Major Project - Money Tracker
Final Report

Author:	Steven Twerdochlib (stt31)
Supervisor:	Chris Loftus (cwl)
Degree Scheme:	G400 Computer Science Bsc
Module Code:	CS39440
Config. Ref.:	MMP.FinRep
Date:	1st May 2020
Version:	1.0
Status:	Release

Department of Computer Science,
Aberystwyth University,
Aberystwyth,
Ceredigion, SY23 3DB,
U.K.

CONTENTS

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Steven Twerdochlib

Date: 1st May 2020

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth computer Science Department.

Name: Steven Twerdochlib

Date: 1st May 2020

Acknowledgements

I'd like to thank my supervisor Chris Loftus for his consistent feedback and ideas throughout the progression of my project in regard to improvements, alternative solutions and overall help.

I'd like to also thank all the user testers that took part in the questionnaire created for this project.

Thank you for everything.

Abstract

In a majority of people's daily life, people often spend more money than they originally think. According to a CNBC article [9] which did a survey, 79 percent of consumers who said they had a budget ended up failing to stick to that budget. This is obviously not in the consumer's interest so they need a way of tracking how they spend their money.

Most of these consumers who are in need of a way to track their money spendage use online tools such as money recording apps or simply their bank statements. Depending on the app they download they can get a variety of different features to help them manage their finances, however most of these apps tend to be either overly complicated with their features by making the user fill in a lot of information that the user isn't interested in or is too vague such as, only showing the sum of multiple items bought. Furthermore, most apps don't provide a way for multiple users to utilize the same finances which can be an important feature for users who have a shared budget.

Upon the full implementation of the project, a user will be able to input simple information regarding an item they have bought or sold and view all these transaction whenever desired. Moreover, the user will have a way of sharing the information they have stored with multiple users, as well as allowing those other users to also use that information. The app will also provide the basic necessities for interacting with the stored user information such as; filtering through the items as desired by the user.

1 Introduction

1.1 Objectives

The objectives of this document are:

- To give a conclusion on for all the project deliverables created throughout the project.
- To provide in depth detail about the implementation of the Final System deliverable.
- To describe and explain the strengths and weaknesses of my performance throughout this project.
- To describe and explain the strengths and weaknesses of the projects progression, as well as the problems that occurred during the project.
- To sum up the result of the project in a short paragraph.

1.2 The Project Product

The product of this project will be an android app designed to allow users to keep track of their finances and share those finances with others when desired. On start-up of the app, the user has a variety of simple options that they can choose from, these include;

- Creating a new Google Drive plain-text file database to store item information.
- Add new item information to any Google Drive plain-text file database of their choice.
- View old item information from any Google Drive plain-text file database of their choice.

If the user has previously accessed a database on the app then they will also have the options to;

- Add new item information to that previously accessed database without having to search for it again in their Google Drive.

- View old item information that previously accessed database without having to search for it again in their Google Drive.

The user is able to create a Google Drive database by selecting the option and entering a name for the new database, once entered the user just needs to submit the title and the database will be created.

The user is able to add item information to a Google Drive plain-text file by either; manually inputting the necessary item information required to make the item information useful for future use or automatically inputting the necessary item information using a bar code scanner. Once inputted all the user has to do is confirm their inputs by submitting the items they have created. The necessary item information to be inputted is;

- Item Name
- Item Category
- Item Amount
- Date of Purchase
- Payment Option used
- Payee (Whoever bought the item)

The user will also be able to view the item information from a Google Drive plain-text file by selecting the option, once selected, the user can view all the key item information stored in the Google Drive database and has the option to expand the items to show all of an the expanded items information. The key information shown without expanding an item is; the item name, the item category and the item amount.

While viewing a Google Drive database, the user has the option to filter through the items displayed. To filter through the items displayed the user has to select the option and then input the filters they desire, the filters have the following limits;

- Can filter by as many item names as desired.
- Can filter by as many item categories as desired.
- Can filter by as many payees as desired.
- Can filter by only a single item amount range.
- Can filter by only a single purchase date range.
- Items are filtered by an "Any" filter feature and NOT an "All" filter feature.

Furthermore, while viewing a Google Drive database, the user has the option to delete items individually, the user can only delete an item if they are looking at an up-to-date version of the database so if they tried to delete an item while the database wasn't up-to-date they would be shown a popup that notifies them of this problem and the app will update the database they are currently looking at.

Outside of the app, the user has the ability to go into their Google Drive and share their created databases with other people, once shared the shared users can access the database using the projects app just like any other Google Drive database or they can use the information stored in the Google Drive database any other way they desire.

Internet is not required for this app as it works on an offline version when there is no internet connection, however this may cause problems in unintentional data loss when using Google Drive databases on multiple devices but a warning is conveyed to the user whenever this is a possible issue.

More information on the project can be found in the Project Outline document [Appendix 1].

1.3 Example Product Scenario

An example of when this app would be used is after a going shopping, the user will add the items that they recently bought to a Google Drive database. This allows the user keep track of what they have spent money on and how much they have spent in given time frames.

2 Background

2.1 Ethics

Within finance, having someone know what another person has been spending their money on can be dangerous in many ways such as; determining a users daily schedule or using that information to find answers to security questions like "What's your favourite food?", often asked for in other websites. Upon completion of the project, each user will have full control on the databases that they create in terms of the contents and who can view or edit those contents. This means that the user will still be able to change the permissions of any shared users for any databases they have created to reduce risk to themselves and the user will be able to add any user that they deem appropriate to see their information.

All data on/from individuals follows the General Data Protection Regulation (EU) 2018 which essentially sets out the principles, rights and obligations for most processing of personal data. To ensure that the data will be gathered in an appropriate process, an ethics form [Appendix 8] was filled out and sent to a professional to examine it to make sure that none of the General Data Protection Regulations are being broken.

2.2 Money recording apps

The aim of all money recording apps is to store information on how the user is spending and/or receiving money and display that information back to the user, when desired, in a way that is useful to the user. The most common information used in popular money recording apps when the wants to input information are; the items name, a category for the item, the items cost, the payment option used when buying that item and the date the item was bought. When displaying information back to the user, a large majority of money recording apps utilise pie charts and graphs to show this information appropriately, however all of these apps also had a default display showing all the items information as simple text.

2.3 Issues with current apps

It is important to note that the issues mentioned below aren't actually issues but rather areas of dislike by different target audiences.

- For some users who are using the app for some organisation in their finances, they may dislike most of the money recording apps available as they offer too many features, for example Figure 1 shows a money recording app called 'Monthly Budget Planner & Daily Expense Tracker' which provides 8 different possible features available when the user may only ever use about 2. Moreover Figure 2 shows something similar for the same money recording app, where the screen is filled with too many different features such as the pie chart, which may be unnecessary to the user. All these different features cause the user to be confused and makes the learnability of the apps to be too much effort for the common user.
- Most money recording apps don't have a feature which provides multiple users to share their information together and work together, this can cause obvious problems with people with shared budgets or joint accounts.

- Some money recording apps are too vague, particularly money recording apps for specific banks such as; Santander Online Banking app and Nationwide app (See Figure 3 for an example of how vague transactions are for a Santander Banking App). These apps record transactions automatically based off bank card usage, which is desired by some target audiences, however this also cause the problem of information for items being lacking in detail towards individual users and most of these apps also focus on a total amount spent rather than the individual items bought.

- Most money recording apps require the user to input item information manually and individually, this is beneficial for some users as it makes sure the information for each item is definitely inputted and so can always be accessed. On the other hand, some user may see this as time consuming and thus an inconvenience.

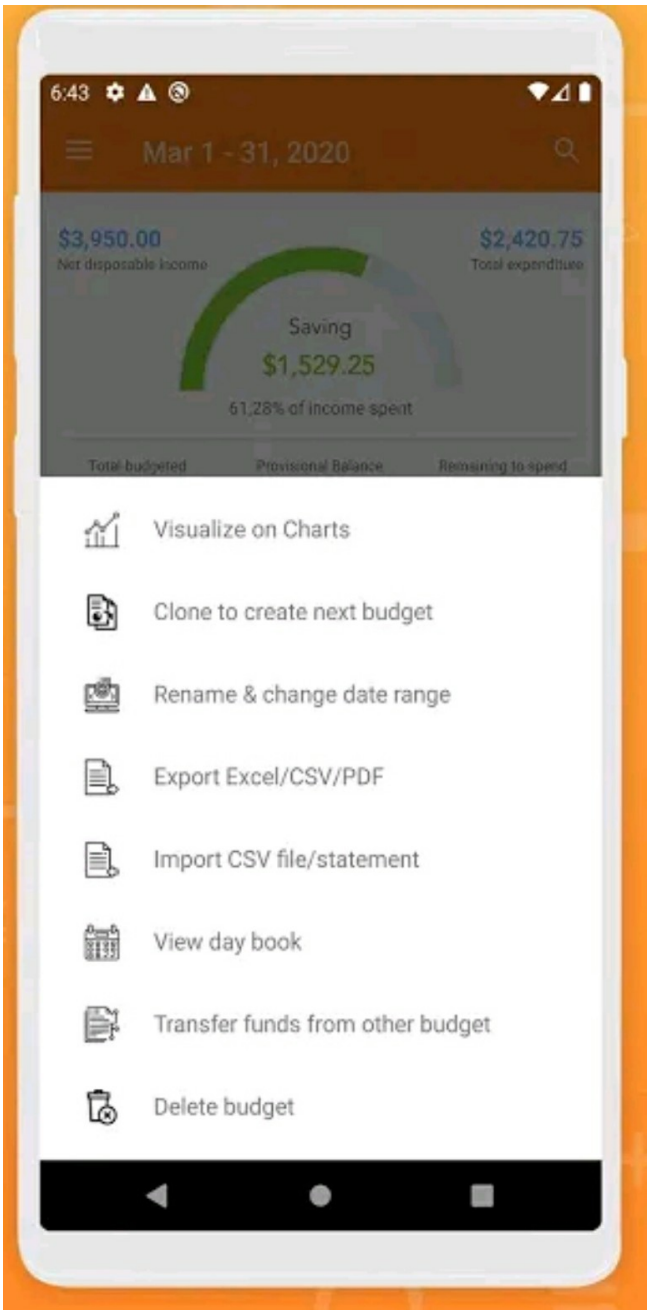


Figure 1 - Too many unnecessary features available.

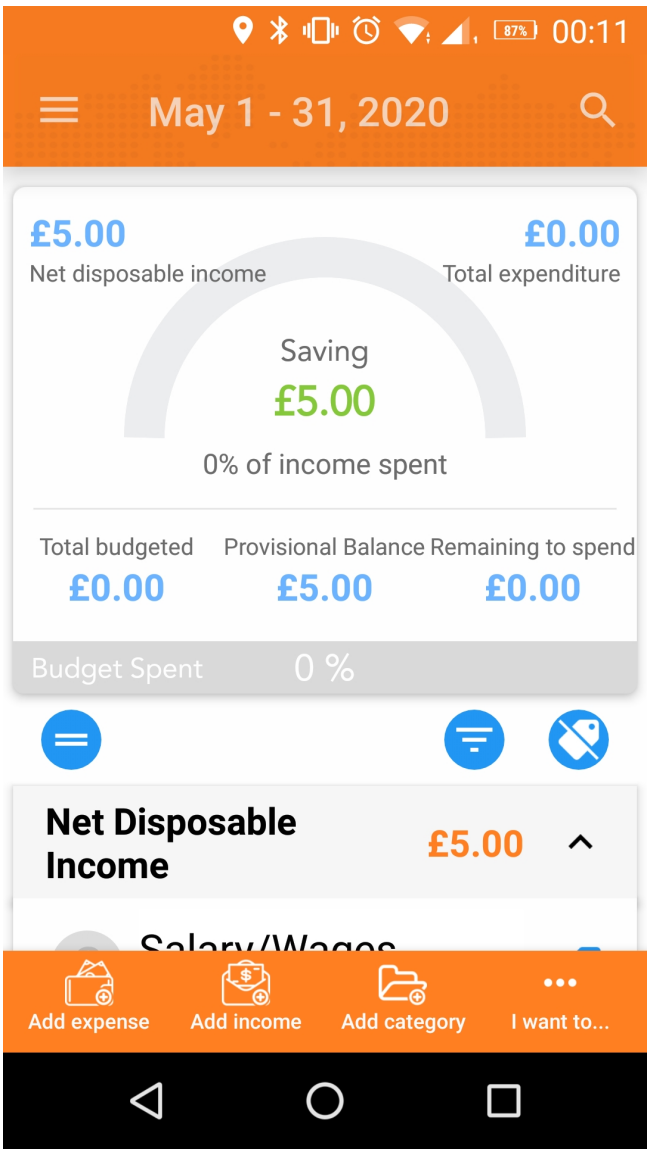


Figure 2 - Too many undesired features filling up screen

Figure 3 - Santander Transaction Example

2.4 Motivation and interest

The motivation behind this project choice was to create something interesting and useful in everyday life that puts into practice the skills to adapt and research unknown methods rather than using facts taught in practical modules at University. This means rather than reusing what had been done in University practical's, instead researching new ways of producing similar results was more interesting.

Furthermore, the developer's family is mostly made up of computer scientists and accountants, which meant that as a key developer in the project, he was greatly influenced with his ideas by his upbringing which was involved with a lot of organisation, particularly around finances. Having these connections provided possible, suitable testers for the project which also influenced the project theme towards a more financial theme.

Once the theme of the project had been decided, the next stage was to find a goal for the project. As Aberystwyth University was involved in the project, this lead to thinking about what university students could use that related to their finances as it also produced a variety of possible, suitable testers. However, it was desired to keep the goal of the project useful to a wider target audience than just university students as to not limit the project. All this, lead to a money recorder app idea, as most university students tend to not keep track of how their spending their money but also the app could be used by anyone wanting to keep track of their transactions.

Moreover, knowing that this project was being created by a university student, rather than being a perfect app by the end of the project, instead it was desired that the project produced a piece of software that showed clear signs of possible improvement. This was because it shows that the app can still grow to suit the target audience more and even expand to more audiences.

With this sort of project, common techniques in Android Studio can be used in the creation of the user interface design, which is taught in Mobile Development with Android module (CS31620).

3 Process

3.1 Lifecycle chosen

The lifecycle model used for this project was an iterative waterfall lifecycle.

The iterative waterfall lifecycle is an extension of the waterfall lifecycle with a few modifications made to improve the performance of development from a more practical and realistic viewpoint. The key stages of any iterative waterfall model are; Planning and Requirements, Analysis and Design, Implementation, Software Testing, Maintenance and Evaluation, respectively.

The reasons an iterative waterfall lifecycle was chosen are as follows:

The main difference between the waterfall model and the iterative waterfall model is that the iterative waterfall model allows the developers to go back to previous stages and make changes, if necessary. This is extremely useful and more realistic as is allows changes to be made in previous phases based off any problems that occur later on in the project which is likely to appear in a University project and also allows us to make changes in previous phases based on user testing feedback which is ideal for improving the software.

A disadvantage of the iterative waterfall model is that it does not allow overlapping of phases which can often result in poor time efficiency. However, due to the fact that this project will be created by a single person, this is no longer a problem as working on multiple stages at once would be practically impossible for a single developer.

Another advantage to the iterative waterfall model compared to other models is that the single person in creating the project is most experienced in using the waterfall lifecycle so by using a similar model (iterative waterfall model), that person won't have a difficult time adjust to the new process.

Furthermore, the iterative waterfall model can be very costly for late-stage changes, on the other hand, this is not an issue as for this project, since the time available for the project is relatively small, late-stage changes won't be as costly and most likely be quicker to apply than if the project was much larger.

A disadvantage for using an agile approach is that it tends to have an extreme lack of necessary documentation because requirements are clarified just in time for development. This means that future development becomes much harder as misunderstandings and difficulties can easily occur when trying to convey what was attempted or even completed [5].

More reasons as to why the iterative waterfall lifecycle was chosen can be found in the Project Plan document [Appendix 3], as well as other details on how this lifecycle will affect the project.

This iterative waterfall lifecycle was good for the project as it provided set deadlines to make sure that all areas of work were given a necessary amount of time. It also gave much more organisation to the project through the use of these deadlines as it allowed the developer to focus on specific deliverables rather than losing track of progress by trying to attack all the deliverables at once [6].

3.2 Schedule

To stick to a clear schedule that follows an iterative waterfall lifecycle, Gantt charts were made throughout the project. These Gantt charts were update throughout the project to; show the necessary changes required to the schedule, add more detail to an existing Gantt chart and/or to improve estimates for incomplete phases.

These Gantt charts show a rough design of how tasks were separated however, unlike a usual iterative waterfall model, these Gantt charts have overlapping phases, it is important to note that these overlapping phases were not done simultaneously, but rather it was the developers choice on how much time was spent on each phase as long as they both were completed by the deadline set.

The Gantt charts produced throughout the project can be found in the Project Plan document [Appendix 3], where the details on their creation and changes are explained in more detail.

3.3 Blog

The blog [3] is a website where there have been weekly updates on the progress of the project. Each weekly update contains information on what has been done that week (Starting Wednesday) and provides ideas on what work is being considered for the following week, however this does not mean that the work will be done but just for consideration as more ideas are being considered during the weekly meetings with the supervisor.

3.4 Version Control

In order to keep track of the software over different updates so to have a back-up provided data was lost/modified unintentionally, a version control system was necessary. The version control system used for this project was GitHub [2]. GitHub is a free online version control system that is easily accessible and provides backpedalling of the software and documents to previous versions.

4 Requirements

The requirements phase of the iterative waterfall lifecycle is an important part of the lifecycle as, although this lifecycle allows backtracking to previous stages, it is not recommended still to make any big changes late in the lifecycle.

The core requirements for the final system of the project are:

- Must have at least 3 screens created that are easily accessible and interchangeable; Home screen, Input screen and Display screen.
- Must allow the user to choose the Google Drive plain-text file that they wish to use.
- Must allow the user to manually input all necessary items information.
- Must provide an option to allow the user to use a bar code scanner to automatically fill in items information.

- Must send item information to a desired Google Drive plain-text file when desired by the user.
- Must retrieve data from the desired Google Drive plain-text file and display it appropriately when desired by the user.
- Must provide options on filtering the data being displayed to the user.

It was originally planned that the final system would connect to a Google Sheet instead of a Google Drive plain-text file. This was changed due to problems occurring when try to connect to a Google Sheet and altering its contents; this is explained in more detail in the [Prototype 1 section] further on in the document.

These requirements, as well as other requirements (optional and external), are explained in more detail in the Requirement Specification document [Appendix 2].

5 Design

There are 2 parts to the design phase of the iterative waterfall methodology that is being followed throughout this project; user interface (UI) design and system design. UI design is about the process of making the interfaces for devices with a focus on looks and style. The aim of all UI design is to create designs that users find easy to use and pleasurable. System design definition is the process of developing, expressing, documenting, and communicating the realization of the architecture of the system through a complete set of design characteristics described in a form suitable for implementation.

5.1 Initial UI Design

The Initial Design is the first completed version of the UI design, before any user testing is done with the UI design. The goal of the Initial UI Design is to set the foundation of the UI design so that it is suitable to send out to the testers and software developers, to provide feedback for improvements that wouldn't be, otherwise known, as obvious. More detail on the user testing and its results can be found in the [Testing section] and the Test Document document [Appendix 5].

To make the Initial UI Design suitable for testers, it has a fluid navigation between slides so the user can get a good idea of how the navigation of the app would be. Moreover, to stick to one of the aims of the project, to keep the app produced simplistic, the number of interactions to perform any action in the design, was kept to a minimum. More reasons as to the design decisions can be found in the Final UI Design [Appendix 7].

5.2 Final UI Design

The Final UI Design [Appendix 7] is the UI design created for the developers of this project to aim to achieve during the creation of the final system. It is important to note that this Final UI Design may not be what is achieved in the actual final system as time constraints and other problems may occur, preventing features from being implemented successfully.

The Final UI Design [Appendix 7] was meant to be a modified version of the initial UI design; however the final UI Design is exactly the same as the Initial UI Design created due to the fact that there was no time available to make the desired changes to make the UI more suitable to users. The Final UI Design was going to be modified based on the results of the user testing done. More detail on the user testing and its results can be found in the [Testing section] and the Test Document document [Appendix 5].

More detail on the Final UI Design can be found in the project deliverable Final UI Design [Appendix 7], which is a PowerPoint presentation designed with key screen navigation and notes.

5.3 System Design

The system design is a design for how the architecture of the final system code will be built based on, it includes what classes are to be created, what variables and functions will be inside these classes and how classes link together to make up the final systems architecture. System diagrams are commonly designed using UML diagrams and so for this project a UML was used to depict a vague structure of the final systems architecture.

The UML diagram (See Figure 4) created contains 4 classes that utilise inheritance. Inheritance is used between these classes so that all the classes can efficiently access the Google Drive and select files, the classes are;

- CreateFileClass
- RetrieveContentsClass
- RewriteContentsClass
- LogoutClass

As well as inheritance classes, the UML (See Figure 4) shows a variety of activities, these activities are used for displaying the necessary UI to perform and implement features. These activities include;

- MainActivity
- NoInternetPopup
- NewFilePopActivity
- FilterPopupActivity
- ViewDatabaseActivity
- AddItemsActivity

More information on the system design and the UML diagram created can be found in the Design Specification document [Appendix 4].

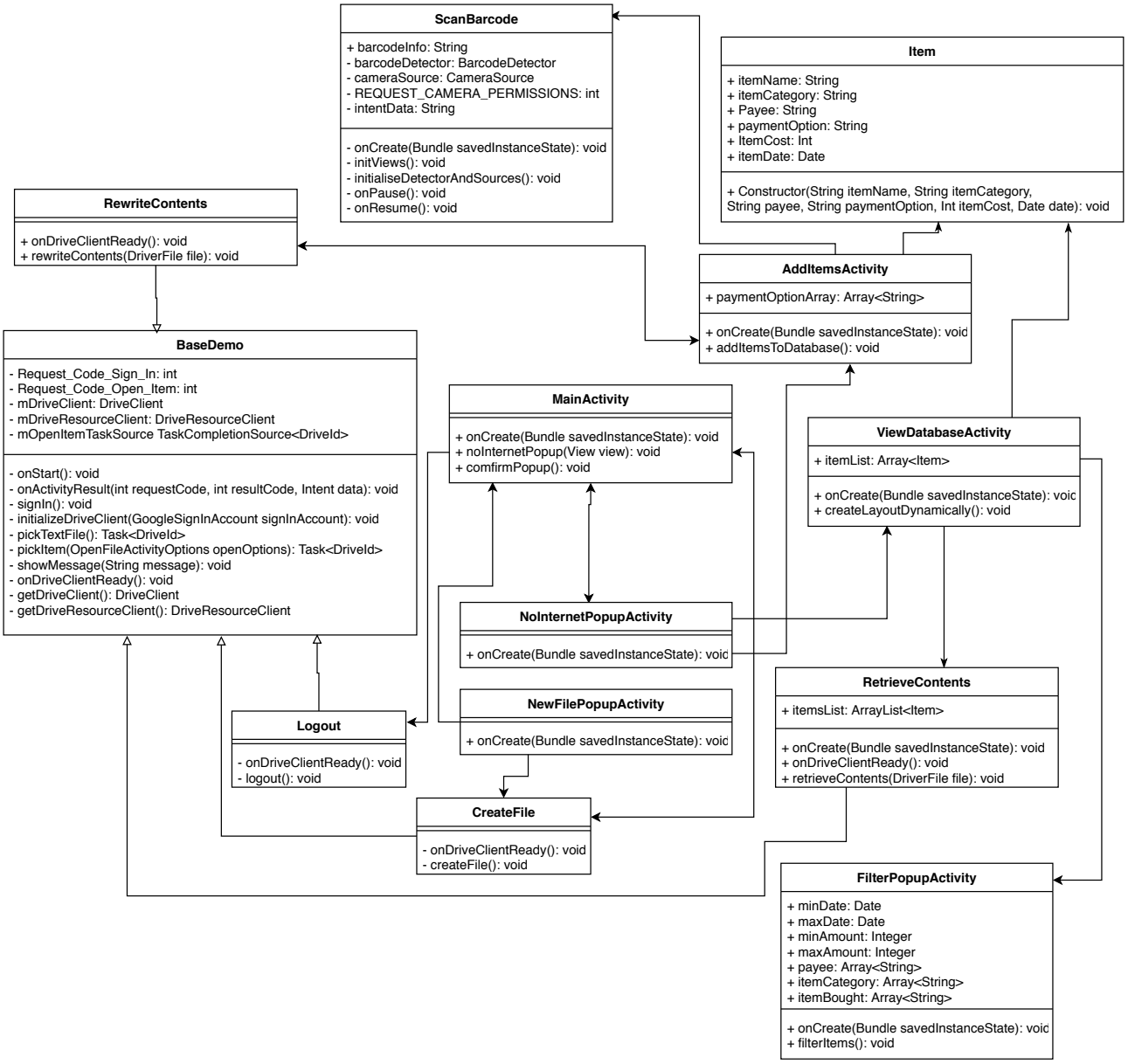


Figure 4 - UML Diagram

6 Prototypes

The prototypes for this project are reusable pieces of software that focus on fulfilling specific key features that will be implemented in the final system. The aim of these prototypes is to provide insight into whether a key feature is more difficult to implement than first anticipated and therefore provide more of a risk assessment than is usually possible in an iterative waterfall methodology. Furthermore, the prototypes are also created for the mid-project demonstration so that a clear sign of progress can be shown, as well as the future development to be done. More information on the mid-project demonstration can be found in the [Critical Evaluation, Demonstration section] further on in this document.

6.1 Prototype 1

Prototype 1 is a reusable piece of code that focused on the feature of utilising the Google Drive, this involves; connecting to the users Google account, creating Google Drive files, retrieving data from a Google Drive file, display data from a Google Drive file and alter the contents of a Google Drive

file. The purpose of this specific prototype was to get a better understanding of how to go about interacting with the Google Drive and finding potentional risks for implementation. Figures 5-8 show screenshots of Prototype 1 and its features.

While attempting to retrieve and alter the contents of a Google Sheet file numerous errors occurred, these erros involved being unable to directly alter the contents of a Google Sheet from outside of the Google Drive. While trying to solve these errors a decent amount of time had gone by with little to no progress so an alternative solution had to be made regarding how the data would be stored. After multiple solutions were considered, connecting to a Google Drive plain-text file instead of a Google Sheet was thought to be the best of the solution as it still allowed the usage of some of the information already researched and not as big of a change would have had to be made in most areas of development as it is still related to the Google Drive being used to store information. Other solutions considered were;

- using a NoSQL database - The problem with this solution is that sharing a NoSQL database between multiple users whenever and with whomever the user desires will take time and is not well experienced within the developer and so there is a lot of risk of having the same problem as the Google Sheet.

- using a Firebase database - The problem with this solution was that it would lose out on some features that Google Drive allows such as easability in sharing files because of how the Google Drive has been designed.

- continue to try to solve the errors involving the Google Sheet - The problem with this solution is that if the errors are not solved then a lot of time will have been wasted and a core part of the project will be lost.

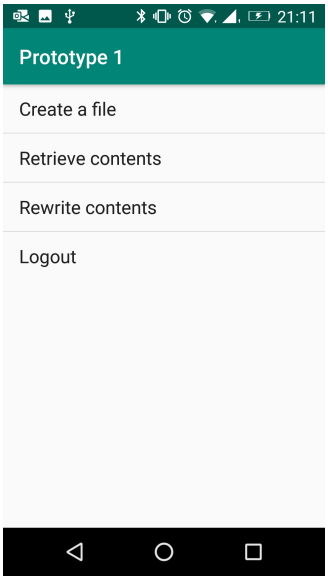


Figure 5 - Prototype 1 Home Screen

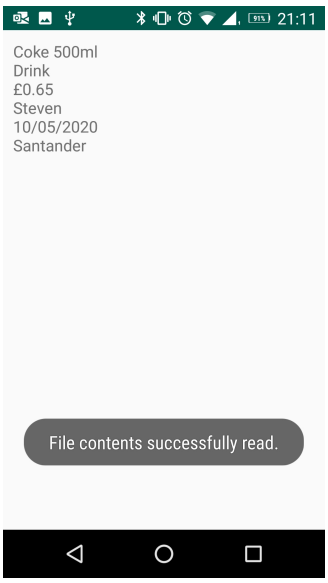


Figure 6 - Prototype 1 Retrieve Contents Feature

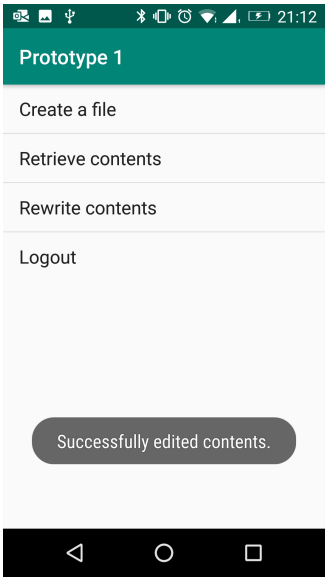


Figure 7 - Prototype 1 Rewrite Contents Feature

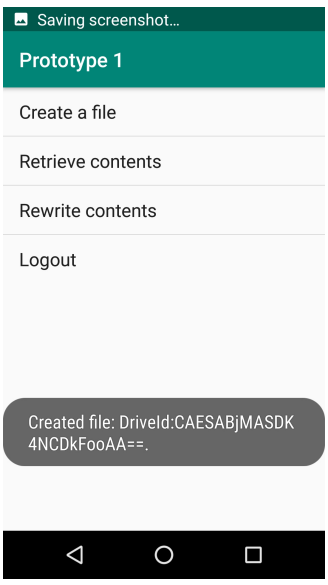


Figure 8 - Prototype 1 Create File Feature

6.2 Prototype 2

Prototype 2 is a reusable piece of code that focused on the feature of retrieving information from a bar code. The purpose of this specific prototype was to get a better understanding of how to go about implementing a bar code scanner and finding potentional risks for implementation. Figures 5-6 show screenshots of Prototype 2 and its features.

When this prototype was being created it was assumed that the software would retrieve a string from the bar code that would contain all the information required. However, after the completed version of the prototype 2 was completed and shown to the supervisor it became clearer that what the original intent was that the barcode would contain a string of a unique id and the software would connect to a database and retrieve information using this unique code. The reason a database would be an important asset to reading the bar code scanner feature is because, a bar code increases in width depending on the size of the string it contains and so this would make the bar codes unscannable if the string is too long. Despite having noticed the misunderstanding there was no time to change the Prototype 2 to utilise the use of a database and so this would've had to be implemented straight into the final system if time was available, more information on how this implementation went can be found in the [Implementation, Phase 3 - Bar Code Scanner section].

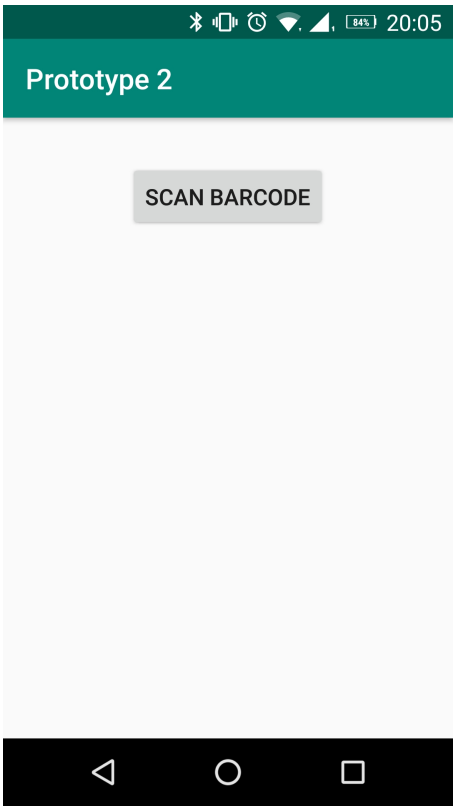


Figure 9 - Prototype 2 Home Screen

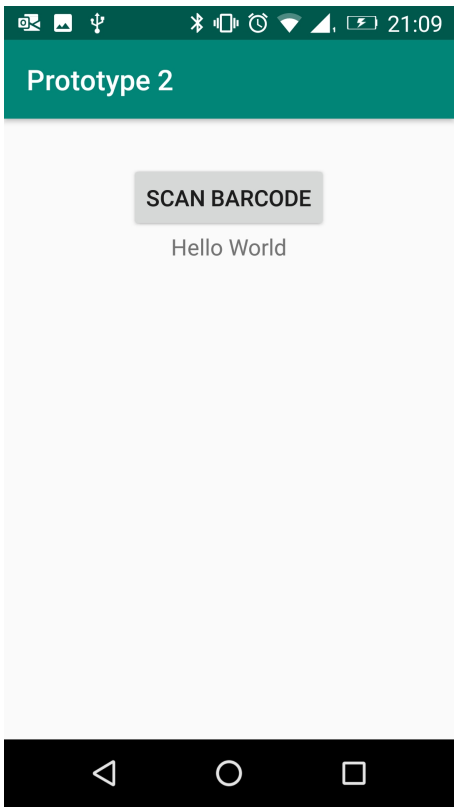


Figure 10 - Prototype 2 Scan Bar Code Feature

6.3 Prototype 3

Prototype 3 is a reusable piece of code that focused on recreating the Final UI Design [Appendix 7] but in Android Studio. The aim of this prototype was to check if all the desired UI was possible to implement for the final system and to give a better idea to see if all the UI was possible to implement within the time frame available. The purpose of this specific prototype was to get a better understanding of how to go about implementing a the UI in an actual android studio environment and to find risks that may occur during implementation and to see if it was possible to implement all of the UI features within a designated time frame. Figures 11-14 show screenshots of Prototype 3 and its features.

Prototype 3 was unfinished because there was not enough time to add all the UI that was decided upon in the Final UI Design [Appendix 7] but knowing this was beneficial as it meant the developer could think ahead before the implementation phase as to how they adjust what had been made in Prototype 3 so that a working UI could be created. More detail on how the UI had changed for the final system can be found in the [Implementation section].

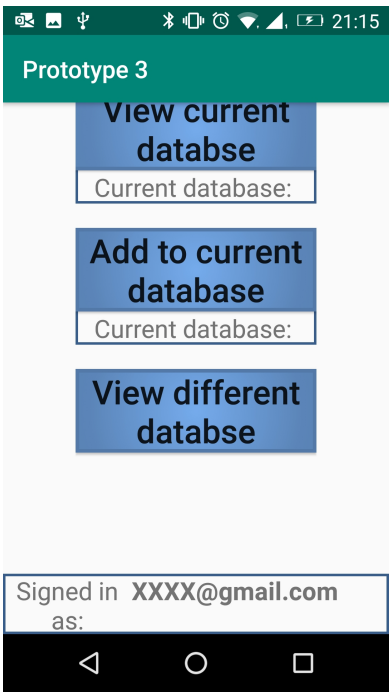


Figure 11 - Prototype 3 Home Screen

Figure 12 - Prototype 3 Add Items Feature

Figure 13 - Prototype 3 View Database Feature

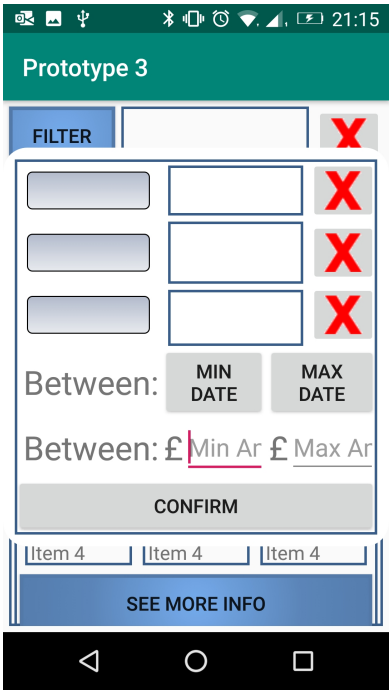


Figure 14 - Prototype 3 Filter Feature

7 Implementation

This implementation section goes into detail about the process of the implementation, as well as any problems that occurred throughout the implementation and how these problems were dealt with.

7.1 Phase 1 - UI

The first phase of the implementation, following the Gantt charts created [Appendix 3], was to use and modify the Prototype 3 code to create the UI for the final system. As mentioned in the previous

section of this document, Prototype 3, it was made clear previous to starting the implementation that not all the UI features would be possible to implement in the time available, to solve this many different alteration were made to the UI;

- The infinite scroll UI for bank options was changed so that it only scrolls in one direction.
- Dropdowns/spinners were changed into EditText views.
- More info icons for viewing hidden item information was changed to text.

As well as these changes to the UI to save time, a variety of other UI issues were discovered which are explained below:

In Prototype 3 it was not considered how different device screen sizes would affect the UI of the project product, this would be a big problem as Prototype 3 was not designed for an average device but rather a large device and so would greatly limit the possible user base for the product. To solve this problem, the solution chosen was to add a ScrollView layout around then entire screen; this allows the user to scroll down the screen so that all the UI is reachable to the user even if it isn't immediately visible to the user on start-up. There is still a necessary screen size to run the project product, mentioned in the Requirement Specification document [Appendix 2] but this solution at least improved the possible user base by a large amount.

Another feature that was in Prototype 3 was that the filtering feature was displayed using a popup window (See Figure 14), this caused a problem as it was unintentional that not all filters could not be seen at once by the user due to different device screen sizes. A solution to this was to simply added a ScrollView like mentioned previously, however since it was desired that the user could see all their filters at once before submitting for a better UI design, it was decided to instead turn the filter popup feature into a separate screen to be displayed instead. This solves the problem as long as the user sticks to the recommended screen size mentioned in the Requirement Specification document [Appendix 2].

Overall, a working UI design was successfully implemented by the end of phase 1 and the only UI that was missing were ones that had to be created during the creation of other features such as but not limited to; a camera view for scanning a bar code and displaying the filtering options chosen. See Figures 15-21 to see screenshots of what the apps UI looks like.

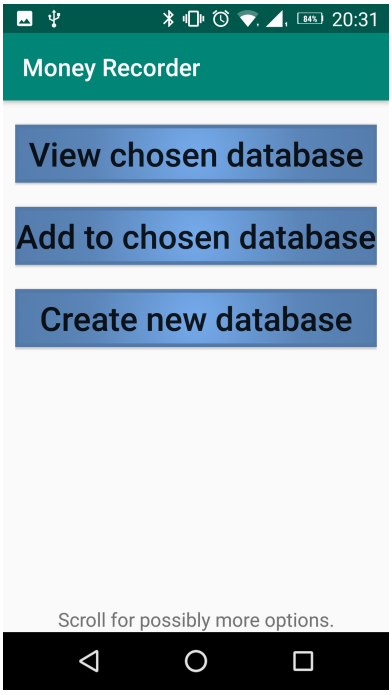


Figure 15 - Signed Out Home Screen

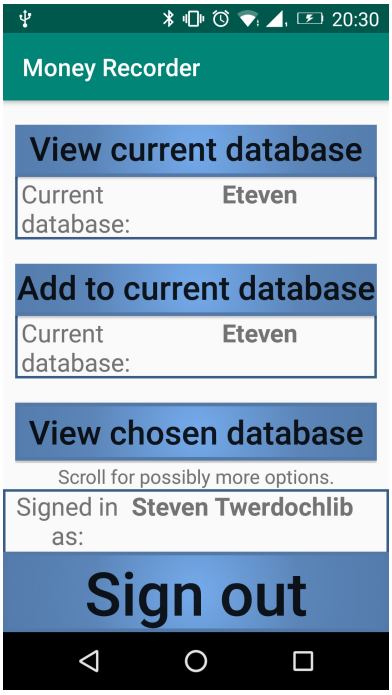


Figure 16 - Signed In Home Screen

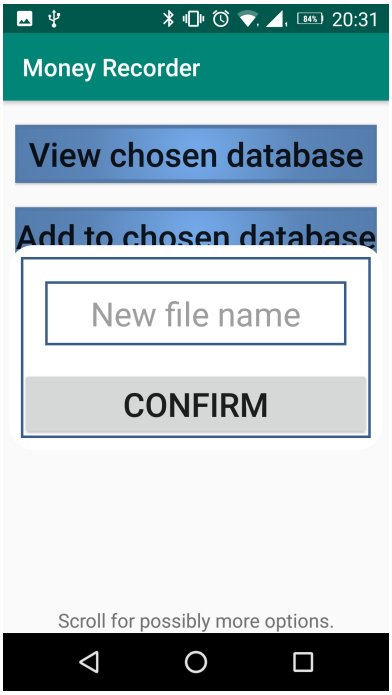


Figure 17 - Create New Database Screen

Figure 18 - Add Items Screen

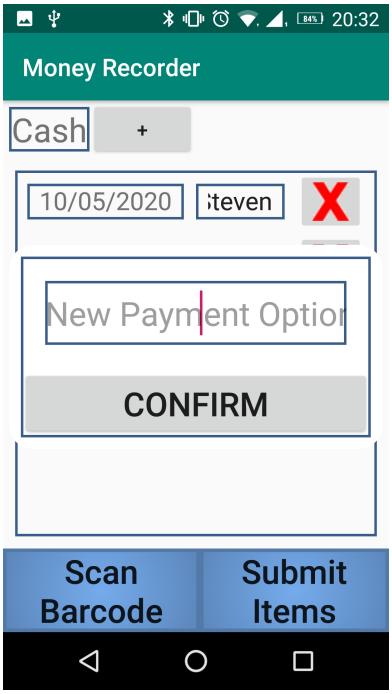


Figure 19 - Create New Payment Type Screen

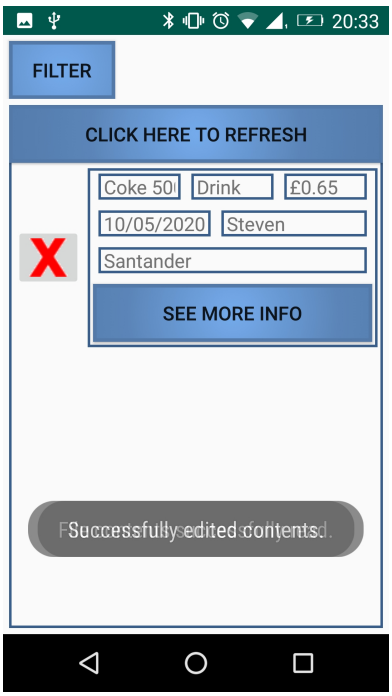


Figure 20 - View Database Screen

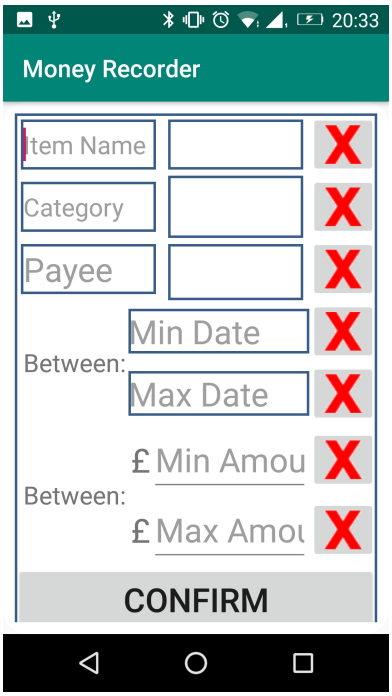


Figure 21 - Filter Screen

7.2 Phase 2 - Utilising Google Drive

The second phase of the implementation, following the Gantt charts created [Appendix 3], was to utilise the Google Drive, this involved connecting the existing UI to add the features of; adding item information to a Google Drive file, retrieving and displaying information from a Google Drive file and connecting/disconnecting to a users Google account.

Due to the code produced in Prototype 1, this phase was much easier to implement that originally thought as key classes didn't have to be altered as they already fit the required needs of the final system. Some changes in the final system compared to Prototype 1 were necessary to make the code adapt for the final system, these adaptions are explained below;

In Prototype 1, the app will bring up the Google Drive every time the user wants to perform an action on a file in the Google Drive, this however, is not necessary in the final system as it is desired that the final system will remember what file is being worked used and automatically choose that file. As the Google Drive was required to be opened for the code to then retrieve the up-to-date file that was known to be being worked on, the Google Drive activity couldn't be not opened, however an alternative solution was decided upon were the Google Drive activity was made invisible/transparent to the user. By making the activity invisible to the user, this allowed the activity to still be opened for the software to retrieve the up-to-date file while preventing the user from seeing the process.

In addition to this, in Prototype 1 the user could create a file in their Google Drive with a pre-determined name, this is not desired for the final system as it is desired that the user can enter their own name choice for a file their creating. To alter the code to suit the final system the string used in Prototype 1 for the pre-determined file name had to be changed so that it contained a new string during runtime based off an input for the user.

Another important alteration that had to be made in Prototype 1's code to make it suitable for the final system is that different activities had to be opened depending on the when a file was retrieved or altered. For example, when the user wants to view a Google Drive file's contents then the activity needs to change so that the contents of the file can be displayed on the other hand, when the user wants to add an item to a Google Drive file's contents then the software must retrieve the file's contents again to make sure that the file being altered is up-to-date without changing the activity to display the contents of that file. This change was made by using Boolean variables to determine if certain interactions where being made, such as a specific button being clicked, and using this Boolean to run similar but slightly different code suited to the action being preformed.

Overall, connecting to the Google Drive and Google account and creating the necessary features that utilised them was successfully and the user can now sign in and sign out of their Google account, can add and remove items from a Google Drive database and multiple users working on the same database (Google Drive file) will not cause issues with losing data unexpectedly.

7.3 Phase 3 - Bar Code Scanner

The next phase of the implementation, following the Gantt charts created [Appendix 3], was to implement a bar code scanner feature were the user will be able to scan a bar code and the item information stored in the bar code will be automatically added to a visible list for user submission.

Due to the code produced in Prototype 3, most of the necessary code could be added, however because of the misunderstanding on how the information is retrieved from the bar code, found after creating the Prototype 3 an important feature of creating a database that utilises a unique id retrieved from the bar code. More information on this misunderstanding discovered can be found in the Prototype 3 section of this document. To add this database feature it was decided upon to use a Firebase database because Firebase provides real-time changes which would be extremely useful for the final system as the database is essentially a third-party database that the average users is unlikely to have access to but will need to be able to update any time for each individual user. The concern with adding this database feature is that it could limit the number of potential partners that the final system could work with as most other companies will have a their own, already created database and so by creating a specific database for my final system I would be limiting the number of potential partners to those who would be willing to take the time to move their databases to my final systems database. It would be more ideal that a database and code to read the database is add once a partnership with another company is decided so that the code can be specified for those individual companies. On the other hand, this feature was still chosen to be implemented as it is extremely useful for partners who haven't yet created a database and/or users who want to make their own bar codes.

The database section of this feature unsuccessful due to errors in attempting to connect to a Firebase database and unfortunately because of the time available it had to be scraped, however for maintenance

purposes an alternative solution was made where a hardcoded database was implemented directly into the final system using 'if statements'. This solution provides future developers knowledge on what type of information would be retrieved and how unique ID's from bar codes would be used to find specific items in the database. More detail on how further development to implement this feature can be found in the Maintenance Document document [Appendix 6] and the [Maintenance section] of this document.

Overall, the final system can read information from a bar code successfully, but it does not utilise the use of a database making this feature very unreliable. However, precautions were taken to provide clear instructions on how to go about adding a database for future development.

7.4 Phase 4 - Filtering, Caching and Other

The next phase of the implementation, following the Gantt charts created [Appendix 3], was to implement the rest of the minor, but still necessary, features that make the final system more worthwhile to users. These minor features included but not limited to; filtering items based on users choice, caching important information for use on restarting the app, bug fixes and UI improvements. All these features were combined into a single phase because it was difficult to determine how much time would be required to implement them all.

The filtering feature wasn't difficult to implement but it was surprisingly time consuming, this resulted in it not being fully completed but finished to a standard where the feature is still usable. The filtering activity allows the user to input multiple; item names, item categories and payee names as intended as well as an item cost range and an item date range. On inputting filters the user has the option to remove specific filter before submitting the filters or to submit the filters inputted, the unfinished part of the filtering feature is that once the filters have been submitted, they cannot be individually removed and so only all the filters can be removed together. The filtering feature works by checking all the inputted filters against each of the items in the database being looked at, if the item matches at least one of the filters then it is added to a list which displays the filtered items. It is important to note that only a single filter is need to pass for an item to be added to the filtered list and so even if an item fails another filter it will still be added.

The caching feature was a fairly straight forward process, it was decided to be done using Shared Preferences to store variables as it's relatively easy to implement and provides the necessary features to store the required detail. Shared Preferences is a way in which one can store and retrieve small amounts of primitive data as key-value pairs to a file on the device storage. The most difficult part of this feature was storing a file DriveId as a DriveId is not a primitive data type, but this was overcome by encoding the DriveId to a String on saving and decoding that String on retrieval.

https://www.geeksforgeeks.org/shared-preferences-in-android-with-examples/

One bug fix that was made to the final system was that before the fix, there was a bug where the money amounts displayed were not displayed in a money format. To solve that the amount being displayed simply had to be altered from its current format to a more appropriate format using a DecimalFormatter.

To improve the usability of the app it was desired to make the software constantly check the Google Drive file being used for any changes while the data from it was being viewed. This feature was attempted using an AsyncTask and a Service, however both attempts failed, it is assumed that these attempts failed due to being unable to change variables in a different activity while being in a background thread and running an activity despite being in a background thread. Some of the code was left in the final system as comments to allow future developers to have an idea of what was attempted in hopes of them overcoming the issue.

Overall, the final system is able to save and load data for future use successfully and the number of bugs throughout the final system has been reduced. Furthermore, the filtering feature has been successfully

implemented, however there is clear signs of improvement available for this feature in particular as there was not enough time to add those improvements. These improvements are mentioned in more detail in the Maintenance Document document [Appendix 6] and the [Maintenance section].

7.5 Multi-user and Offline features

This section explains how the final system provides multi-user access to files and offline access.

The final system allows multiple users to access the same Google Drive file and use just as any other file with limited errors occurring for unintentional data lose. To allow the users to share files they have to go into their own Google accounts a share the files with other, desired Google account users. To add data to a Google Drive file without losing data unintentionally, the final system retrieves data from the file just before adding information to it so that it is never deleting data unintentionally. This process is repeated for deleting an item from a Google Drive file, the final system first retrieves the information, however for deleting an item, if the file has been changed elsewhere it is not so simple as to just retrieve the file contents again as the item to be deleted might already be deleted and therefore a notification is display instead to let the user know that the contents they are viewing has been updated and they need to re-delete the item if it's still there.

The final system also provides an offline feature where it still connects to the Google Drive offline. This could cause errors however as when the final system regains an online connection, data in an updated file may be lost if it was altered in another device or by another user. To solve this, a simple popup notifying the user that this could occur is displayed when the user tries to alter the contents of a file without internet.

8 Testing

The Test Document [Appendix 5] explains the different testing that was done throughout the project as well as, the results and other information from each test. The different testing that was attempted throughout this project include; Beta testing, Espresso tests and System tests.

- Beta Tests - Beta testing was done for the Final Design UI project deliverable using a questionnaire [Appendix 9] that was given out, along with the UI Design, to a selected group of people to get user feedback on the overall feel of the UI design. The questionnaire was designed to get an overall idea of how the users felt about the UI design and was not intended to get specific details on what could be improved in the app to allow the testers more freedom in their answers, the Test Document [Appendix 5] explains the questions chosen in more detail.

- Espresso tests - Espresso testing was attempted in order to provide some automated testing to save time for the testers and developers, however the Espresso testing was discarded after a few attempts due to problems that occurred with interacting with the emulator when the tests were run. The Test Document [Appendix 5] explains the errors received in more detail.

- System tests - System tests was done after a complete system was created that fulfilled the major requirements. Each test has a unique ID, short description, necessary input, output produced and a pass criterion. These systems tests are expected to be run in order and it should be kept in mind that if a test fails this could potentially lead to multiple tests failing and some tests rely on others to pass. It is important to note that these system tests do not cover all possible outcomes and especially because of time constraints some features don't have any system tests related to them. More detail is provided in the Test Document [Appendix 5] on what should be done or kept in mind before and while doing these system tests.

9 Maintenance

This section breifly explains the suggestions for improvements and other key information that future developers should be aware of but all of these explanations are explained in more detail in the Maintenance Document [Appendix 6].

- Partner with a popular shopping company - This improvement involves utilising another company's database of items to allow users to scan that company's items bar codes and have the program accurately get the necessary item information required to make the bar code feature work as intended.
- Make 'All' filter option - This improvement involves making the user have the option to make items have to pass all the filters or any of the filters.
- Make the program UI more accurate to the Final UI Design deliverable [Appendix 7] - This improvement involves changing the apps UI interface so that it matches what was origionally intended, but it is recommended that the below improvement 'Edit the Final UI Design based on feedback from the user testing' is done first.
- Edit the Final UI Design [Appendix 7] based on feedback from user testing - This involves changing the origionally designed UI into something better based on what the user testers feedback entailed from the questionnaires sent out.
- Change from a Google Drive plain-text file database to a Google Sheet database - This improvement involves changing the database type from being stored in a Google Drive plain-text file to a Google Sheet database because a Google Sheet offers the user a wider range of features outside the app and so indirectly improving the projects app.

When making changes the developers have to be aware of the 2 main data areas;

- fileContents - This is a String variable which can be found in the MainActivity class, this variable is used to store a single, long String that contains all the item information for every item stored in a Google Drive plain-text file. This variable is accessed in most classes to retrieve or change information on specific items when desired.
- itemList - This is an ArrayList of Item objects which can be found in the ViewDatabase class, this variable is used to store the fileContents in a more approriate and easy to access way. This variable was meant to replace the fileContents variable but there was not enough time to make all the changes.

The Item objects mentioned above in the itemList are objects which are used to store all the information for a single item, this object class is used to make accessing specific item information easier to understand and retrieve or change.

As well as the main data areas, future developers have to also consider the order of which item information is retrieved and sent as it has to be in a specific order. It is also important to keep in mind that the order for retrieving a bar code information and a database information is different. Moreover, there are physical limitations on the current app such as; the device has to be an android device and has to have a camera to use the bar code scanner feature.

More information on suggested improvements and things to be careful of when making changes, as well as other important facts to be aware of can be found in the Maintenance Document [Appendix 6].

10 Critical Evaluation

10.1 Demonstrations

This section will explain the personal thoughts on the project demonstrations as well as the feedback received from these project demonstrations. This section will only talk about the mid-project demonstration as the final project demonstration has not yet been completed by the time of submission of this document.

The biggest fact that was received from the feedback of the mid-project demonstration was that the project wasn't using a Waterfall methodology that was first intended but rather an Iterative Waterfall methodology as the project was constantly backtracking to previous stages to make minor changes based on results from other stages further on in the lifecycle.

The mid-project demonstration also provided insight into areas of demonstrating which needed to be improved upon for the final demonstration. In the feedback from the mid-project demonstration it states that it was not clear how the projects product will differ to that of other money recording apps and the understanding of the projects technologies and technical issues were marked as 'good', but with several topics where the discussion is more general. Both these issues were because these areas simply were not discussed in enough detail and not that this information wasn't available. For the final demonstration I plan to re-explain these areas with more detail in how they work and why they were chosen in the way they were.

Overall, the mid-project didn't go as well as intended because due to panicking and lack of preparation, information was falsely conveyed across to the demonstrator and a lack of detail was brought up in the more technical areas of the project. On the other hand, the feedback and result from the demonstration as well as the experience has been more than helpful in preparing for the final project demonstration.

10.2 Personal Strengths

This section will go into detail about the strengths of the developer Steven Twerdochlib throughout this project.

The biggest strength of Steven Twerdochlib is how when an error or problem occurred during the project he was able to calmly think about whether it was worth trying to fix the problems or coming up with alternative solutions. Throughout the project Steven Twerdochlib has managed to think of numerous alternatives to problems that were considered not possible to solve during the current time constraints. These alternatives include but are not limited to;

- The changes in UI to fit the time constraint.
- Creating a hardcoded database for future developers.
- Providing pop-up's to notify the user of possible problems.
- Using a ScrollView to allow the UI to fit more devices.

10.3 Personal Weaknesses

This section will go into detail about the key weakness of the developer Steven Twerdochlib throughout this project.

The biggest weakness of Steven Twerdochlib through the project was his ability to prioritise tasks in advance. With the large time constraint on the project, prioritising the necessary features was an important aspect for the project but occasionally Steven Twerdochlib's decisions for what required

time to be worked on and how much time were often excessive or unnecessary. Some examples of unnecessary prioritise include;

- Checking the Google Drive file constantly in a background thread - Although a nice feature, it is unnecessary for producing a complete final system and so shouldn't have been attempted to save time and instead mentioned in the Maintenance Document [Appendix 6] from the start.
- User testing - Although this was a great way to find improvements in the Final UI Design [Appendix 7], considering the time constraint, it was just not possible to make use of the improvements found and so the user testing should've never been done and mentioned in the Maintenance Document instead [Appendix 6].

10.4 Self Improvement

This section will go into detail about how the developer Steven Twerdochlib learnt from this project and what changes they can make for future projects to improve their work efficiency.

To improve himself for future projects, Steven Twerdochlib has mentioned that he shall be working on his initial preparation. He believed this would be the best way to improve his work efficiency as for this project he believed that there was some research he could have done prior to the start of the project which would have greatly decreased the amount of time necessary to perform certain actions throughout the project.

10.5 Project Strengths

This section will go into detail about the key strengths of the project as a whole.

The biggest strength of the final system is the multi-user feature, this is due to the fact that most other money recording apps don't take into account multiple users sharing a budget or just want to share a database and so this gives my project product some individuality and makes it stand out compared to those other money recording apps.

The biggest strength of the project outside of the final system is that it shows clear signs of possible improvement to the final system, this means that future developers are not only given an idea of how to improve the final system upon receiving the project but also steps have been taken to making these future developments easier, such as including comments of code and small examples in the final system like the hardcoded database explained in the Implementation, Phase 3 - Bar Code Scanner.

10.6 Project Weaknesses/problems

This section will go into detail about the key weaknesses of the project as a whole.

The biggest weakness for this projects product is that it is time consuming by itself, this is due to the fact that the user of the projects product will have to manually input all item information individually because the bar codes scanned must be in a desired format which only the company that made them can make them into, as explained in the section [Implementation, Phase 3 - Bar Code Scanner].

Furthermore, a huge weakness in the project is the testing as there just seems to be a lack of it or parts of it. For example, the system tests do not cover all the the functionality of the app which is a big issue and as well as this the Espresso testing (automatic testing) had to be scraped due to problems occuring with clicking buttons automatically. Moreover, there is a lack of participants in the beta testing for the questionnaire [Appendix 9] on the Final UI Design. All these areas have resulted in the testing section of the project to become a weakness of the entire project

10.7 Project Improvements

This section explains how the project could've been improved outside of the software improvements, for information on improvements regarding the product of the project then look at the Maintenance Document document [Appendix 6] and the [Maintenance section].

To improve the project a clear improvement would be to remove the user testing as this was time consuming to perform and also the result from the user testing were not usable as there was no time available make changes based on these results. In replacement for removing the user testing, increasing the time for implementation would be the best choice, particularly the Phase 4 - Filtering, Caching and Other phase, as this section was not difficult to implement but rather it was just time consuming and so some features of it were unable to be implemented.

Another improvement would be that more thought should've gone into the Gantt chart at the start of the project as changing the deadlines for project deliverables is not ideal, especially when following the iterative waterfall methodology that the project was using.

11 Conclusion

In conclusion, this project was a success as it met all the core functional requirements even if there is a large amount of room for improvement. This project is most considered a success not only because it had met the core functional requirements, stated in the Requirement Specification [Appendix 2], but also because the project shows clear signs as to where the project will go from here on out, meaning that future development is a obviously possible.

12 Appendix

12.1 [Appendix 1] - Project Outline

Project Description

The Major Project (Money Tracker) will develope an app for android devices using Android Studio [1] software which allows the user(s) to input and view the contents of various plaint text files stored on thegoogle drive that contain information about the different things that the user(s) have spent money on. The user(s) will be able to input data manually or the app will provide a bar code scanner, which will automatically input the data once it has scanned an appropriate bar code, and then the user has the option to edit the data and/or send it to the google drive. Some research has already been done to see if these features are plausible [4].

The targeted audience for this app is anyone who is interested in keeping track of what they are spending their money on. Usually this target audience would involve people who are already have good organisational skills such as accountants and/or people who are in need of keeping track of how their spending their money due to having a little amount of it or spending it unwisely, such as university students.

For this app to be worthwhile for users it will attempt to be user-friendly and so have excellent learnability, memorablility and overall be a delight to use. This will be discussed in the design section of the project. Furthermore, besides having the key features to allow the user to use the app as intended, the app will also provide a way to utilize offline access as best as it can by saving a copy of the most recent google drive text file accessed. The key features will involve; setting up a bar code scanner, creating a new text file in the google drive, reading/writing to the new text file created and/or an already existing text file in the google drive, displaying the contents of a text file in an appropriate way and, as mentioned early, saving the most recently retrieved text file data and using that when there is no internet.

This project will use a waterfall lifecycle methodology to manage the different tasks, however unlike a usual waterfall lifecycle methodology, I will also create a few prototypes of features for the mid-project deomnstration. The early part of the project will be securing clear requirements and design, setting up an appropriate timetable for the entire project and creating a few prototypes of features. The middle part of the project will involve the actual creation of the app and using a test-last development method approach to create automated tests. The last part of the project will involve maintenance and producing a final report. The progress of the project will be recorded weekly on an online blog [3] as well as all created software and documents being stored on a GitLab [2] repository.

Proposed Tasks

The following tasks will be preformed throughout this project:

- Make and decide on requirments with a Requirement Specification document.
- Make and decide on design with a Design Specification document [Appendix 4], UI Design (interactable) and some testing.
- Make prototypes of features.
- Development of app.
  - Creating automated tests.
  - Creating the app features.
- Project meetings and project diary.
- Preparation for demonstrations.
- Run and record the results of the automated tests with a Test Specification document.
- Discuss possible maintenance and useful information on maintenance with a Maintenance Specification document.
- Discuss the final outcome of the project with a Final Report document.

Project Deliverables

The project deliverables will include the following:

- A Requirement Specification document - This document will specify the key features to be created for the app as well as its performance requirements and constraits. This document will also provide a short overview of the app being created and its use.
- A Final UI Design (interactable) with notes/explanations - This UI design (interactable) will be the a preview of how the app will look as well as give an explanation as to why specific features were included.
- The results from questionaires on the possible design features - These questionaires will be given out to a sample of people, the questionaire will involve questions based off an example UI design to provide feedback on what was most liked and disliked about the UI design.
- A Design Specification document - This document will specify the underlying structure of the apps code, including but not limited to; the functions used, variables to be considered and the different classes to be created.

- A Final System with automated tests - This system will be the final app created for the users, it should be providing all the functionality mentioned in the Requirement Specification document. This system will also have a class containing tests which can be run at any point throughout or after the project.
- A blog of weekly progress - This is an online website that records weekly progress reports, from the start of the project to the end.
- A Test Specification document - This document will talk about the tests used for my project, keep track of the tests and their results throughout the project and provide a detailed explanation of the tests.
- Prototypes of key features - These prototypes are undeveloped key features of the app which will be used to show the progress of the development of the app and provide a basic understanding of how some features were adapted to suit the final system (useful for maintenance).
- Demonstrations - There are two demonstrations throughout this project, a mid-demonstration done in around half-way into the project and a final demonstration at the end of the project. Although no new documents will be produced it is still important to think of these deliverables throughout the project.
- A Maintenance Specification document - This document will discuss future development possiblilities and the problems and some solutions towards making changes to the app in the future.
- A Final Report document - This document summarises the project as a whole including, but not limited to; the final outcome, the progress throughout the development, difficulties and strengths throughout the project and personal thoughts and feelings on the project.

12.2 [Appendix 2] - Requirement Specification

Introduction

Purpose of this Document

This document describes the requirements for the Major Project - Money Tracker and any constraints upon the creation/usage of the project.

Scope

The Requirements Specification describes the functions needed to produce a system that enables users to access a Google Drive text file, send/get data to/from the connected Google Drive text file and allow the user to input data via a barcode scanner.

Objectives

The objectives of this document are:

- To describe the background information on the Major Project - Money Tracker.
- To provide details of the criteria that the Major Project - Money Tracker product must meet.
- To describe the types of interaction with the system that must be supported.

General Description

Product Perspective

The Major Project - Money Tracker will develope an app for android devices using Android Studio software which allows the user(s) to input and view the contents of various Google Drive text files that contain information about the different things that the user(s) have spent money on. The user(s) will be able to input data manually or the app will provide a bar code scanner, which will automatically input the data once it has scanned an appropriate bar code, and then the user has the option to edit the data and/or send it to the Google Drive text file. [Appendix 1]

Product Functions

The product will provide the following features:

- Display data from a Google Drive text file.
- Write data to a Google Drive text file
- Allow user to manually input item information.
- Allow user to scan a bar code to retrieve multiple items information.
- Display all item information inputted by the user/retrieve from a bar code scan.
- Provide an offline version of the Google Drive text file if online version is unavailable.
- Allow user to choose which Google Drive text file they would like to use.

User Characteristics

The targeted audience for this app is anyone who is interested in keeping track of what they are spending their money on. Usually this target audience would involve people who are already have good organisational skills such as accountants and/or people who are in need of keeping track of how their spending their money due to having a little amount of it or spending it unwisely, such as university students. [Appendix 1]

Specific Requirements

Functional Requirements

FR1 Home, Input and Display screens created. The app will include a Home screen, Help screen, input screen and Display Screen that are easily accessible and interchangable.

FR2 Choose Google Drive text file. The Home screen will allow the user to choose the Google Drive text file that they wish to use.

FR3 Inputting Data Manually The Input screen will allow the user to manually input all necessary items information:

- An item name - String
- An item catagory - String
- Item price - Integer

- Item payment option - Slider/selector/dropdown
- Payee name - String
- Date of purchase - Date

FR4 Barcode Scanner The Input screen will provide an option to use a barcode scanner to automatically fill in the items information.

FR5 Send data to Google Drive text file. The Input screen will have a button to send the data given to the chosen Google Drive text file when clicked.

FR6 Retrieve data from Google Drive text file The Display screen will retrieve data from the chosen Google Drive text file and display it appropriately.

FR7 Filter/Sort Data The Display screen will provide options on filter/sorting the data being displayed. These filtering/sorting will including:

- An item catagory - Dropdown
- Item price - Range of Integers
- Item payment option - Dropdown
- Payee name - Dropdown
- Date of purchase - Range of Dates

FR8 Display key help information The Help screen will provide key information on the most expected problems that the user may come across.

Optional Requirements

Below are some requirements which are not required for the app to be considered a success but rather to improve upon its current features to provide easier use and more functionality for the user, These requirements will only be attempted once the Functional Requirements above have been completed:

OR1 Change from using a Google Drive plain text document to a Google Sheets document.

OR2 Allow the user to download the current document being looked at as an Excel document.

OR3 Allow the user to use Excel documents as well as the Google documents.

Security Features

The app will make the user sign in to their Google account at least once before having any access to that users Google Drive.

The user will have full control on the databases that they share with other users and the created app will not directly share the databases with other users without the users permission.

External Requirements

ER1 The users device must have a camera to use the barcode scanner feature.

ER2 The Google Drive text file chosen by the user must be setup correctly for the app to access it correctly.

Accessability Requirements

Due to time contraints, there are no accessability requirements.

Performance Requirements

PR1 Response of program to user input. Any user input in either program should be appropriately reflected on the screen within one second, provided good internet connection and processing speed.

PR2 Target devises. The product produced should run correctly on Android devices.

Design Constraints

DC1 Use of XML and Java. The android app should be coded in XML and Java.

DC2 Reuse of existing software. Use of existing code is encouraged but any taken code has to be referenced in the Project Report.

Process Requirements

There are no process requirements for this project.

12.3 [Appendix 3] - Project Plan

Introduction

Purpose of this Document

This document describes the lifecycles an overall process for the Major Project - Money Tracker and explains the choices made in making the projects Gantts charts.

Scope

The Project Plan describes thelifecycle of the major project as well as the Gantts chart and any changes made to the Gantts chart throughout the project.

Objectives

The objectives of this document are:

- To describe the lifecycle process of the Major Project - Money Tracker.
- To explain the choice of the lifecycle chosen.
- To describe the Gantts Charts created and explain any changes made to them during development.

Project Lifecycle

This project will use a waterfall lifecycle methodology to manage the different tasks, however unlike a usual waterfall lifecycle methodology, I will also create a few prototypes of features for the mid-project deomnstration. The early part of the project will be securing clear requirements and design, setting up an appropriate timetable for the entire project and creating a few prototypes of features. The middle part of the project will involve the actual creation of the app and using a test-last development method approach to create automated tests. The last part of the project will involve maintenance and producing a final report. The progress of the project will be recorded weekly on an online blog as well as all created software and documents being stored on a GitLab repository. [Appendix 1]

Gantts Charts

First Draft Gantts Chart

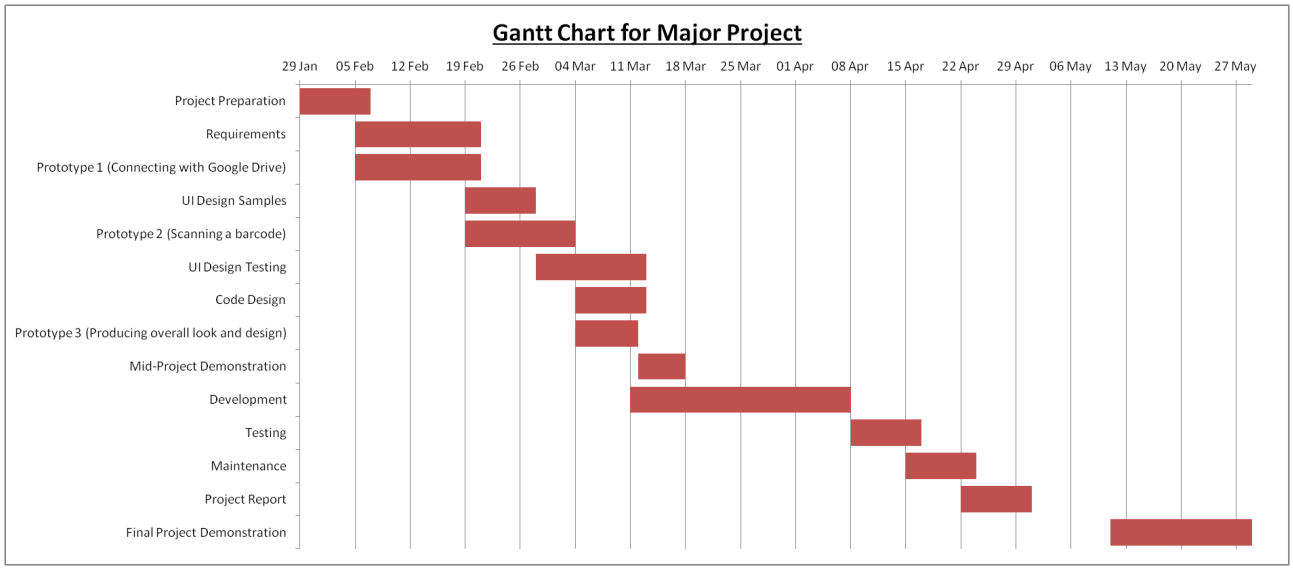


Figure 1

Figure 1 shows the first draft of the Gantts chart, which I will be using at the start of my projects lifecycle as a guide on development progress. See the Project Outline [Appendix 1] for more details on specific sections/areas.

- Documentation sections - As we can see from Figure 1, most of the documentation creation has been given just over a week to complete a first release, this is so that there is enough time to create the document as well as have a review meeting with the supervisor about possible improvements and apply those changes where necessary. On the other hand, Figure 1 shows that the Requirements Analysis is over two weeks long, this is because as I am using the Waterfall lifecycle, going back stages can be incredibly difficult so I wanted to make sure there was no uncertainty. Furthermore, the Requirement Analysis was given a longer time period also due to the fact that Prototype One would involve finding most of the uncertain requirements that may have to be altered due to over-optimistic/unrealistic requirements.

- Development section - The development was given the largest amount of time because it is the area that is estimated to take the most time and provides involves the most uncertainty.

- Prototypes sections -

  - Prototype 1 - The first prototype will involve trying to connect to the Google Drive, once connected the user will be provided an optiosn to; look at the contents of an existing Google

Drive text file, update data in an existing Google Drive text file or create a new Google Drive text file.

  - Prototype 2 - The second prototype will involve creating a bar code scanner feature into an app and display the contents of the barcode scan in a sensible way.
  - Prototype 3 - The third and final prototype will involve simply creating the appropriate UI design of the final app based off the results from the UI testing.

- Demonstration sections - The demonstrations have been give the full time period that the actual demonstration can happen, due to the fact that the precise date of the demonstration is currently unknown, however, any time available before the demonstration will be used for preparation and any time after will be spent reviewing notes from the demonstration and using them to improve the major project.

Gantts Chart Version 2

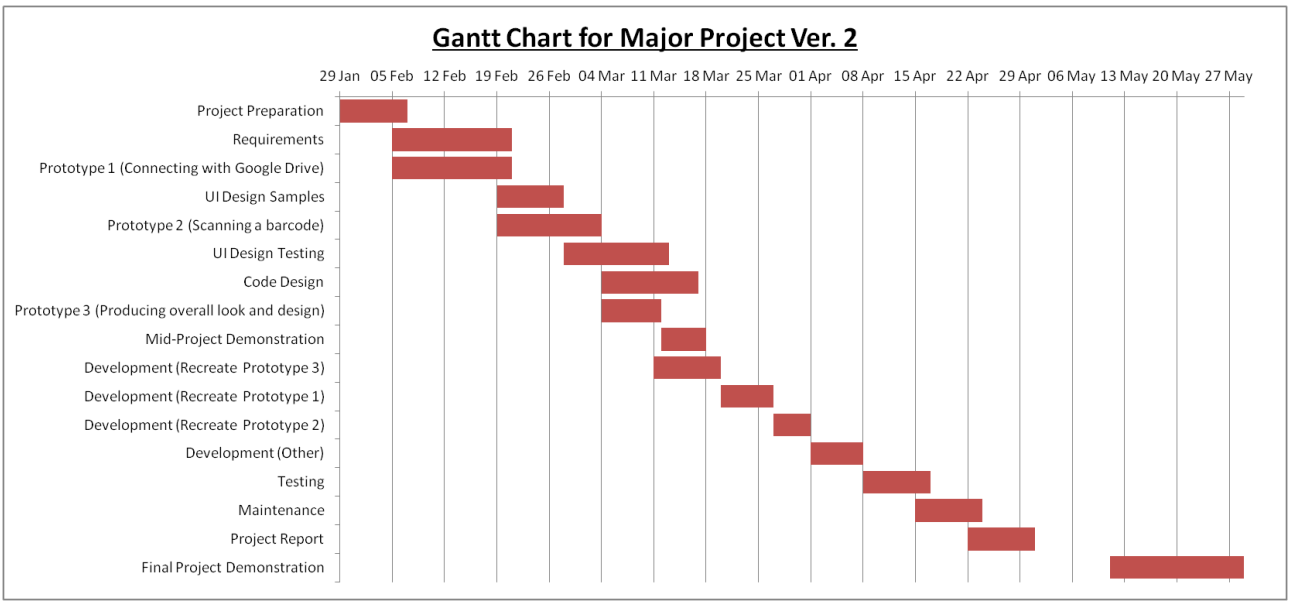


Figure 2

Figure 2 shows are more detailed version of the Gantts chart as to what has happened so far thghout the project and provides a clearer view of the stages involved in the development.

- Code Design - This section has been extended due to myself not being able to work on the project due to personal matters.
- Development (Recreate Prototype 3) - This section involves simply recreating the work done in Prototype 3 into the Final System while incorperating the response from the UI Design testers feedback.
- Development (Recreate Prototype 1) - This section involves adding the features from Prototype 1 and adjusting them so that they work well with the Final System.
- Development (Recreate Prototype 2) - This section involves adding the features from Prototype 2 and adjusting them so that they work well with the Final System.
- Development (Other) - This section is for adding features such as; filtering items, caching information when the app is closed and any other features that have not been implemented yet. This section is also for adding any new features/improving features if time is available.

12.4 [Appendix 4] - Design Specification

Introduction

Purpose of this Document

This document describes the overall structure of the final system.

Scope

The Design Spec breaks the project into separately implementable components and describes the interfaces and interaction between those components.

Objectives

The objectives of this document are:

- To describe the main components of the Final System.

UML Diagram

Figure 1 shows the UML diagram for the final system.

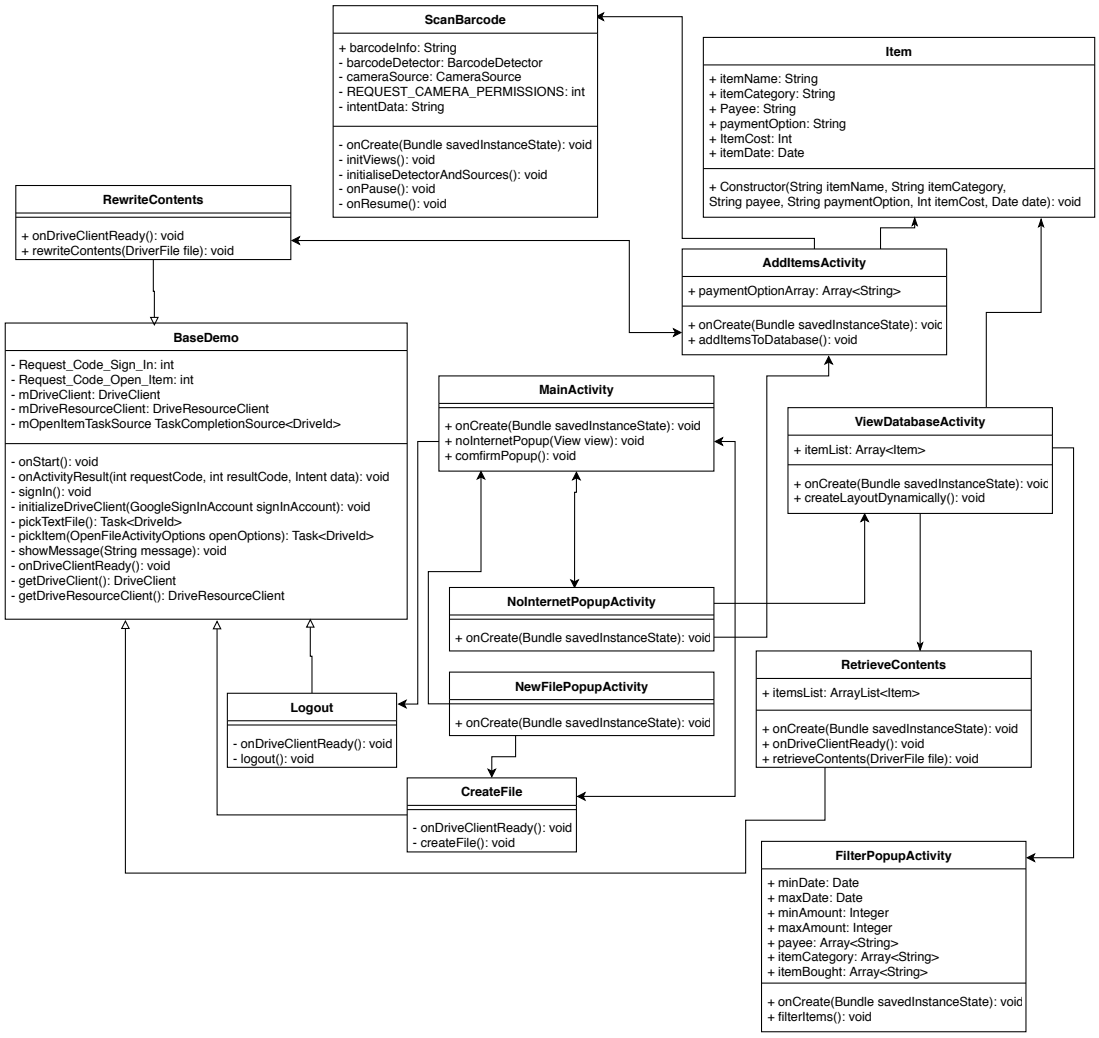


Figure 1 - UML Diagram

Inheritance

A few of the classes used will use inheritance to provide separate functionality in different classes but allow necessary functions used in all those classes to be shared between them. The following classes all extend the class BaseDemoClass so that they can properly access the users Google Drive and selected files:

- CreateFileClass
- RetrieveContentsClass
- RewriteContentsClass
- LogoutClass

Activities (Screens)

The following classes have attached an activity xml class which is used to provide an interface to the user:

- MainActivity - This activity will display the Home Screen, allowing the user to select the feature thatthey wish to use that the app provides.
- NoInternetPopupActivity - This activity provides a popup which notifies the user that they have no internet available and so some features are unavailable to them currently.
- NewFilePopupActivity - This activity provides a popup which allows the user to input a name for the file that they wish to create.
- FilterPopupActivity - This activity provides a popup which allows the user to select the filter options available that they wish to be used on the databse being used.
- ViewDatabaseActivity - This activity provides the user an interface for viewing all the items retrieved from the database being observed.
- AddItemsActivity - This activity provides the user an interface where they can input data for an item manually or use the barcode scanner and send off the inputted/retrieved data to the database currently being used.

Item class

The Item class is used to store all the information on a single item in a single object. For example, all the strings retrieved from a database file on the Google Drive for a single would be retrieved and put into an 'Item' object so that it is easily accessable later in the code.

12.5 [Appendix 5] - Testing Document

Introduction

Purpose of this Document

The purpose of this document is to provide a series of system tests that reliably tests the major functionalities of the mobile system, once a complete system as well as explain the purpose and results from the UI design testing and explain the Expresso Tests used.

Scope

It indicates multiple tests, their input, their expected outputs and pass criteria or results.

Objectives

The objectives of this document are:

- To provide detailed system tests that can be used on finished system with given pass criteria to show what is required to pass/fail each test.
- To describe and explain the the UI design testing methods used and the resilts gathered from the testing.
- To explain the scope and purpose of the Expresso tests created/used.

UI Design Testing

The UI Design Testing was done by performing user tests with a questionnaire and the Final UI Design. The Final UI Design was given out to a selection of testers along with the Final UI Design and those testers were responsible for providing feedback via answering questions on the questionnaire. The questionnaire contained the following questions;

- Scale 1-5, The app was a delight to use.
- Scale 1-5, It was easy to learn how to use the app.
- Scale 1-5, The colour scheme was appropriate to the app design.
- Short answer text, What did you dislike most about the design?
- Short answer text, What did you like most about the design?
- Long answer text, Are there any improvements you would suggest for the app design?
- Long answer text, Are there any features you would like added to improve the app?

In total only 4 people responded out of the 7 people that were sent the questionnaire. See the Questionnaire Results [Appendix 9] to view the testers feedback on the Final UI Design.

Espresso Testing

Espresso testing was considered and attempted due to the fact that the automatic testing would save a lot of time instead of some of the system tests created. However, some errors occurred when using instrumental espresso testing involving the system saying buttons where not clickable because more than 90% of the buttons view were not displayed, these errors occured I believe due to adding margins and padding to the different views. Since changing these mergins and paddings would take a lot of time and most likely change the overall design of the app, instead it was decided best to ignore from adding Espresso tests for now.

System Tests

Below are the system tests that are to be performed once a finished version of the system has been created, these tests should be preformed in order of test reference. These system tests start off assuming it's the users first time opening the app.

Table 1: System Tests.

Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-001	Check, on first start up, that the app displays the main menu appropriately.	Starting up app.	Main Menu is displayed with 3 buttons; one for allowing the user to add to a chosen database, another for allowing the user to view a chosen database and the last one for creating a new database file.	The user should be provided the option to view or add to a database of their choice or create a completely new database.
MMP-TD-002	Check we can add a name for the file we are about to create.	Click the 'Create new file' button.	A popup is displayed allowing the user to enter a name for the new file and providing a confirm button.	The user should be provided a place to enter a name for the new file to be created and a button to confim their choice.
MMP-TD-003	Check the navigation after clicking the confirm button when entering a new file name.	Enter the text "System Test" for the new file name and click the 'Confirm' button. Then sign into the desired Google account.	The activity changes to a new activity that allows the user to input item information.	The activity should change to one that allows the user to input item information appropriately.
MMP-TD-004	Check the file was created outside of the app.	Open the Google Drive.	There is a plain-text file with the name "System Test" which is empty with an appropriate created timestamp.	There should be an empty plain-text file under the name "System Test" that was created roughly when MMP-TD-003 test was done.

Continues on next page

Continuation				
Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-005	Check submitting 0 items.	Click the 'Submit' button.	The activity changes back to the main menu as displayed in MMP-TD-001 but this time the main menu also displays; the current database being used "System Test", the username of the Google Account that was used and 3 new buttons which allow the user to view or add to the current database being used or sign out of the current Google account being used.	The activity should change back to the main menu but the main menu should now also allow the user to add or view the current database being looked at or sign out of the current Google account while making it clear which buttons do what.
MMP-TD-006	Check the navigation to view the database currently being looked at and check the app can display an empty database.	Click the 'View Current Database' button.	The activity changes to an activity that displays the items in the database but no items are displayed.	The activity should change and not display any items.
MMP-TD-007	Check the navigation to add the database currently being looked at.	Click the back button and then the 'Add to Current Database' button.	The activity changes to the main menu and then an activity that allows the user to input item information just like in MMP-TD-003.	The activity should change to the main menu and then change again after the second click so that it now allows the user to input item information.

Continues on next page

Continuation				
Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-008	Check the clear buttons.	Input "Steven" into the Payee EditText, "Sandwich" into the Item Name EditText, "Lunch" into the Item Category EditText and "2.50" into the amount EditText and then proceed to click the corresponding clear buttons.	Before clicking any buttons the text was displayed as intended and after a clear buutton was clicked the corresponding EditText no longer contained text except the origional hint.	After clicking a clear button the corresponding EditText should not display any text besides the defualt hint.
MMP-TD-009	Check user can add payment options.	Click the '+' symbol button by the payment options, input "Nationwide" into the 'New Payment Option' EditText and click the 'Confirm' button.	A TextView with the word "Nationwide" appears in the list of payment options with a 'deletel' button next to it.	The activity should provide a new payment option with the name "Nationwide" shown appropriately once the 'Confirm' button has been clicked with an option to remove the payment option.
MMP-TD-010	Check user can delete payment options.	Click the "Nationwide" TextView and then click the 'Delete' button just right of it.	The "Nationwide" TextView gains a border after clicking it and the Cash TextView loses its border. After clicking the delete button the "Nationwide" Textview and its delete button disappears and the "Cash" TextView regains its border.	The "Nationwide" payment option should no longer be a possible option.

Continues on next page

Continuation				
Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-011	Check 'Date' TextView input.	Click the 'Date' TextView and then select the date "17th April 2020".	Before the 'Date' Textview is clicked it should contain the current date but after it is clicked the app should display a popup of a calander. Once the user has clicked a date in the calander popup, the popup dissapears and the selected date is shown in the 'Date' TextView.	The 'Date' TextView origionally shows the current date before any clicks. After clicking the 'Date' TextView a calender popup shold be displayed allowing the user to select any dat they want. After selecting a date the calender popup should disappear and the 'Date' TextView should display the selected date.
MMP-TD-012	Check that the user can add an item to the list to be submitted.	Input "Steven" into the Payee EditText, "Sandwich" into the Item Name EditText, "Lunch" into the Item Category EditText and "2.50" into the amount EditText and then proceed to click the 'Add' button.	Inside the ScrollView, a layout is displayed showing "Sandwich", "Lunch" and "£2.5" as well as a 'Delete' button and the EditTexts used show the hints origionally shown.	Once the user has clicked the 'Add' button, the screen should display elsewhere the words "Sandwich", "Lunch" and "£2.50" with a 'Delete' button and the origional EditTexts used should be empty besides the hint origionally shown.
MMP-TD-013	Check that the user can add a single item with a unique code to the list to be submitted by using the scan bar code feature.	Click the 'Scan Barcode' button and hover the camera over a barcode that contains the string "0000001".	Once the user clicks the 'Scan Barcode' button then the activity changes asking for the users permission to use the camera. After accepting the permission the camera remains black.	The activity should change to display what the camera sees. Once the user has hovered over the barcode with string "0000001" then the app should go back to the 'Add Items' activity and also display the words "Apple", "Fruit" and "£0.50".

Continues on next page

Continuation				
Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-014	Check that the user can add multiple items with a unique code to the list to be submitted by using the scan bar code feature.	Click the 'Scan Barcode' button and hover the camera over a barcode that contains the string "0000003".	Once the user clicks the 'Scan Barcode' button then the activity changes to display what the camera sees. After hovering over the barcode the screen then turns back to the 'Add Items' activity and also display the words "Cookie", "Snack" and "£1.5" with a 'Delete' button and also display the words "Banana", "Fruit" and "£0.7" with a 'Delete' button.	The activity should change to display what the camera sees. Once the user has hovered over the barcode with string "0000003" then the app should go back to the 'Add Items' activity and also display the words "Cookie", "Snack" and "£1.5" with a 'Delete' button and also display the words "Banana", "Fruit" and "£0.7" with a 'Delete' button.
MMP-TD-015	Check that the user can add a single item without a unique code to the list to be submitted by using the scan bar code feature.	Click the 'Scan Barcode' button and hover the camera over a barcode that contains the string "Squash\nDrink\n20/04/2020\n1\nSteven\nNationside".	Once the user clicks the 'Scan Barcode' button then the activity changes to display what the camera sees. After hovering over the barcode the screen then turns back to the 'Add Items' activity and also display the words "Squash", "Drink" and "£0.50" with a 'Delete' button.	The activity should change to display what the camera sees. Once the user has hovered over the barcode with string "Squash\nDrink\n20/04/2020\n1\nSteven\nNationside" then the app should go back to the 'Add Items' activity and also display the words "Squash", "Drink" and "£1" with a 'Delete' button.
MMP-TD-016	Check that the user can delete the last item from the list to be submitted.	Click the 'Delete' button next to the "Squash" layout.	The "Squash" layout dissapears as well as the 'Delete' button that was clicked.	The 'Squash' layout and the 'Delete' button that was clicked, should no longer be displayed.

Continues on next page

Continuation				
Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-017	Check that items have been submitted.	Click the 'Submit' button and go into your Google Drive account and check if the plain-text file under the name "System Test" has the 3 items that were submitted.	The activity goes back to the main menu and in the plain next file with the name "System Test" there are 3 items with the same information as the items that were submitted and created in the previous systems tests; MMP-TD-013 and MMP-TD-014.	The activity should go back to the main menu screen just like in MMP-TD-005. The plain-text file with the name "System Test" should contain all the information about the 3 items submitted.
MMP-TD-018	Check the app correctly gets the data stored in the database.	Click the 'View Current Database' button.	The activity changes and displays the items information that is stored in the Google Drive plain-text file being currently used.	The activity should change to display the items that are in the current database being used.
MMP-TD-019	Check the navigation for filtering.	Click the 'Filter' button.	The activity changes and displays a variety of different filtering input options.	The activity should display input options for the user to use to to add filters.
MMP-TD-020	Check the user can input a single name filter.	Input "TV" into the 'Item Name' EditText View and press enter.	The activity displays "TV" in the ScrollView next to the 'Item Name' EditText View with a 'x' delete option.	The activity should display "TV" with an option to delete the 'TV' filter appropriately.
MMP-TD-021	Check the user can input a single category filter.	Input "Electronics" into the 'Item Category' EditText View and press enter.	The activity displays "Electronics" in the ScrollView next to the 'Item Category' EditText View with a 'x' delete option.	The activity should display "Electronics" with an option to delete the 'Electroncs' filter appropriately.
MMP-TD-022	Check the user can input a single payee filter.	Input "Joe" into the 'Payee' EditText View and press enter.	The activity displays "Joe" in the ScrollView next to the 'Payee' EditText View with a 'x' delete option.	The activity should display "Joe" with an option to delete the 'Joe' filter appropriately.

Continues on next page

Continuation				
Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-023	Check the user can input a minimum date filter.	Click on the 'Min Date' TextView and select the date '20th April 2020'.	The activity displays "20/04/2020" in the 'Min Date' TextView.	The activity should display "20/04/2020" appropriately.
MMP-TD-024	Check the user can input a maximum date filter.	Click on the 'Max Date' TextView and select the date '28th April 2020'.	The activity displays "20/04/2020" in the 'Max Date' TextView.	The activity should display "20/04/2020" appropriately.
MMP-TD-025	Check the user can input a minimum amount filter.	Input "0.50" into the 'Min Amount' EditText View.	The activity displays "0.50" in the 'Min Amount' EditText View.	The activity displays "0.50" in the 'Min Amount' EditText View.
MMP-TD-026	Check the user can input a maximum amount filter.	Input "2" into the 'Max Amount' EditText View.	The activity displays "2" in the 'Max Amount' EditText View.	The activity displays "2" in the 'axn Amount' EditText View.
MMP-TD-027	Check the user can delete a single item name filter.	Click the 'TV' filter.	The 'TV' filter is removed from the ScrollView it was inside.	The 'TV' filter should no longer be displayed.
MMP-TD-028	Check the user can delete a single item category filter.	Click the 'Electronics' filter.	The 'Electronics' filter is removed from the ScrollView it was inside.	The 'Electronics' filter should no longer be displayed.
MMP-TD-029	Check the user can delete a single payee filter.	Click the 'Joe' filter.	The 'Joe' filter is removed from the ScrollView it was inside.	The 'Joe' filter should no longer be displayed.
MMP-TD-030	Check the user can delete a minimum date filter.	Click the 'Delete' button next to the 'Min Date' EditText View	The 'Min Date' EditText View displays the origional hint "Min Date".	The 'Min Date' EditText View should display the hint "Min Date".
MMP-TD-031	Check the user can delete a maximum date filter.	Click the 'Delete' button next to the 'Max Date' EditText View	The 'Max Date' EditText View displays the origional hint "Max Date".	The 'Max Date' EditText View should display the hint "Max Date".
MMP-TD-032	Check the user can delete a minimum amount filter.	Click the 'Delete' button next to the 'Min Amount' EditText View	The 'Min Amount' EditText View displays the origional hint "Min Amount".	The 'Min Amount' EditText View should display the hint "Min Amount".
MMP-TD-033	Check the user can delete a maximum amount filter.	Click the 'Delete' button next to the 'Max Amount' EditText View	The 'Max Amount' EditText View displays the origional hint "Max Amount".	The 'Max Amount' EditText View should display the hint "Max Amount".

Continues on next page

Continuation				
Test Ref	Test Content	Input	Output	Pass Criteria
MMP-TD-033	Check the user can delete a maximum amount filter.	Click the 'Delete' button next to the 'Max Amount' EditText View	The 'Max Amount' EditText View displays the origional hint "Max Amount".	The 'Max Amount' EditText View should display the hint "Max Amount".
MMP-TD-034	Check the user can input multiple name filters.	Input "TV" into the 'Item Name' EditText View and press enter and input "Coke" into the 'Item Name' EditText View and press enter.	The activity displays "TV" and "Coke" in the ScrollView next to the 'Item Name' EditText View with an 'x' delete option next to both filters.	The activity should display "TV" and "Coke" with options to delete the 'TV' and "Coke" filters individually and appropriately.
MMP-TD-035	Check the user can input multiple category filters.	Input "Electronics" into the 'Item Category' EditText View and press enter and input "Drink" into the 'Item Category' EditText View and press enter.	The activity displays "Electronics" and "Drink" in the ScrollView next to the 'Item Category' EditText View with an 'x' delete option next to both filters.	The activity should display "Electronics" and "Drink" with options to delete the 'Electronics' and "Drink" filters individually and appropriately.
MMP-TD-036	Check the user can input multiple payee filters.	Input "Joe" into the 'Payee' EditText View and press enter and input "Bob" into the 'Payee' EditText View and press enter .	The activity displays "Joe" and "Bob" in the ScrollView next to the 'Payee' EditText View with an 'x' delete option next to both filters.	The activity should display "Joe"and "Bob" with options to delete the 'Joe' or "Bob" filters individually and appropriately.
MMP-TD-037	Check the user can delete multiple item name filters.	Click the 'x' delete button next to the item name filters ScrollView.	The 'Item Name' ScrollView becomes empty.	All the name filters should no longer be displayed.
MMP-TD-038	Check the user can delete multiple item category filters.	Click the 'x' delete button next to the item category filters ScrollView.	The 'Item Category' ScrollView becomes empty.	All the item category filters should no longer be displayed.
MMP-TD-039	Check the user can delete multiple payee filters.	Click the 'x' delete button next to the payee filters ScrollView.	The 'Payee' ScrollView becomes empty.	All the payee filters should no longer be displayed.
End of Table				

12.6 [Appendix 6] - Maintenance Document

Introduction

Purpose of this document

The goal of this document is to answer all of the specific questions that installers or maintainers of the software can have.

Scope

The project maintenance manual document describes the program, its structure and algorithms used in it. It includes an information about interfaces, suggestions for improvements and things to watch when making changes. It mentions physical limitations of the program, a way of rebuilding and testing the software.

Objectives

The objectives of this document are:

- Program description.
- The main data area.
- Files.
- Interfaces.
- Suggestions for improvement.
- Things to watch for when making changes.
- Physical limitations of the program.
- Rebuilding and testing.

Project Description

The product of this project is an android app designed to allow users to keep track of their finances and share those finances with others when desired. On start-up of the app, the user has a variety of simple options that they can choose from, these include;

- Creating a new Google Drive plain-text file database to store item information.
- Add new item information to any Google Drive plain-text file database of their choice.
- View old item information from any Google Drive plain-text file database of their choice.

If the user has previously accessed a database on the app then they will also have the options to;

- Add new item information to that previously accessed database without having to search for it again in their Google Drive.
- View old item information that previously accessed database without having to search for it again in their Google Drive.

The user is able to create a Google Drive database by selecting the option and entering a name for the new database, once entered the user just needs to submit the title and the database will be created.

The user is able to add item information to a Google Drive plain-text file by either; manually inputting the necessary item information required to make the item information useful for future use or automatically inputting the necessary item information using a bar code scanner. Once inputted all the user has to do is confirm their inputs by submitting the items they have created. The necessary item information to be inputted is;

- Item Name
- Item Category
- Item Amount
- Date of Purchase
- Payment Option used
- Payee (Whoever bought the item)

The user will also be able to view the item information from a Google Drive plain-text file by selecting the option, once selected, the user can view all the key item information stored in the Google Drive database and has the option to expand the items to show all of an the expanded items information. The key information shown without expanding an item is; the item name, the item category and the item amount.

While viewing a Google Drive database, the user has the option to filter through the items displayed. To filter through the items displayed the user has to select the option and then input the filters they desire, the filters have the following limits;

- Can filter by as many item names as desired.
- Can filter by as many item categories as desired.
- Can filter by as many payees as desired.
- Can filter by only a single item amount range.
- Can filter by only a single purchase date range.
- Items are filtered by an "Any" filter feature and NOT an "All" filter feature.

Furthermore, while viewing a Google Drive database, the user has the option to delete items individually, the user can only delete an item if they are looking at an up-to-date version of the database so if they tried to delete an item while the database wasn't up-to-date they would be shown a popup that notifies them of this problem and the app will update the database they are currently looking at.

Outside of the app, the user has the ability to go into their Google Drive and share their created databases with other people, once shared the shared users can access the database using the projects app just like any other Google Drive database or they can use the information stored in the Google Drive database any other way they desire.

Internet is not required for this app as it works on an offline version when there is no internet connection, however this may cause problems in unintentional data loss when using Google Drive databases on multiple devices but a warning is conveyed to the user whenever this is a possible issue.

Main data area

There are 2 main data area's;

- fileContents - This is a String variable which can be found in the MainActivity class, this variable is used to store a single, long String that contains all the item information for every item stored in a Google Drive plain-text file. This variable is accessed in most classes to retrieve or change information on specific items when desired.
- itemList - This is an ArrayList of Item objects which can be found in the ViewDatabase class, this variable is used to store the fileContents in a more approriate and easy to access way. This variable was meant to replace the fileContents variable but there was not enough time to make all the changes.

The Item objects mentioned above in the itemList are objects which are used to store all the information for a single item, this object class is used to make accessing specific item information easier to understand and retrieve or change.

Files

The application doesn't require access to any files to be run but requires access to a Google Drive plain-text file that's in an appropriate format to use any of the important features.

This may change if the app is partnered with another company so that it uses their database for the scanning bar code feature. This is explained in the suggestions for improvements section.

Interfaces

Having no acces to the Interent does enable running the program and allows the user to use an offline version of their Google Drive, however it is important to keep in mind that the users changes in an offline version may be lost or overwritten on another device. On the other hand, the online version will be updated at an appriopriate time when the offline version regains an internet connection on that device and the user is made aware of the possible data lose when making changes to an offline version of the Google Drive databases.

Suggestions for improvement

There are many suggestions for improvements as this project aimed to focus on providing a clear path for possible improvements and show where the program should go from here. Below are the suggested improvements as well as a description as to why they should be implemented and how to go about implementing them;

- Partner with a popular shopping company - This is probably the best thing you can do to improve the program as by partnering with a popular shopping company you will then have access to their databases and therefore adapt the programs bar code scanner feature to use those databases. By using another popular shopping company's database the average user will be able to use the bar code scanner feature as intended meaning that they won't have to manually input individually items constantly and so improving the apps usability by a significant amount. To implement this change the program will have to connect to the databases, the way to connect to the databases is depended on the type of databases being used. Once connected, the program should search through the database for a unique ID that matches the unique ID found in the bar code, after the program has found the item(s) with the unique ID, then it should store the

item(s) information in a String variable in an appropriate format (format is explained in Things to watch out for when making changes section).

- Make 'All' filter option - At the moment the filter feature passes all items that pass a single filte created, this improvement suggests adding the option for the user to make the filter feature have the items pass all the filters created. This is an extremely nice feature as it allows the user to be precise with their filters allowing them to get a more accurate representation of what items they wanted to see and so improves the usability of the program. To implement this improvement, it is suggested to use a Boolean variable and set it to true by defualt at the start of checking each item and for each item make the Boolean false if it fails to match any of the filters created.

- Make the program UI more accurate to the Final UI Design deliverable - This improvement is for improving the accessibility of the app and involves adding new UI such as an infinite scroll for the payment options. Implementation of this improvement is dependent on the UI being added but for the infinite scroll it is important to note that an actual infinite scroll is impossible, however it is possible to make a ScrollView look infinite to the user by moving its contents around when they are no longer directly visible to the user. It is highly suggested that before doing this improvement, to do the improvement suggested below involving editing the Final UI Design based on feedback from user testing.

- Edit the Final UI Design based on feedback from user testing - This improvement is good for improving the overall feel of the program, by doing this improvement it will make the program much more comfortable for the user and so improving the usability of the program. There are a variety of different optionsnthat could be chosen for improving the program based on the feedback from the questionnaires collected, one such option that is suggested would be changing the red 'x' buttons to trash can icons to reduce the number of squares which was considered offputting in the feedback from the questionnaires. To change the icon from red 'x' to a trash can icon, a trash can image must be saved in the drawables file and then whereas the red 'x' image is called must be replaced with that new trash can image.

- Change from a Google Drive plain-text file database to a Google Sheet database - This is a good improvement because a Google Sheet has a lot more features which the user can use outside of the program and so indirectly improves the functionality of the program. This was attempted at the start of the project but changed due to problems with modifying a Google Sheet via the program and the time constraint of the project. To implement this improvement the program needs to change the MIME types it uses from that of a plain-text file MIME type to a Google Sheet MIME type, but to fix the errors that appeared at the start of the project more research needs to be done in regards to creating a Google Sheet Spreadsheet within the program and then sending it to replace the Google Sheet being used.

Things to watch out for when making changes

One of the major things to watch out for when making changes is how item information is retrieved and changed in the code, since the program uses the fileContents variable, mentioned in the Main Data Area section, the order of which the item information is recieved and changed matters a lot. The order in which item information is retrieved and inputted from/to a Google Drive plain-text file is as follows (with each item information separated with '\n' characters):

- Item Name
- Item Category
- Item Cost
- Item Payee

- Item Date Bought
- Item Payment Option

An example String that would be used for a single item would be (multiple items would simply be the same ordered item information added onto the end of the String);

"Coke 550ml\nFizzy Drink\n0.70\n15/04/2020\nSteven\nCash\n"

However, its also important to keep in mind that the bar code scanner feature retrieves data in a slightly different order, this was a due to a mistake being made during development and there is no reason the code can't be changed so that the orders match in the future. The order the bar code string must be is as follows (with each item information separated with '\n' characters);

- Item Name
- Item Category
- Item Date Bought
- Item Cost
- Item Payee
- Item Payment Option

An example String that would be used for a single item would be (multiple items would simply be the same ordered item information added onto the end of the String);

"Coke 550ml\nFizzy Drink\n15/04/2020\n0.70\nSteven\nCash\n"

Physical limitations of the program

The user's mobile has to be an Android device. It is advised that it has an API up to level 26. To use the bar code scanner feature, the user's mobile device has to have a camera available.

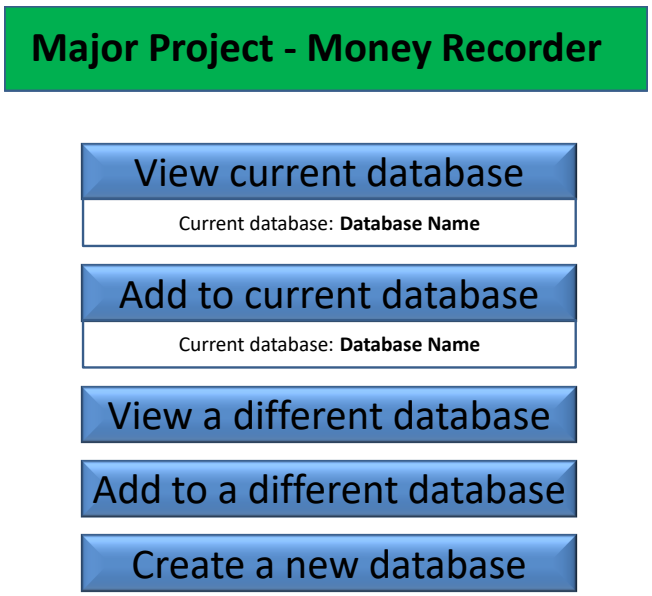
It is also important to keep in mind when adding new features or changing the UI design that one of the major benefits to the program compared to other similar money recording apps is that is has a simplistic UI design with easy to use features and accessability. Therefore, any new features or changes to UI should not disrupt the overall accessibility of the program by an excessively large amount.

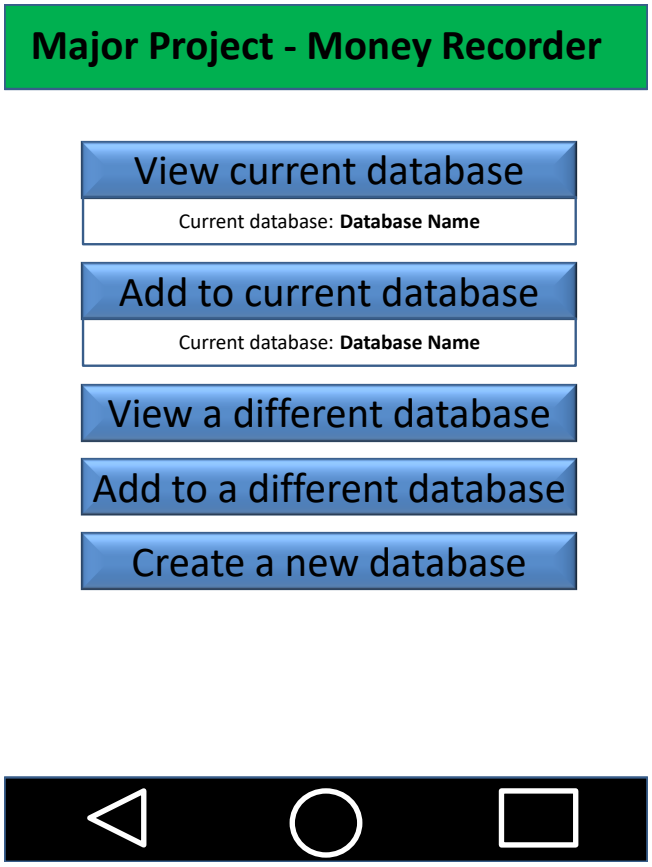
Rebuilding and Testing

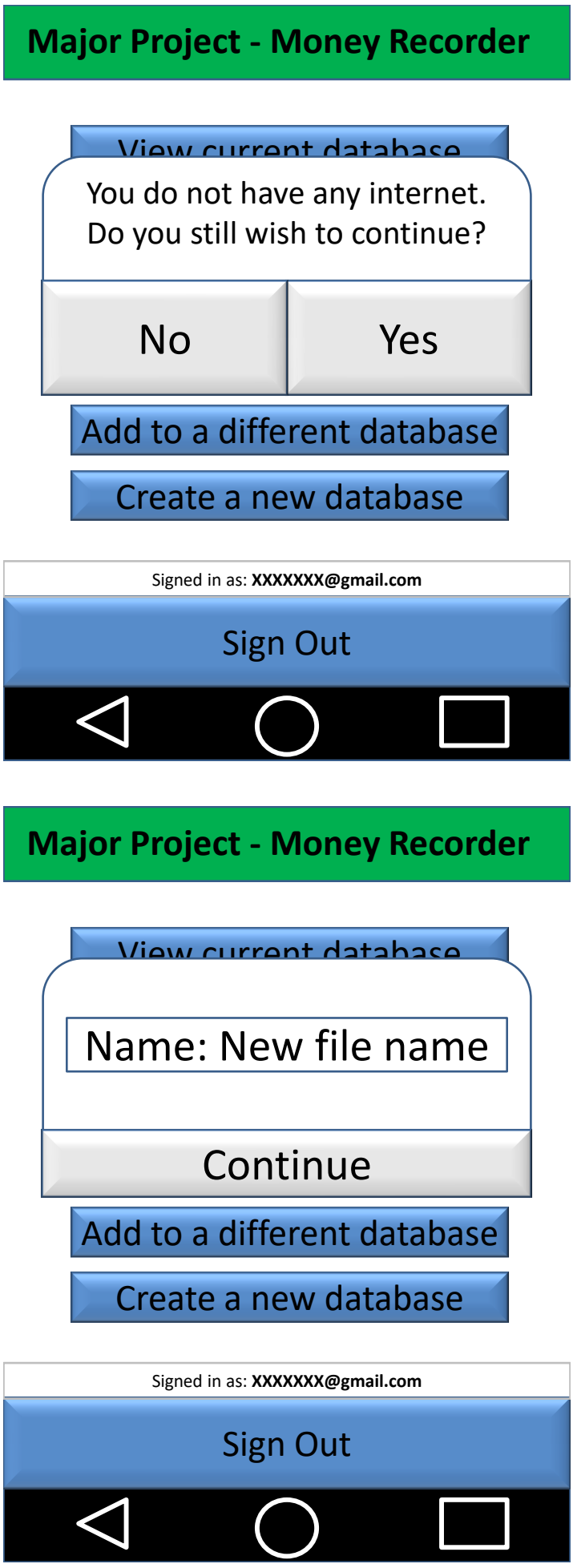
It is recommended that the System Tests in the Test Document deliverable is done every time there is a isgnificant update to the programs features to make sure non of the pre-existing features have been affected unintentionally. These system tests should be updated accordingly if the pass criteria changes with an update or new system tests should be added if new features are implemented.

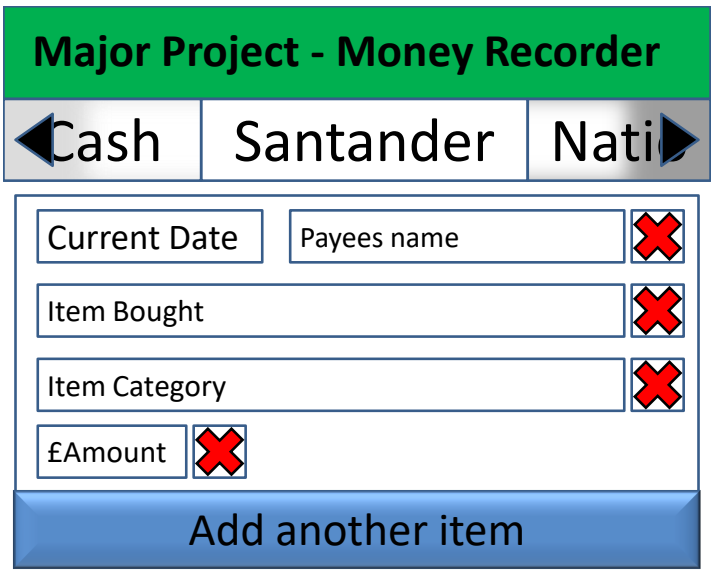
For rebuilding the program, Android Studio provides the option to rebuild the program and therefore a standard procedure for rebulding the system should be followed.

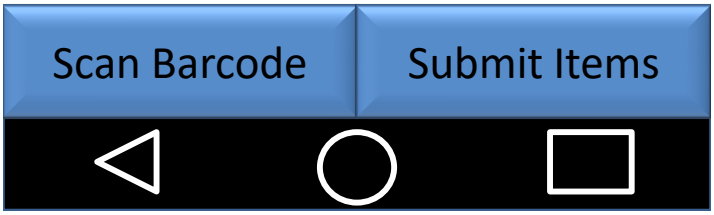
12.7 [Appendix 7] - Final UI Design

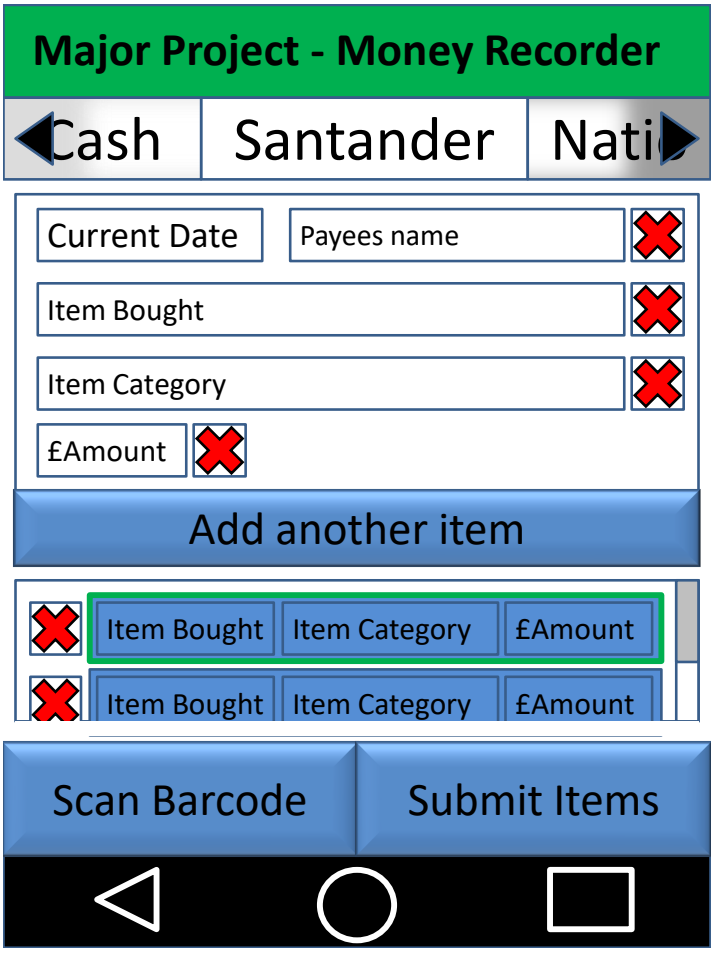


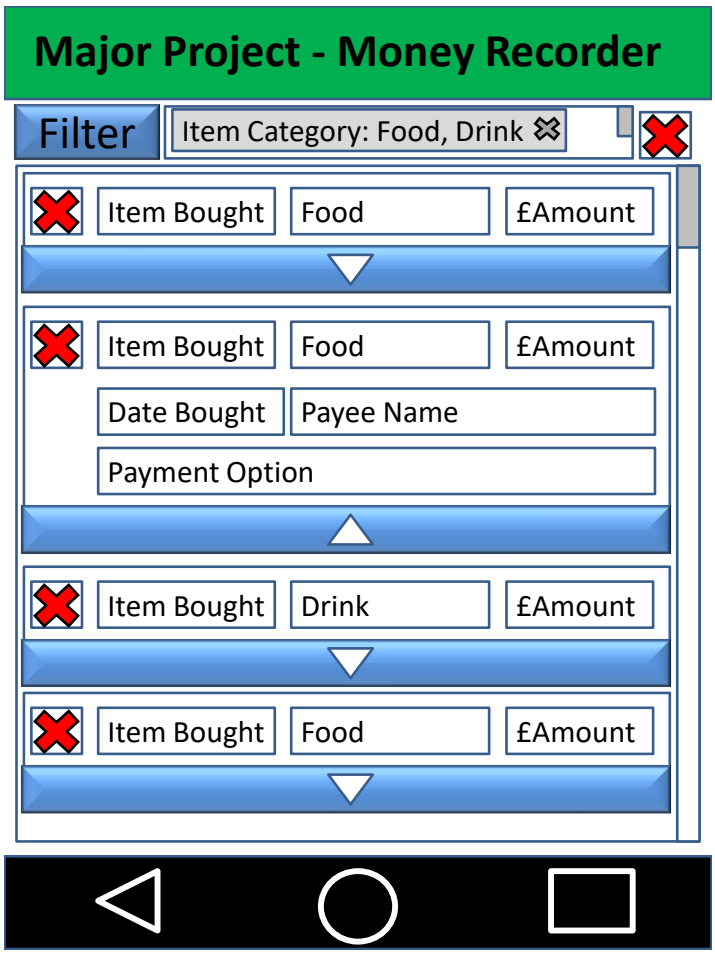



Major Project - Money Recorder
View current database
You do not have any internet.
Do you still wish to continue?
No
Yes
Add to a different database
Create a new database
Signed in as: XXXXXXX@gmail.com
Sign Out
Major Project - Money Recorder
View current database
Name: New file name
Continue
Add to a different database
Create a new database
Signed in as: XXXXXXX@gmail.com
Sign Out


Major Project - Money Recorder
Cash
Santander
Nati
Current Date
Payees name
Item Bought
Item Category
£Amount
Add another item


Scan Barcode
Submit Items


Major Project - Money Recorder
Cash
Santander
Nati
Current Date
Payees name
Item Bought
Item Category
£Amount
Add another item
Item Bought
Item Category
£Amount
Item Bought
Item Category
£Amount
Scan Barcode
Submit Items


Major Project - Money Recorder
Filter
Item Category: Food, Drink
Item Bought
Food
£Amount
Item Bought
Food
£Amount
Date Bought
Payee Name
Payment Option
Item Bought
Drink
£Amount
Item Bought
Food
£Amount


Major Project - Money Recorder
Filter
Item Category: Food, Drink
Payee
Item Bought
Item Category
Food
Drink
Between:
Min Date
Min Date
Between:
£Min Amount
£Max Amount
Confirm
Item Bought
Food
£Amount

12.8 [Appendix 8] - Ethics Form

29/02/2020

For your information, please find below a copy of your recently completed online ethics assessment

Next steps

Please refer to the email accompanying this attachment for details on the correct ethical approval route for this project. You should also review the content below for any ethical issues which have been flagged for your attention

Staff research - if you have completed this assessment for a grant application, you are not required to obtain approval until you have received confirmation that the grant has been awarded.

Please remember that collection must not commence until approval has been confirmed.

In case of any further queries, please visit www.aber.ac.uk/ethics or contact ethics@aber.ac.uk quoting reference number **15317**.

Assesment Details

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
stt31@aber.ac.uk

Full Name
Steven Twerdochlib

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department

cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
Major Project - Money Recorder

Proposed Start Date
27 January 2020

Proposed Completion Date
01 June 2020

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
Yes

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
My project is about an android app that is allowing users to connect to their google drive to save/retrieve data on items that they have bought. The user testing for this project will involve the testers to look over the user interface design and navigation and fill in a questionnaire on their opinions of it. The user interface deigns does not implement any features except navigation between screens and therefore won't be connecting to the users google drive.

I can confirm that the study does not involve vulnerable participants including participants under the age of 18, those with learning/communication or associated difficulties or those that are otherwise unable to provide informed consent?
Yes

I can confirm that the participants will not be asked to take part in the study without their consent or knowledge at the time and participants will be fully informed of the purpose of the research (including what data will be gathered and how it shall be used during and after the study). Participants will also be given time to consider whether they wish to take part in the study and be given the right to withdraw at any given time.
Yes

I can confirm that there is no risk that the nature of the research topic might lead to disclosures from the participant concerning their own involvement in illegal activities or other activities that represent a risk to themselves or others (e.g. sexual activity, drug use or professional misconduct).

Yes

I can confirm that the study will not induce stress, anxiety, lead to humiliation or cause harm or any other negative consequences beyond the risks encountered in the participant's day-to-day lives.
Yes

Please include any further relevant information for this section here:

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

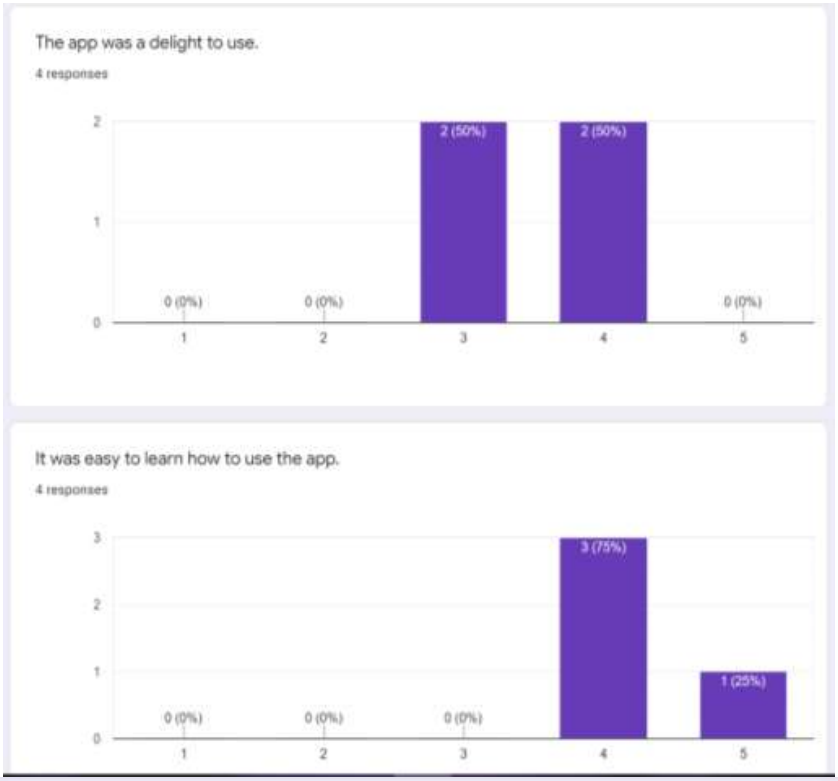
Please include any further relevant information for this section here:

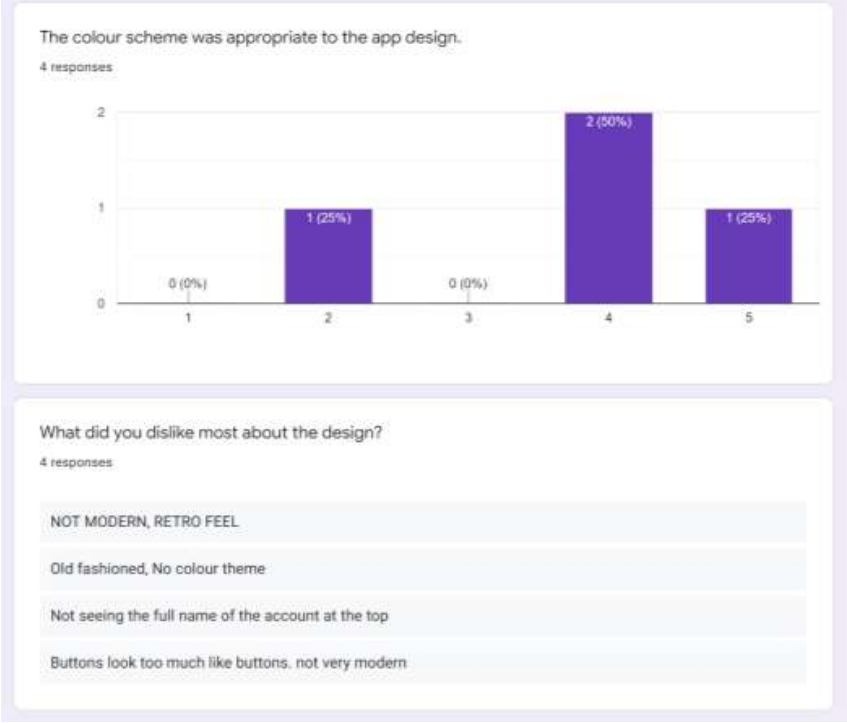
If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

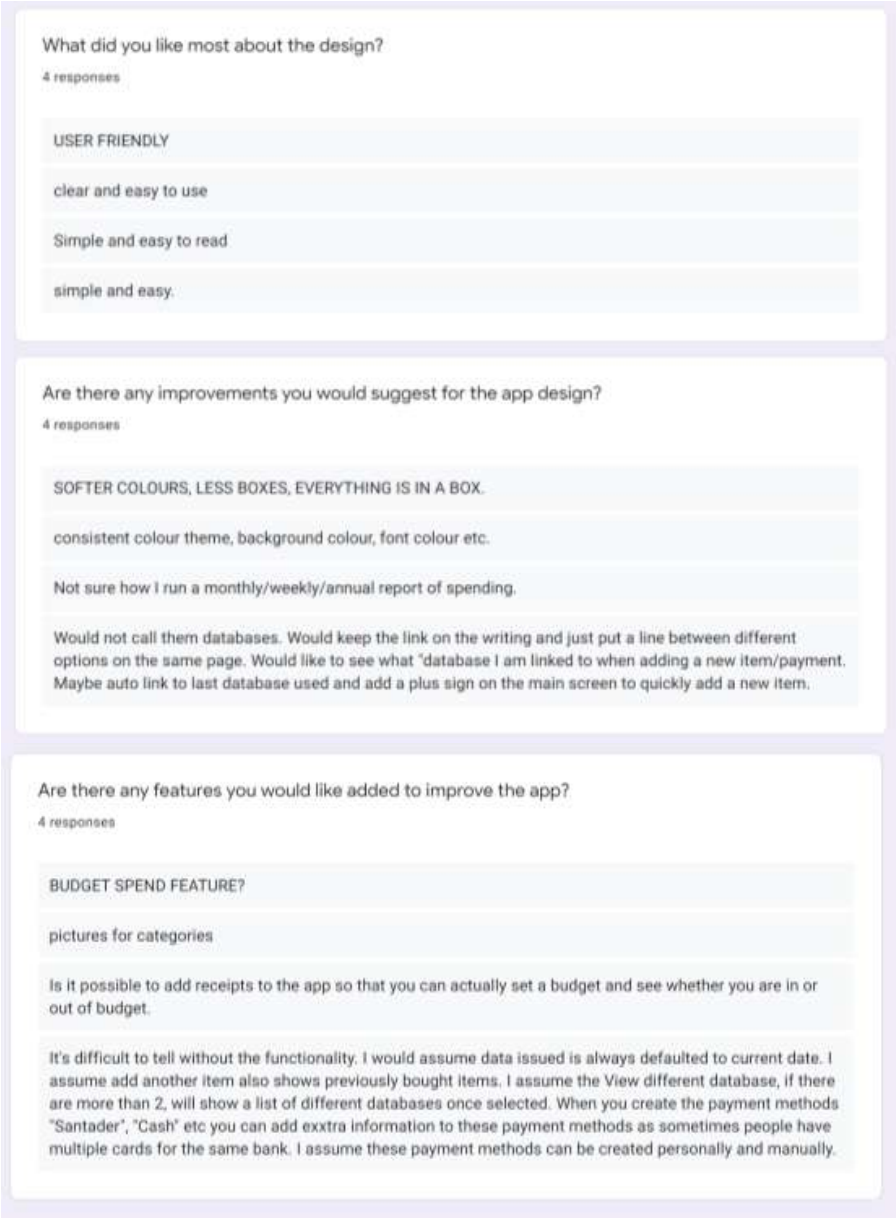
Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:

12.9 [Appendix 9] - Questionnaire Results






What did you like most about the design?
4 responses
USER FRIENDLY
clear and easy to use
Simple and easy to read
simple and easy.
Are there any improvements you would suggest for the app design?
4 responses
SOFTER COLOURS, LESS BOXES, EVERYTHING IS IN A BOX.
consistent colour theme, background colour, font colour etc.
Not sure how I run a monthly/weekly/annual report of spending.
Would not call them databases. Would keep the link on the writing and just put a line between different options on the same page. Would like to see what "database I am linked to when adding a new item/payment. Maybe auto link to last database used and add a plus sign on the main screen to quickly add a new item.
Are there any features you would like added to improve the app?
4 responses
BUDGET SPEND FEATURE?
pictures for categories
Is it possible to add receipts to the app so that you can actually set a budget and see whether you are in or out of budget.
It's difficult to tell without the functionality. I would assume data issued is always defaulted to current date. I assume add another item also shows previously bought items. I assume the View different database, if there are more than 2, will show a list of different databases once selected. When you create the payment methods "Santader", "Cash" etc you can add exxtra information to these payment methods as sometimes people have multiple cards for the same bank. I assume these payment methods can be created personally and manually.

13 References

[1] 'Android Studio' by Google and JetBrains, Last Updated January 22nd 2020, Accessed 7th February, *https://developer.android.com/studio* - This website was and will be used in determining the android studio version I will use and provide an understanding of the limitations on the development of the app, such as the sdk version.

[2] 'GitLab' [online] by GitLab Inc., Website created 2011, Last Updated January 22nd 2020, Accessed 31st January, *https://about.gitlab.com/* - This website will be used for version control, keeping records of all documents and previous versions of documents before changes were made.

[3] 'Major Project Blog' by Steven Twerdochlib, Created 1st February, Last Updated February 4th 2020, *https://stt310.wixsite.com/majorproject* - This website will be used to provide a record of the weekly progress of the project.

[4] 'Reading and creating a Spreadsheet in Android' by Pedro Carrillio, Last Updated February 27th 2018, Accessed 22nd January, *https://medium.com/@pedrocarrillo/creating-a-spreadsheet-in-android-210d68412a2e* - This website provides some insight into how to make a Google Sheet in the Google Drive of the user.

[5] 'The Massive Downside of Agile Software Development' Adam Fridman, Last Updated May 5th 2016, Accessed 5th February, *https://www.inc.com/adam-fridman/the-massive-downside-of-agile-software-development.html* - This website was used in researching the advantages and disadvantages of agile methodologies.

[6] 'Iterative Waterfall Model in SDLC', Last Updated April 13th 2020, Accessed 5th February, *https://prepinsta.com/software-engineering/iterative-waterfall-model/* - This website was used in researching the advantages and disadvantages of the iterative waterfall methodology.

[7] 'User Interface (UI) Design', Accessed 19th February, *https://www.interaction-design.org/literature/topics/ui-design* - This website was used in researching UI design and the techniques involved in creating 'good' UI.

[8] 'System Design' by Alan Faisandier and Rick Adcock, Last Updated 31st October 2019, Accessed 19th February, *https://www.sebokwiki.org/wiki/System_Design* - This website was used in researching software design and the techniques involved in creating 'good' software design.

[9] 'Consumers overspend by $7,400 a year. Here are the weekly splurges that cause the most trouble' by Sarah O'Brien, Last Updated 26th December 2019, Accessed 1st February, *https://www.cnbc.com/2019/12/26/consumers-overspend-by-7400-a-year-here-are-weekly-trouble-spots.html* - This website was used in researching the use of a money recording app.

DOCUMENT HISTORY

Version	Date	Changes made to Document	Changed by
1.0	01/05/2020	Initial creation.	Steven Twerdochlib