healthassured.org/blog/why-is-financial-wellbeing-so-important/. (last accessed: 22.04.2020).

[6] moneyadviceservice.org.uk. *Beginner's guide to managing your money.* 2019. URL: https://www.moneyadviceservice.org.uk/en/articles/beginners-guide-to-managing-your-money. (last accessed: 22.04.2020).

[7] Owen Andrew. *The History and Evolution of the Smartphone: 1992-2018.* 2018. URL: https://www.textrequest.com/blog/history-evolution-smartphone/. (last accessed: 24.04.2020).

[8] S O'Dea. *Smartphone ownership in the United Kingdom (UK) 2012-2019, by age.* 2019. URL: https://www.statista.com/statistics/271851/smartphone-owners-in-the-united-kingdom-uk-by-age/. (last accessed: 24.04.2020).

[9] techopedia.com. *Mobile Application (Mobile App).* 2019. URL: https://www.techopedia.com/definition/2953/mobile-application-mobile-app. (last accessed: 24.04.2020).

[10] JetRuby. *Brief History of Mobile Apps.* 2017. URL: https://expertise.jetruby.com/brief-history-of-mobile-apps-286fbbf766a9. (last accessed: 24.04.2020).

[11] J Clement. *Total global mobile app revenues 2014-2023.* 2019. URL: https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/. (last accessed: 24.04.2020).

[12] J Clement. *Number of apps available in leading app stores 2019.* 2019. URL: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. (last accessed: 24.04.2020).

[13] Sensor Tower. *Top apps worldwide for Q2 2019 by Downloads.* 2019. URL: https://sensortower.com/blog/top-apps-worldwide-q2-2019-downloads-data-digest. (last accessed: 24.04.2020).

[14] gs.statcounter.com. *Mobile Operating System Market Share United Kingdom.* 2019. URL: https://gs.statcounter.com/os-market-share/mobile/united-kingdom. (last accessed: 24.04.2020).

[15] Apple. *Start Developing iOS Apps.* 2019. URL: https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift/. (last accessed: 24.04.2020).

[16] The Manifest. *Android vs iOS: Which Platform to Build Your App for First?.* 2018. URL: https://medium.com/@the_manifest/android-vs-ios-which-platform-to-build-your-app-for-first-22ea8996abe1. (last accessed: 24.04.2020).

[17] J Callaham. *The history of Android OS: its name, origin and more.* 2019. URL: https://www.androidauthority.com/history-android-os-name-789433/. (last accessed: 24.04.2020).

[18] Markoff. *Publishing private apps just got easier.* 2019. URL: https://medium.com/androiddevelopers/publishing-private-apps-just-got-easier-40399c424b8a. (last accessed: 24.04.2020).

[19] A Dossey. *A Guide to Mobile App Development: Web vs. Native vs. Hybrid.* 2019. URL: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/. (last accessed: 24.04.2020).

[20] Mounaim Latif, Younes Lakhrissi, Najia Es-Sbai, et al. "Cross platform approach for mobile application development: A survey". In: *2016 International Conference on Information Technology for Organizations Development (IT4OD)*. IEEE. 2016, pp. 1–5.

[21] T Rai. *Seven Reasons Why Native App Development is a Better Solution.* 2019. URL: https://appinventiv.com/blog/seven-reasons-native-app-development-better-solution/. (last accessed: 24.04.2020).

[22] A Sinicki. *I want to develop Android Apps — What languages should I learn?.* 2019. URL: https://www.androidauthority.com/develop-android-apps-languages-learn-391008/. (last accessed: 24.04.2020).

[23] A Dossey. *Kotlin vs. Java: Which is the Better Option for Android App Development?.* 2019. URL: https://clearbridgemobile.com/java-vs-kotlin-which-is-the-better-option-for-android-app-development/. (last accessed: 24.04.2020).

[24] J Aniola. *Java vs. Kotlin — Part 1: Performance.* 2019. URL: https://medium.com/@bards95/comparative-evaluation-of-selected-constructs-in-java-and-kotlin-part-1-dynamic-metrics-2592820ce80. (last accessed: 24.04.2020).

[25] John Chirillo and Edgar Danielyan. *Sun certified security administrator for Solaris 9 & 10 study guide.* McGraw-Hill, Inc., 2005.

[26] Ravi S Sandhu and Pierangela Samarati. "Access control: principle and practice". In: *IEEE communications magazine* 32.9 (1994), pp. 40–48.

[27] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. "Password strength: An empirical analysis". In: *2010 Proceedings IEEE INFOCOM*. IEEE. 2010, pp. 1–9.

[28] Google. *Secure your site with HTTPS.* 2020. URL: https://support.google.com/webmasters/answer/6073543?hl=en. (last accessed: 24.04.2020).

[29] Luke Probasco. *ENCRYPTION REQUIREMENTS FOR BANKS FINANCIAL SERVICES.* 2017. URL: https://info.townsendsecurity.com/encryption-requirements-for-banks-financial-services. (last accessed: 24.04.2020).

[30] investopedia.com. *Open Banking.* 2019. URL: https://www.investopedia.com/terms/o/open-banking.asp. (last accessed: 24.04.2020).

[31] openbanking.com. *What is open banking?.* 2019. URL: https://www.openbanking.org.uk/customers/what-is-open-banking/. (last accessed: 24.04.2020).

[32] openbanking.com. *Third party providers.* 2019. URL: https://www.openbanking.org.uk/providers/third-party-providers/. (last accessed: 24.04.2020).

[33] truelayer.com. *Third party providers.* 2019. URL: https://truelayer.com/openbanking/. (last accessed: 24.04.2020).

[34] J Pritchard. *What Is Open Banking (and How Will It Affect You)?.* 2019. URL: https://www.thebalance.com/what-is-open-banking-and-how-will-it-affect-you-4173727. (last accessed: 24.04.2020).

[35] Emma Technologies. *Meet Emma.* 2020. URL: https://emma-app.com/. (last accessed: 24.04.2020).

[36] Money Dashboard. *Staying on top of your money is easy with the UK's best personal finance app.* 2020. URL: https://www.moneydashboard.com/. (last accessed: 24.04.2020).

[37] YNAB. *Gain Total Control of Your Money.* 2020. URL: https://www.youneedabudget.com/. (last accessed: 24.04.2020).

[38] YNAB. *It's Your Money. Own it.* 2020. URL: https://www.meetcleo.com/. (last accessed: 24.04.2020).

[39] D Faggella. *Machine Learning in Finance – Present and Future Applications.* 2019. URL: https://emerj.com/ai-sector-overviews/machine-learning-in-finance/. (last accessed: 24.04.2020).

[40] D Soni. *Supervised vs. Unsupervised Learning.* 2018. URL: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d. (last accessed: 24.04.2020).

[41] Jan De Spiegeleer et al. "Machine learning for quantitative finance: fast derivative pricing, hedging and fitting". In: *Quantitative Finance* 18.10 (2018), pp. 1635–1643.

[42] Huseyin Ince and Theodore B Trafalis. "Kernel methods for short-term portfolio management". In: *Expert Systems with Applications* 30.3 (2006), pp. 535–542.

[43] Kyoung-jae Kim. "Artificial neural networks with evolutionary instance selection for financial forecasting". In: *Expert Systems with Applications* 30.3 (2006), pp. 519–526.

[44] Alyssa Schroer. *AI AND THE BOTTOM LINE: 15 EXAMPLES OF ARTIFICIAL INTELLIGENCE IN FINANCE.* 2020. URL: https://builtin.com/artificial-intelligence/ai-finance-banking-applications-companies. (last accessed: 09.05.2020).

[45] Android. *Android Jetpack.* 2020. URL: https://developer.android.com/jetpack. (last accessed: 23.04.2020).

[46] Matthew Dolan. *Android Security: SQL injection with the Room Persistence Library.* 2017. URL: https://medium.com/@appmattus/android-security-sql-injection-with-the-room-persistence-library-69f4e286960f. (last accessed: 23.04.2020).

[47] Sunil Kuntal. *SQLite Made Easy : Room Persistence Library.* 2017. URL: https://medium.com/mindorks/sqlite-made-easy-room-persistence-library-ecd1a5bb0a2c. (last accessed: 23.04.2020).

[48] Android. *Navigation.* 2020. URL: https://developer.android.com/guide/navigation. (last accessed: 22.04.2020).

[49] Android. *Intents and Intent Filters.* 2020. URL: https://developer.android.com/guide/components/intents-filters. (last accessed: 24.04.2020).

[50] Android. *MediatorLiveData.* 2020. URL: https://developer.android.com/reference/android/arch/lifecycle/MediatorLiveData. (last accessed: 08.05.2020).

[51] Kotlin. *Kotlin Coroutines.* 2020. URL: https://kotlinlang.org/docs/reference/coroutines-overview.html. (last accessed: 24.04.2020).

[52] Android. *RecyclerViews.* 2020. URL: https://developer.android.com/guide/topics/ui/layout/recyclerview. (last accessed: 22.04.2020).

[53] Pixplicity. *Sharp.* 2020. URL: https://github.com/Pixplicity/sharp. (last accessed: 06.05.2020).

[54] Android. *ViewPager.* 2020. URL: https://developer.android.com/guide/navigation/navigation-swipe-view. (last accessed: 23.04.2020).

[55] Android. *TabLayout.* 2020. URL: https://developer.android.com/reference/com/google/android/material/tabs/TabLayout. (last accessed: 23.04.2020).

[56] square. *Android Times Square.* 2020. URL: https://github.com/square/android-times-square. (last accessed: 09.05.2020).

[57] Android. *Spinner.* 2020. URL: https://developer.android.com/guide/topics/ui/controls/spinner. (last accessed: 23.04.2020).

[58] Android. *Create Deep Links to App Content.* 2020. URL: https://developer.android.com/training/app-links/deep-linking. (last accessed: 25.04.2020).

[59] Retrofit. *Retrofit.* 2020. URL: https://square.github.io/retrofit/. (last accessed: 25.04.2020).

[60] TrueLayer. *TrueLayer Security.* 2020. URL: https://truelayer.com/security/. (last accessed: 06.05.2020).

[61] Tensorflow. *Tensorflow.* 2020. URL: https://www.tensorflow.org/. (last accessed: 06.05.2020).

Appendices

Appendix A

Appendix A: Ethics Approval

A.1 Official Ethics Approval

Downloaded: 09/05/2020
Approved: 15/04/2020

Alex Morris
Registration number: 170164838
Computer Science
Programme: Computer Science MComp, 3rd year

Dear Alex

PROJECT TITLE: Personal Money Manager Using A.I.
APPLICATION: Reference Number 031586

On behalf of the University ethics reviewers who reviewed your project, I am pleased to inform you that on 15/04/2020 the above-named project was **approved** on ethics grounds, on the basis that you will adhere to the following documentation that you submitted for ethics review:

- University research ethics application form 031586 (form submission date: 14/04/2020); (expected project end date: 06/05/2020).
- Participant consent form 1076944 version 2 (14/04/2020).
- Participant consent form 1076940 version 3 (14/04/2020).
- Participant consent form 1075488 version 2 (08/04/2020).

If during the course of the project you need to deviate significantly from the above-approved documentation please inform me since written approval will be required.

Your responsibilities in delivering this research project are set out at the end of this letter.

Yours sincerely

Com Ethics
Ethics Administrator
Computer Science

Please note the following responsibilities of the researcher in delivering the research project:

- The project must abide by the University's Research Ethics Policy: https://www.sheffield.ac.uk/rs/ethicsandintegrity/ethicspolicy/approval-procedure
- The project must abide by the University's Good Research & Innovation Practices Policy: https://www.sheffield.ac.uk/polopoly_fs/1.671066!/file/GRIPPolicy.pdf
- The researcher must inform their supervisor (in the case of a student) or Ethics Administrator (in the case of a member of staff) of any significant changes to the project or the approved documentation.
- The researcher must comply with the requirements of the law and relevant guidelines relating to security and confidentiality of personal data.
- The researcher is responsible for effectively managing the data collected both during and after the end of the project in line with best practice, and any relevant legislative, regulatory or contractual requirements.

A.2 Ethics Application

Application 031586

Section A: Applicant details

Date application started:
Sun 3 November 2019 at 15:10

First name:
Alex

Last name:
Morris

Email:
amorris3@sheffield.ac.uk

Programme name:
Computer Science MComp, 3rd year

Module name:
COM3610
Last updated:
15/04/2020

Department:
Computer Science

Applying as:
Undergraduate / Postgraduate taught

Research project title:
Personal Money Manager Using A.I.

Has your research project undergone academic review, in accordance with the appropriate process?
No

Similar applications:
- not entered -

Section B: Basic information

Supervisor

Name	Email
Olakunle Olayinka	o.olayinka@sheffield.ac.uk

Proposed project duration

Start date (of data collection):
Mon 20 April 2020

Anticipated end date (of project)
Wed 6 May 2020

3: Project code (where applicable)

Project code
- not entered -

Suitability

Takes place outside UK?
No

Involves NHS?
No

Health and/or social care human-interventional study?
No

ESRC funded?
No

Likely to lead to publication in a peer-reviewed journal?
No

Led by another UK institution?
No

Involves human tissue?
No

Clinical trial or a medical device study?
No

Involves social care services provided by a local authority?
No

Is social care research requiring review via the University Research Ethics Procedure
No

Involves adults who lack the capacity to consent?
No

Involves research on groups that are on the Home Office list of 'Proscribed terrorist groups or organisations?
No

Indicators of risk

Involves potentially vulnerable participants?
No
Involves potentially highly sensitive topics?
No

Section C: Summary of research

1. Aims & Objectives

My aim for this project is to create a mobile application to aid in managing personal finances. Students will be the primary target audience. The application will present all of a user's financial information in one place, in an easily digestible format and provide features that will help the them to understand and manage their finances.The application will use machine learning techniques to power some of it's features. I plan on using human participants to test the usability of my application and to provide feedback on the design. Any financial information used during testing will be mock data; no real financial data will be used at any point in the project.

2. Methodology

There will be a couple of methods of gathering data from subjects.
1) I intend on running a survey on how students currently manage their finances. I will also show them a prototype of my application and ask questions on whether they would find such an app helpful and what features they would use/desire.
2) I plan on asking the subjects to download and play with my application and then provide feedback on the design and features by completing a questionnaire.
3) Users will be asked to complete tasks while being observed by the researcher to see where they encounter problems or experience confusion.

Due to the current situation with COVID-19 all testing will happen virtually - there will be no face-to-face contact with any participants.

3. Personal Safety

Have you completed your departmental risk assessment procedures, if appropriate?

Not applicable

Raises personal safety issues?

No

The research will be conducted in the University during normal hours and doesn't involve any dangerous activities or threatening people.

Section D: About the participants

1. Potential Participants

Any Student who is in university at Sheffield.

2. Recruiting Potential Participants

The study will be advertised on student Facebook groups and pages, the posts will include a google form where they can sign up and give their details. Volunteers will be given information on the tasks required for the testing and the amount of time it would be expected to take.

2.1. Advertising methods

Will the study be advertised using the volunteer lists for staff or students maintained by CiCS? No

- not entered -

3. Consent

Will informed consent be obtained from the participants? (i.e. the proposed process) Yes

Participants will provide consent by completing a google form with their personal details and an electronic signature.

4. Payment

Will financial/in kind payments be offered to participants? No

5. Potential Harm to Participants

What is the potential for physical and/or psychological harm/distress to the participants?

Very low potential for any physical and/or psychological harm, just a slight time inconvenience.

How will this be managed to ensure appropriate protection and well-being of the participants?

Testing will be conducted online to allow students the least inconvenience in their day and prevent any COVID -19 spread.

Section E: About the data

1. Data Processing

Will you be processing (i.e. collecting, recording, storing, or otherwise using) personal data as part of this project? (Personal data is any information relating to an identified or identifiable living person).
Yes

Which organisation(s) will act as Data Controller?

University of Sheffield only

2. Legal basis for processing of personal data

The University considers that for the vast majority of research, 'a task in the public interest' (6(1)(e)) will be the most appropriate legal basis. If, following discussion with the UREC, you wish to use an alternative legal basis, please provide details of the legal basis, and the reasons for applying it, below:

- not entered -

Will you be processing (i.e. collecting, recording, storing, or otherwise using) 'Special Category' personal data?
No

3. Data Confidentiality

What measures will be put in place to ensure confidentiality of personal data, where appropriate?

The personal data will only be recorded in the process of gaining consent, and will not be published publicly or made accessible to anyone other than myself and my supervisor.

4. Data Storage and Security

In general terms, who will have access to the data generated at each stage of the research, and in what form

Only me, and my supervisor, will have access to personal data at any point during the research. Data gathered from the surveys and tests will be published without the names or information of the subjects.

What steps will be taken to ensure the security of data processed during the project, including any identifiable personal data, other than those already described earlier in this form?

The information will be kept in secure storage, to which only I will have permission to access. Either stored locally on my device, or on google forms.

Will all identifiable personal data be destroyed once the project has ended?
Yes

Please outline when this will take place (this should take into account regulatory and funder requirements).

1 month after publication

Section F: Supporting documentation

Information & Consent

Participant information sheets relevant to project?
No

Consent forms relevant to project?
Yes

Document 1076944 (Version 2) **Sample Questions**	All versions
Document 1076940 (Version 3) **Information Sheet**	All versions
Document 1075488 (Version 2)	All versions

Additional Documentation

External Documentation

- not entered -

Section G: Declaration

Signed by:
Alex Morris
Date signed:
Tue 14 April 2020 at 12:17

Offical notes

- not entered -

A.3 Supporting Documents

A.3.1 Information Sheet

UREC May 2018

Participant Information Sheet

Personal Money Manager Using AI

1. Invitation paragraph

You are being invited to take part in a research project. Before you decide whether or not to participate, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this

2. What is the project's purpose?

The purpose of the project is to create and test a personal finance mobile app that uses A.I elements to help users stay on budget. The project is being completed as a dissertation for my degree program.

3. Do I have to take part?

'It is up to you to decide whether or not to take part. If you do decide to take part you will be given this information sheet to keep (and be asked to sign a consent form) and you can still withdraw at any time without any negative consequences. You do not have to give a reason. If you wish to withdraw from the research, please contact me at amorris3@sheffield.ac.uk*

4. What will happen to me if I take part? What do I have to do?

Participants will only need to be involved for the time it takes them to complete the allotted tasks set for them depending on the stage of development/testing.

1) In stage 1 you will be asked to fill in a questionnaire about how you currently manage your finances and what features you would use/desire in a budgeting app. Estimated time ~5 mins.

2) In stage 2 you will be asked to download and complete tasks with the mobile application and then provide feedback on the design, features and any errors encountered by completing a questionnaire. Estimated time ~20 mins.

3) In stage 3 participants will be asked to test the app by completing tasks while being observed by the researcher, over a video call, to see where they encounter problems or experience confusion. Estimated time ~30 mins.

The majority of questions asked in all stages will be multiple choice/closed answer style and there will be a few optional open answer questions where subjects can give further feedback or depth if they want to.

5. What are the possible benefits of taking part?

There are no immediate benefits for those people participating in the project although it may provide some education or inspiration for those looking to better manage their finances and it is hoped that the work may lead to a tool that could be used by Students in the future to save money and improve their financial and mental wellbeing.

6. Will my taking part in this project be kept confidential?

All the information that we collect about you during the research will be kept strictly confidential and will only be accessible to members of the research team. You will not be identified in any reports or publications.

7. What is the legal basis for processing my personal data?

'According to data protection legislation, we are required to inform you that the legal basis we are applying in order to process your personal data is that 'processing is necessary for the performance of a task carried out in the public interest' (Article 6(1)(e)). Further information can be found in the University's Privacy Notice https://www.sheffield.ac.uk/govern/data-protection/privacy/general.'

8. What will happen to the data collected, and the results of the research project?

Data collected will be used to improve the design of my application, the results of the testing and answers to the questionnaires will be published but no personal information will be made available to anyone outside the research team. Identifiable personal data will be deleted as soon as the project is published.

9. Who is organising and funding the research?

The research is not funded and is organised by me (Alex Morris) as part of a dissertation project for my Computer Science undergraduate degree.

10. Who is the Data Controller?

The University of Sheffield will act as the Data Controller for this study. This means that the University is responsible for looking after your information and using it properly.

11. Who has ethically reviewed the project?

This project has been ethically approved via the University of Sheffield's Ethics Review Procedure, as administered by the Department of Computer Science

12. What if something goes wrong and I wish to complain about the research?

If you have any complaints about the research please contact my supervisor, Mr Olakunle Olayinka,
at o.olayinka@sheffield.ac.uk. If you wish to escalate your complaint further please contact the head of department, Prof Guy Brown, at g.j.brown@sheffield.ac.uk.

If the complaint relates to how the participants' personal data has been handled, information about how to raise a complaint can be found in the University's Privacy Notice: https://www.sheffield.ac.uk/govern/data-protection/privacy/general.

13. Contact for further information

If you have any further questions please contact me or Ola with the details below:
Alex Morris :
Email - amorris3@sheffield.ac.uk
Mobile - 07903987656

Mr Olakunle Olayinka :
Email - o.olayinka@sheffield.ac.uk

Participants will be given a copy of this sheet + their signed consent form.

UREC May 2018

Thanks for taking part – Alex

A.3.2 Consent Form

Personal Money Manager using AI using AI Consent Form

Please tick the appropriate boxes	**Yes**	**No**
Taking Part in the Project		
I have had the project has been fully explained to me and understand what will be involved. (If you will answer No to this question please do not proceed with this consent form until you are fully aware of what your participation in the project will mean.)	☐	
I have been given the opportunity to ask questions about the project.	☐	
I agree to take part in the project. I understand that taking part in the project will include using a mobile application, taking a questionnaire or being interviewed.	☐	
I understand that my taking part is voluntary and that I can withdraw from the study at any time. I do not have to give any reasons for why I no longer want to take part and there will be no adverse consequences if I choose to withdraw.	☐	
How my information will be used during and after the project		
I understand my personal details such as name, phone number, address and email address etc. will not be revealed to people outside the project.	☐	
I understand and agree that my words may be quoted in publications, reports, web pages, and other research outputs. I understand that I will not be named in these outputs unless I specifically request this.	☐	
I understand and agree that other authorised researchers will have access to this data only if they agree to preserve the confidentiality of the information as requested in this form.	☐	
I understand and agree that other authorised researchers may use my data in publications, reports, web pages, and other research outputs, only if they agree to preserve the confidentiality of the information as requested in this form.	☐	
So that the information you provide can be used legally by the researchers		
I agree to assign the copyright I hold in any materials generated as part of this project to The University of Sheffield.	☐	

Name of participant [printed]	Signature	Date
Name of Researcher [printed]	Signature	Date

If you have any further questions please contact me or my supervisor with the details below:

Alex Morris :

Email - amorris3@sheffield.ac.uk

Mobile – 07903987656

Supervisor

Mr Olakunle Olayinka :

Email - o.olayinka@sheffield.ac.uk

A.3.3 Sample Questions

Sample Question Sheet

Stage 1: Pre Requirements Survey

Link to google form:

https://forms.gle/hj7F49p6rwyXLRsTA

Stage 2: Initial Testing + Feedback on incomplete application.

Link to google form:

https://forms.gle/xHMopqufGeZnqqA2A

Stage 3: Usability Testing + Feedback on complete application.

Subjects will be asked to perform tasks under observation:

1) Please add a new savings goal and then generate a weekly allowance.
2) Please add a new account through open banking.
3) Please change the password of the current account and then log out.
4) **Optional** Do you have any other feedback on the usability and design of the app and can you suggest any areas of improvement or features that could be added?

Appendix B

Appendix B: Consent Form + Information Sheet

Participant Information Sheet

Personal Money Manager Using AI

1. Invitation paragraph

You are being invited to take part in a research project. Before you decide whether or not to participate, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this

2. What is the project's purpose?

The purpose of the project is to create and test a personal finance mobile app that uses A.I elements to help users stay on budget. The project is being completed as a dissertation for my degree program.

3. Do I have to take part?

'It is up to you to decide whether or not to take part. If you do decide to take part you will be given this information sheet to keep (and be asked to sign a consent form) and you can still withdraw at any time* without any negative consequences. You do not have to give a reason. If you wish to withdraw from the research, please contact me at amorris3@sheffield.ac.uk

4. What will happen to me if I take part? What do I have to do?

Participants will only need to be involved for the time it takes them to complete the allotted tasks set for them depending on the stage of development/testing.

1) In stage 1 you will be asked to fill in a questionnaire about how you currently manage your finances and what features you would use/desire in a budgeting app. Estimated time ~5 mins.

2) In stage 2 you will be asked to download and complete tasks with the mobile application and then provide feedback on the design, features and any errors encountered by completing a questionnaire. Estimated time ~20 mins.

3) In stage 3 participants will be asked to test the app by completing tasks while being observed by the researcher, over a video call, to see where they encounter problems or experience confusion. Estimated time ~30 mins.

The majority of questions asked in all stages will be multiple choice/closed answer style and there will be a few optional open answer questions where subjects can give further feedback or depth if they want to.

5. What are the possible benefits of taking part?

There are no immediate benefits for those people participating in the project although it may provide some education or inspiration for those looking to better manage their finances and it is hoped that the work may lead to a tool that could be used by Students in the future to save money and improve their financial and mental wellbeing.

6. Will my taking part in this project be kept confidential?
All the information that we collect about you during the research will be kept strictly confidential and will only be accessible to members of the research team. You will not be identified in any reports or publications.

7. What is the legal basis for processing my personal data?
'According to data protection legislation, we are required to inform you that the legal basis we are applying in order to process your personal data is that 'processing is necessary for the performance of a task carried out in the public interest' (Article 6(1)(e)). Further information can be found in the University's Privacy Notice https://www.sheffield.ac.uk/govern/data-protection/privacy/general.'

8. What will happen to the data collected, and the results of the research project?
Data collected will be used to improve the design of my application, the results of the testing and answers to the questionnaires will be published but no personal information will be made available to anyone outside the research team. Identifiable personal data will be deleted as soon as the project is published.

9. Who is organising and funding the research?
The research is not funded and is organised by me (Alex Morris) as part of a dissertation project for my Computer Science undergraduate degree.

10. Who is the Data Controller?
The University of Sheffield will act as the Data Controller for this study. This means that the University is responsible for looking after your information and using it properly.

11. Who has ethically reviewed the project?
This project has been ethically approved via the University of Sheffield's Ethics Review Procedure, as administered by the Department of Computer Science

12. What if something goes wrong and I wish to complain about the research?
If you have any complaints about the research please contact my supervisor, Mr Olakunle Olayinka,
at o.olayinka@sheffield.ac.uk. If you wish to escalate your complaint further please contact the head of department, Prof Guy Brown, at g.j.brown@sheffield.ac.uk.

If the complaint relates to how the participants' personal data has been handled, information about how to raise a complaint can be found in the University's Privacy Notice:
https://www.sheffield.ac.uk/govern/data-protection/privacy/general.

13. Contact for further information
If you have any further questions please contact me or Ola with the details below:
Alex Morris :
Email - amorris3@sheffield.ac.uk
Mobile - 07903987656

Mr Olakunle Olayinka :
Email - o.olayinka@sheffield.ac.uk

Participants will be given a copy of this sheet + their signed consent form.

Thanks for taking part – Alex

*Required

1. Email address *

2. I have read and understood the information sheet - (Select yes to continue) *

Mark only one oval.

- Yes
- No

Consent Form

Please carefully read and answer the questions below, to participate in the project the answer to every statement must be 'Yes'

3. Please select the appropriate answer *

Mark only one oval per row.

	Yes	No
The project has been fully explained to me and understand what will be involved.	◯	◯
I have been given the opportunity to ask questions about the project.	◯	◯
I agree to take part in the project. I understand that taking part in the project will include using a mobile application, taking a questionnaire or being interviewed.	◯	◯
I understand that my taking part is voluntary and that I can withdraw from the study at any time. I do not have to give any reasons for why I no longer want to take part and there will be no adverse consequences if I choose to withdraw.	◯	◯
I understand my personal details such as name, phone number, address and email address etc. will not be revealed to people outside the project.	◯	◯
I understand and agree that my words may be quoted in publications, reports, web pages, and other research outputs. I understand that I will not be named in these outputs unless I specifically request this.	◯	◯
I understand and agree that other authorised researchers will have access to this data only if they agree to preserve the confidentiality of the information as requested in this form.	◯	◯
I understand and agree that other authorised researchers may use my data in publications, reports, web pages, and other research outputs, only if they agree to preserve the confidentiality of the information as requested in this form.	◯	◯
I agree to assign the copyright I hold in any materials generated as part of this project to The University of Sheffield.	◯	◯

Appendix C

Appendix C: Budget Survey

Budget Survey

All forms submitted will remain anonymous, if you do not feel comfortable answering any questions then feel free to leave them blank.

1. I feel in control of my financial situation

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

2. My finances are a regular source of worry for me

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

3. How would you rate your financial knowledge?

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Weak	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Strong

4. Do you use a budget to help manage your finances?

Mark only one oval.

- ◯ I have never made a budget
- ◯ I have made a budget in the past but don't regularly use one
- ◯ I use a budget but struggle to stick to it or keep it updated
- ◯ I regularly use a budget and comfortably stick to it and keep it updated

5. What do you use to track your budget?

Tick all that apply.

- ☐ I don't track my budget
- ☐ Pen + Paper
- ☐ Spreadsheet
- ☐ Mobile App
- ☐ Desktop App

Other: ☐ _____

6. What period of time would you ideally prefer to budget around/ set an allowance for?

Mark only one oval.

- ◯ Weekly
- ◯ Monthly
- ◯ Annually
- ◯ Other: _____

7. Do you think you would find a mobile budget app helpful?

Mark only one oval.

	1	2	3	4	5	
Definitely Not	◯	◯	◯	◯	◯	Definitely/Use one already

8. What platform is your mobile device on?

Mark only one oval.

- ◯ I don't have a mobile/smartphone
- ◯ Android
- ◯ iOS/Apple
- ◯ Windows
- ◯ Other: _____

9. What features would you find desirable in a budgeting app?

Tick all that apply.

	Needed	Desirable	Not Helpful
Auto Generate Budget	☐	☐	☐
Manually add accounts/transactions etc	☐	☐	☐
Allocate budget into categories	☐	☐	☐
Automatically link to bank accounts	☐	☐	☐
Warning if you are projected to go over budget	☐	☐	☐
Track saving goals	☐	☐	☐
Track income + bills (inc student loan)	☐	☐	☐
Allow bank statements to be uploaded via .csv files	☐	☐	☐
See past spending + performance against your budget over time	☐	☐	☐
Financial Advice	☐	☐	☐

10. Are there any other features you would like to see in a budgeting application?

_____

_____

_____

_____

_____

Appendix D

Appendix D: Evaluation Questionaire

Budget App Testing

Installing the app

You will need an Android device to install and run the app.

The app is installed using the .apk file found here - https://drive.google.com/open?id=1KnR79AmH7P0_oz7JaJnAfvGY7dmnyXsq

Downloading the file on an Android device should prompt an install wizard that will install the app.

On most phones you will need to enable the "Allow installation from Unknown Sources" option. You should hopefully get prompted to turn the option on when opening the file, If not try this - https://www.technipages.com/where-did-allow-installation-from-unknown-sources-go-in-android

If you have any trouble installing the app, or have any other questions then please message me on facebook or email me at amorris3@sheffield.ac.uk

Budget App Testing

Please complete the set tasks using the app and answer the questions, all responses are anonymous. If you're struggling to complete a task for more than a couple of minutes then select the 'unable to complete task' option and move on :)

Task 1 - Manually add both a new account and a transaction.

1. How easily were you able to complete the task

Mark only one oval.

	1	2	3	4	5	
Unable to complete the task	◯	◯	◯	◯	◯	Easily

2. Do you have any feedback on how the feature could be improved?

Task 2 - Add a linked account from the Account page

3. How easily were you able to complete the task

Mark only one oval.

	1	2	3	4	5	
Unable to complete the task	◯	◯	◯	◯	◯	Easily

4. Do you have any feedback on how the feature could be improved?

Task 3 - Delete or edit one of the generated accounts

5. How easily were you able to complete the task?

Mark only one oval.

	1	2	3	4	5	
Unable to complete the task	◯	◯	◯	◯	◯	Easily

6. Do you have any feedback on how the feature could be improved?

Task 4 - Generate a budget

7. How easily were you able to complete the task?

Mark only one oval.

	1	2	3	4	5	
Unable to complete the task	◯	◯	◯	◯	◯	Easily

8. The budget is generated by calculating the amount of money that needs to be saved each week/month so that all the saving goals are achieved while also ensuring that all the bills can be payed and that the balance never dips below the "Saved" amount. This is done based on their current total balance and expected income for the period up to the final saving goal. How useful would you find a budget generated in this way?

9. Do you have any feedback on how the feature could be improved?

General Feedback + Bugs

10. Would you be likely to use the app in its current form?

Mark only one oval.

	1	2	3	4	5	
Unlikely	◯	◯	◯	◯	◯	Very Likely

11. How do you rate the general look and feel of the app?

Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Very Good

12. How would you rate the overall quality of the app?

Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Very Good

13. How responsive did you find the app?

Mark only one oval.

	1	2	3	4	5	
Unresponsive	◯	◯	◯	◯	◯	Very Responsive

14. Did you encounter any errors or unusual behaviour while using the app? Could you describe them?

15. Do you have any other feedback on any aspect of the app?