Mobile Applications Development Report

Roderick Ewles
40330977@napier.ac.uk
Edinburgh Napier University - Mobile Applications Development (SET08114)

1 Introduction

"Annual income twenty pounds, annual expenditure nineteen pounds nineteen and six, result happiness. Annual income twenty pounds, annual expenditure twenty pounds nought and six, result misery." Wilkins Micawber. (Dickens, 1850)

The initial idea for the app was a budgeting app, as a recent Forbes article stated that '...two thirds of the worlds population' (Mcgrath, 2015) is financially illiterate. Given this and the 'Micawber Principle' above, a budgeting app seemed like a useful app to develop. After establishing the general idea for the app I then contacted friends and colleagues to see if they would use a budgeting app if one was developed, everyone who was contacted responded positively to the idea.

The range of budgeting apps available where then analysed. After examining what was available on the apple app store and reading an article online about the Best budgeting apps available (Smith, 2018) it became apparent that all of the available apps were quite complicated. Considering the level of financial literacy in the world, it seemed strange that none of these apps catered to users who are new to budgeting. This lead to the app evolving from a generic budgeting app to one that is targeted at beginners.

The inspiration as to how the app would function came from the Paperchase Budget Planner (Thornton, 2014). This book is an extremely useful tool for those learning how to budget. It explains the basics of budgeting in simple terms and has exercises to teach simple budgeting. However, the fact it is a book makes these exercises time consuming and difficult to do while on the move. The Ideas from this book were broken down and analysed to inform the design, taking into account both the advantages and disadvantages it has. The main advantage being the explanation, which translated to the avoidance of what could be perceived as esoteric terms like balance and expenditure in favour of expressions like 'Can I afford it?' and 'Spend Money', the Idea being 'what would the user be thinking when they want to press a given button?' The other main design consideration was that the app had to be quick and easy to use on the go so that users could update their app when they spend money. In addition to this, the app would be designed to display analysis graphically as either a line graph for the budget and a bar graph for expenses. These graphs could be used to immediately inform the user as to where they might be going wrong.

As the app is designed not just to keep track of a personal budget, but also to teach the user the basic principles of budgeting, it is designed to work in tandem with a short PDF or web page explaining how the app works and giving example analysis graphs so the user knows what their personal budget analysis should look like. This would be released with the app.

2 Software Design

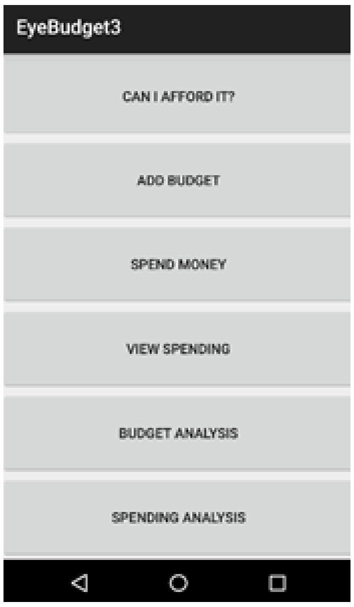
The initial design was carried out using a low fidelity approach. Each screen was mocked up on sheets of A4 paper which was then 'used' by a volunteer. This raised an issue as the initial design had no way of saying how much budget remained as the budget data was initially designed to be displayed as a list view. However after this, it was decided that such an approach displayed the balance information in a way that was both too complicated and too slow. The list was replaced with a 'can I afford it button' which would generate a toast immediately telling the user how much of their budget remained. If a user is in a shop about to spend money they need to know immediately if they can afford the item, rather than scrolling through a list of data, which they may not have time for.

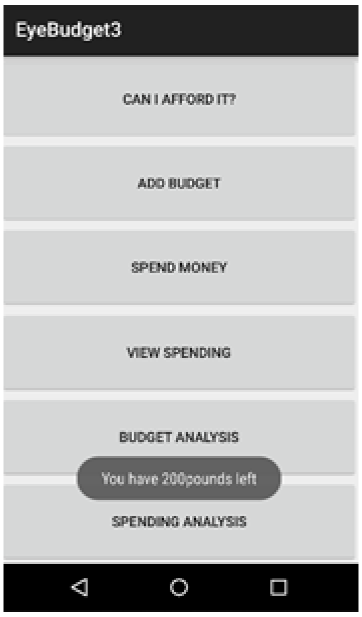
This process also highlighted several design features that relate to the app being a self teaching tool for the user as well as a budgeting app. In the initial design, the budget would be set once and automatically add next weeks budget. This would be a great function if the user was well practiced at budgeting, however, it was decided that having the user manually add their budget every week was a better way to get them to start thinking about their spending. Similarly, there were going to be 5 or 6 fixed categories of expense, but this again was dropped in favour of getting the user to manually type in the type of expense to make them think about how they are spending their budget. In addition to this all of the graphical analysis was going to be displayed in one activity, but this was decided to give too complicated a screen to be immediately appreciated by the novice user. There was also going to be a method that would warn the user when their budget was running out, but this was not implemented as the app is supposed to make the user aware of their spending and this could cause the user to rely on the app, rather than learning to become aware of their spending.

This also lead to an SQLite database being selected as the method of data persistence, as it would be the easiest way to store the budget and expense data and allow the data to be formatted for the graphs. SQL code would also allow for quick retrieval of the last recorded balance for the 'can I afford it?' function.

3 Implementation

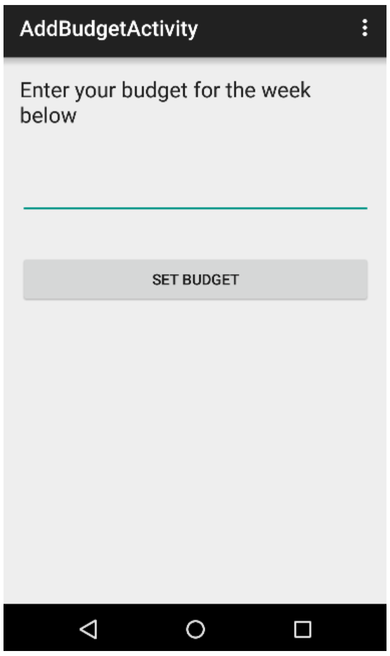
To keep each activity the user can see as simple as possible, the main activity is a 6 button menu that allows the user switch to the other activities as they use the app. It is also where the 'can I afford it' toast appears to allow the user to see their remaining budget in the quickest way possible.



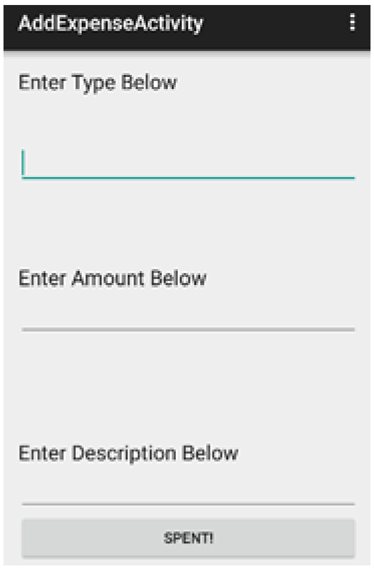


You have 200pounds left

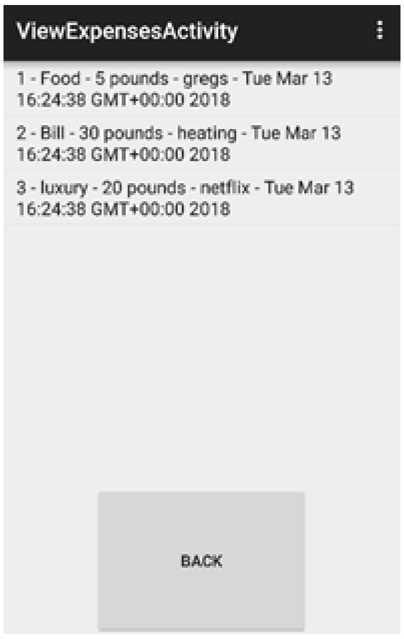
The add budget activity lets a user set a budget and add it to the data set. The add button also returns the user to the main activity.



The add expense activity has three fields for type, amount and a brief description. The add button adds the expense to the database and returns the user to the main activity.

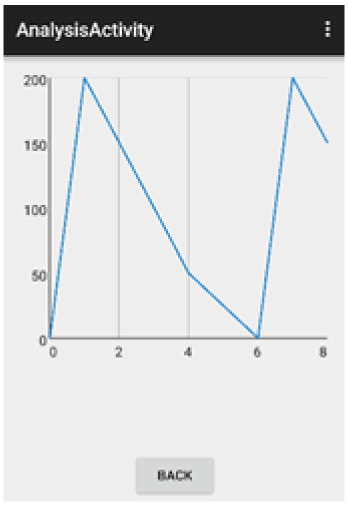


The view expenses activity displays a list of all expenses showing all their details. There is a back button to return to the main.

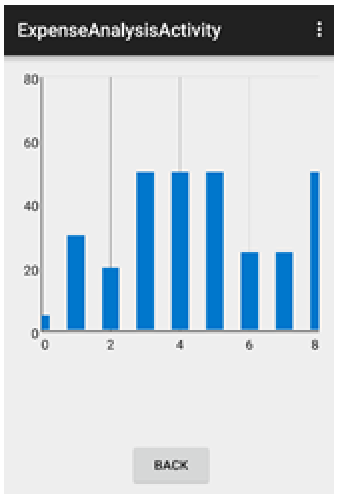


1 - Food - 5 pounds - gregs - Tue Mar 13 16:24:38 GMT+00:00 2018
2 - Bill - 30 pounds - heating - Tue Mar 13 16:24:38 GMT+00:00 2018
3 - luxury - 20 pounds - netflix - Tue Mar 13 16:24:38 GMT+00:00 2018

The analysis activity displays a line graph showing how the users balance changes as they spend money. An ideal example that would be shown as a saw tooth spiking as a new weeks balance is added and slowly falling throughout the week as the user spends money. There is also a back button to return to the main.



The expense analysis activity (above) displays a bar graph of all of the users expenses. This is supposed to act as a quick tool allowing the user to see if they spent a lot in one go. There is also a back button.



There are java classes that define the budget and expense data. These include methods to generate strings and the 'can I afford it toast'.

Finally there is a database handler class that manages the SQLite database. This includes methods for adding and retrieving data from the database as well as methods for formatting the data into data points for the graph.

4 Critical Evaluation

Compared to the original concept the design has remained fairly similar as the app still performs as a budgeting app. The main change that occurred was the change from a typical budgeting app to a beginners budgeting app. This went on to inform many of the design decisions, and changed the original activities that were planned for the app. However this was informed by modelling and speaking to potential users.

Compared to other Budgeting apps, EyeBudget is simple and offers reduced functionality. This is due to its scope of being not only a budgeting app but a self-teaching tool as well. The budget is designed to be set by the user as a weekly amount until they are ready to budget based off of their monthly salary. At this point the user may consider moving to another app that provides this functionality. The app will allow the user to input a monthly salary as a budget, however, it has no functionality for automatically adding new salary on a monthly basis. It also teaches a beginners form of budget where there is one total budget rather than separate budgets for food and luxuries etc. A user should have the data to create these budgets after using EyeBudget for a few weeks or months.

The main feedback received was that although the idea is sound, the aesthetic appearance of the app needs to be improved. This is because, due to time constraints, functionality was prioritised above the apps aesthetics. A brand 'EyeBudget' has been conceptually developed as the app is designed to help users learn to keep an eye on their personal finance as well as being a play on a leading brands product naming convention.

The first possible improvement that should be made is the improvement of the apps aesthetics, particularly if the app were to be released. Another possible improvement would be to offer a more advanced version of the app for users who have understood all that the app can teach them. This version should offer separate budget functions as well as the ability to be linked to a bank account to keep in line with competing apps, however, this functionality would take some time to develop given the amount of encryption required when dealing with users private bank details.

5 Personal Evaluation

I have learned a lot while working on this project. I didnt know any java before undertaking this module and had to teach myself over the Christmas Holidays, Fortunately I found a good beginners java book and with some time management I was able to understand the basics before the module started.

The other major challenges faced were learning things outside of the scope of the course workbook. Getting the SQLite database to work was particularly interesting as it combined what I was learning in this module with what I had learned last year in databases. I found a series of online tutorials that I worked through to understand the specifics of how SQLite works within Android Studio. I found the android developer site particularly useful while getting the database to work. As the SQLite database was core to the functionality of the app, this is the part I developed first. Following this the non-graph activities were developed. Finally the graph functionality was developed using Graph View. I found the Graph View website (Gehring, 2018) extremely useful. The site explained clearly how to add the required library and how it functions within Android Studio.

As the project I had chosen required me to learn new features while developing the app a staged approach was adopted to ensure that an individual part was working before moving on to the next. I also had to carefully manage my time given the competing deadlines I was faced with this semester.

I feel I have learned a lot and performed reasonably well throughout this project. I found it very rewarding to learn about new functionality and get that working bit by bit in my app.

References

Dickens, C., 1850. David Copperfield. London: Bradbury and Evans.

Gehring, J., 2018. Graph View. [Online] Available at: http://www.android-graphview.org/ [Accessed 2018].

Mcgrath, M., 2015. Forbes. [Online] Available at: https://www.forbes.com/sites/maggiemcgrath/2015/11/18/in-a-global-test-of-financial-literacy-the-u-s/1a75054258f0 [Accessed 2018].

Smith, J., 2018. 15 Best Budget Apps for 2018. [Online] Available at: https://www.gottabemobile.com/best-budget-apps/ [Accessed 2018].

Thornton, F., 2014. Budget Planner. 1st ed. London: Paperchase.