# Description of Improvements:

*Descriptions of the components (for the new topic) or improvements (if you’re building upon your assignment 1) that you created.*

DBHandler

ViewAccountActivity

AccountBalancesAdapter

TransactionsAdapter

A large portion of the work done for this assignment didn’t have massively notable difference to a user, other than some activities now becoming useable or interactive.

Refactored the TransactionExpenseFragment and TransactionIncomeFragment to inherit from an abstract TransactionBaseFragment due to them sharing a lot of code, with only the type of transaction being processed being the primary difference. Additionally, they reuse the same XML view since the only difference between a income and expense is managed in the Java code.

# Conceptual Framework:

*Conceptual framework outlining the scope of this assignment.*

*If it is a new project, what functions have you included to meet the learning outcomes?*

*If you choose to build upon your Assignment 1, specify the improvements on this version.*

For the 2nd release of the application, the scope of Assignment 2 was primarily the inclusion of an SQLite Database to store data, alongside it’s related CRUD functionality, and Expresso UI testing.

Alongside these two main components, the rest of the scope of Assignment 2 was to finish implementing functionality that was started in Assignment 1 that required a database to be finished, such as the View Transactions and View Accounts activities, who both take data from the database to display in custom list views.

# Features and Programming Concepts Applied:

*List of features and programming concepts applied in your design.*

I used an Object-Relational Mapper (ORM) type approach to interacting with the database.

With my ORM type approach, each table in the database have a corresponding Java classes who’s field model the table columns of the table. Using the DBHandler class implementation, data from the database was stored in the Transaction and Account class objects (from the Transactions and Accounts tables respectively). All the activities and classes then used the DBHandler class along with the class objects to work with the data.

# Constraints:

*Discussion about the constraints and strategies employed during the development.*

File Naming:

As the number of files within the project increased, it started becoming challenging to give each file a short, but still descriptive name as to what that file does. Some of the files ended up with very similar names. To resolve this issue, I had to start using longer names to describe files which made it easier again to identify the purpose of files.

# Debugging:

*Evidence of debugging performed*

# Espresso Testing:

*List of Test cases and test results (include the Espresso test result screenshots)*

# Application Screenshots:

*Screenshots of the final app components.*