

Budgiecoin

SD6501 – Final Project



v3

Michael du toit

2193161

Contents

[Conceptual Framework 1](#_Toc74079514)

[Improvements, Features and Concepts applied: 1](#_Toc74079515)

[Constraints & Strategies 2](#_Toc74079516)

[Final Application Screenshots 2](#_Toc74079517)

[Instructional Material 2](#_Toc74079518)

[Summary & Recommendations 2](#_Toc74079519)

# Conceptual Framework

Final Project built upon the BudgieCoin application that I have been developing throughout Assignments 1 and 2.

For the final release of the application, the scope of Final Project was to tidy up the application and ensure features from Assignments 1 and 2 where finished, further implementing of tests for the application, develop various documentation and prepare the application for deployment onto the app store.

# Improvements, Features and Concepts applied:

One of the main goals of Final Project was to tidy up the existing features and complete implementation of any features that had not been completed yet.

The following modules saw work during the Final Project:

**Refactored Date & Time** **implementations** – In both the Database & Source Code. As was noted in the Constraint section of Assignment 2’s documentation, the original method of storing date and time had issues when it came to updating transactions. To solve this issue, I ended up re-writing sections of the Database and Source code to now handle the date-time values as long values that stored the time as seconds since the Unix Epoch. This solved the issues that were originally present, and transactions can now be updated without affecting their original time or time, their date or time can be updated properly, and they are now listed in a date and time ascending order on View Transactions. This refactoring had to be done in multiple files.

**Settings Activity** – this activity had been present but underdeveloped since Assignment 1. For the Final Project I added the ability for a user to update their username and/or pin number. This included using Android’s SharedPreferences features to record which user had logged in and then only allow them to edit their details.

**Login Activity** – Bug Fixes & minor improvements. If a user attempts to login with an empty username an error message is shown that the username input field cannot be empty. When a user logs in they are stored to the SharedPreferences as the current user.

**Testing** – I added Unit Tests for the Java Object classes and refined the Espresso UI Tests for testing the application.

**View Accounts Balances and View Transactions** – added Floating Action Buttons to these two Activities. The Floating Action Buttons open the AddAccountActivity and TransactionActivity respectively, allowing users to easily access those Activities from the related display views without having to open the Navigation Drawer.

# Constraints & Strategies

Failing Espresso Test – I had one specific Espresso Test that was failing for a period. The specific test only failed when the whole test class was being run, if the test was run individually it passed. The error message was not very helpful, and I did not find a resolution to it but eventually it stopped failing.

Emulator, Espresso Tests – had to test on device. Tedious, tests failed if screen went off.

# Final Application Screenshots

# Instructional Material

# Summary & Recommendations

## Summary

The BudgieCoin application had a lot of potential scope and features, most of which I did not get around to implementing. To give an idea of the potential scope of this type of project, it got turned down as a viable project for the Capstone course. Features that could have still been built into the application are retirement calculators, charts, full budgeting, predictive algorithms and/or bank statement uploads.

I am happy with everything I achieved in the development of the app, it is possibly not the most pretty or featureful but gave me the opportunity to learn and successfully implement Android features and concepts.

## Recommendations

If you are going to be working with date and time values with SQLite, regardless of the project, its probably best to store them as a Unix Epoch long value in the database. This makes it easier to convert to proper date-time values when called by the codebase.

Be careful with your time management and start working sooner rather than later. Implementing some of the features took longer than expected and small setbacks used ended up using more time than expected.