

Budgiecoin

SD6501 Mobile Application Development



Michael du Toit – 2193161

Table of Contents

[Figures Table 2](#_Toc69499267)

[Executive Summary 2](#_Toc69499268)

[Application Description 2](#_Toc69499269)

[Assignment Requirements 2](#_Toc69499270)

[Application Context 3](#_Toc69499271)

[Planning & Design 3](#_Toc69499272)

[Scope of Assignment 1 3](#_Toc69499273)

[Scope of Assignment 2 4](#_Toc69499274)

[Scope of Final Project 4](#_Toc69499275)

[Resources: 4](#_Toc69499276)

[GitHub 4](#_Toc69499277)

[Classes and Methods 5](#_Toc69499278)

[Account 5](#_Toc69499279)

[Transaction 5](#_Toc69499280)

[User 6](#_Toc69499281)

[Activities and Methods 6](#_Toc69499282)

[AccountBalancesActivity 6](#_Toc69499283)

[AddAccountActivity 6](#_Toc69499284)

[AddTransactionActivity 7](#_Toc69499285)

[AppCompatNavigationDrawerActivity 7](#_Toc69499286)

[LoginActivity 8](#_Toc69499287)

[MainActivity 8](#_Toc69499288)

[SettingsActivity 9](#_Toc69499289)

[ViewTransactionsActivity 9](#_Toc69499290)

[Moving between Activities Model: 9](#_Toc69499291)

[Constraints Encountered 9](#_Toc69499292)

[Application Screenshots 10](#_Toc69499293)

[References 14](#_Toc69499294)

# Figures Table

[Figure 1 – The initial and current Class Diagram 5](#_Toc69499295)

[Figure 2 - Connections between Activities 9](#_Toc69499296)

[Figure 3 - Login Activity 10](#_Toc69499297)

[Figure 4 - Invalid Username 10](#_Toc69499298)

[Figure 5 - Invalid Pin Number 10](#_Toc69499299)

[Figure 6 - Biometric Login 10](#_Toc69499300)

[Figure 7 - MainActivity 11](#_Toc69499301)

[Figure 8 - AccountsBalanceActivity 11](#_Toc69499302)

[Figure 9 - ViewTransactionsActivity 11](#_Toc69499303)

[Figure 10 - Add Transaction 12](#_Toc69499304)

[Figure 11 - Add an Account 12](#_Toc69499305)

[Figure 12 - Settings 13](#_Toc69499306)

[Figure 13 - Navigation Drawer 13](#_Toc69499307)

# Executive Summary

BudgieCoin is a personal finance mobile application for users to help keep track of their personal income, expenses and balances of their account(s). This will enable users to learn where they are possibly spending unnecessary expenses and become more financially literate.

*Assignment 1* will cover the basic layout of the application, logging into the application and navigation between activities.

*Assignment 2* will continue to the develop the application, including addition of a database to store and persist data and adding additional features.

*Final Project* will finish development of the application, addition of final features and polishing the product.

# Application Description

The Android Mobile application idea that we are developing will be called BudgieCoin, which is a personal finance application that will enable users to keep track of their personal income, expenses and balances of their account(s).

The application will have features such as logging in and signing up, creating accounts and transactions, being able to view the transactions they made and view the current balances of their accounts.

For security, the user will need to login to the application either by using a username and pin code combination, or by using the Biometric Authentication functionality of their mobile device (if supported by their device).

# Assignment Requirements

The first assignment for the course SD6501 Mobile Application Development requires the development of a basic Android Application in Android Studio using Java and XML to show their understanding. The application developed in Assignment 1 will also form the basis to be developed further in the course’s Assignment 2 and Final Project.

For Assignment 1, the work that is handed in must include:

* Object-Oriented Programming Concepts.
* A Login Screen
* Two Android features that have not been covered in class.
* Documentation for the Application.

For Logging in, the username is ‘admin’ and the pin is ‘4321’.

# Application Context

Assignment 1 provided the opportunity to learn the specifics of developing in an Android environment. Being a more experienced programmer but not familiar with Android Studio and Java, I learnt how to transfer my existing knowledge and programming principles to Java and Android. I also have the opportunity to learn about Android features and the Android way of implementing features and functionality.

# Planning & Design

Below is the initial overview of the scope for Assignment 1, Assignment 2 and the Final Project. These overviews will very likely change as the project progresses and I learn of new potential features or methods of improving the existing features.

## Scope of Assignment 1

To fulfil the requirements of Assignment 1, the first release version of BudgieCoin will have the following deliverables:

* A Login screen that allows the user to login with either a username and pin number, or by using Android’s Biometric Authentication.
* Have two Android features that have not been covered in class (or not been covered yet). The two features that will be included in the first version of the application to accomplish this requirement is Android’s Biometric Authentication and the Navigation Drawer.
* The Activity pages for each Activity that has been identified in the “Activities and Methods” section, with a consistent look across the activities. Some of these Activity pages will be delivered with no content as part of Assignment 1 but will be populated in the following Assignments. Further Activity pages may also be added later in the project.
* The Classes outlined in the “Classes and Methods” section.
* The application clearly implements concepts of Object-Oriented Programming.
* Project documentation that includes a description of the application that we are building; the requirements and deliverables of each release (Assignment submission); a conceptual framework of the application; screenshots of the application in its current state; the classes, activities, methods and resources used in the application; and discussing the issues encountered during the development of the application.

## Scope of Assignment 2

Currently the Scope for Assignment 2 will include the following aspects, which will be reviewed and reassessed when the requirements for Assignment 2 are released.

* A Database for the application, along with the appropriate database interactions with the application, such as creating, updating, viewing and deleting transactions and accounts.
* Implement custom ListViews and Adaptors for the View Transaction and Account Activities that present the data they hold in a legible format.
* Add a Registration page which will allow a user to register to use the application.
* Implement or complete the features and methods that were scaffolded in Assignment 1 that require the database to be fully operational (these are mentioned in the “Activities and Methods” section)
* Fix any bugs that are present from Assignment 1.
* Finish any remaining work that is left over from Assignment 1.

## Scope of Final Project

The Scope for the Final Project will be assessed when the requirements for the Final Project is released. However, at this stage it will include the following aspects:

* Fix any bus that are present from the previous Assignments.
* Finish any remaining work that is left over from the previous Assignments.
* Preparing the application to be deployed via the Google Play Store.
* Setup the ability for the user to create notifications to remind them of upcoming payments.

# Resources:

Below are the resources that were needed in developing this application:

* A computer running Windows 10 with enough RAM and storage to run the required applications.
* Android Studio IDE
* A device to test the application (or emulator)
* An internet connection and web browser to be able to research features, issues, and access the Android Documentation.

Further, a database will be required later in the project to finish the project.

## GitHub

Git Version Control is used within the project to manage the history of the codebase. For Assignment 1, please use the v1.0 tag (on the master branch) to access the Assignment 1 version.

<https://github.com/MichaelDuToit/SD6501_BudgieCoin_Project/tree/v1.0>

# Classes and Methods

Diagram

Description automatically generated

Figure 1 – The initial and current Class Diagram

Figure 1 outlines the classes currently planned for BudgieCoin.

The first is the User class, which will hold the login information of the application user.

The second is the Account class which will hold information about an Account and is associated to multiple Transaction classes, who will all have an Account.

## Account

The Account class is used when creating objects that manage the accounts of the person (for example, bank accounts or investment accounts). The Account class has a few attributes which hold the details of the Account, with accessor and mutator methods for each attribute. It also has two constructors, a default constructor with no arguments and a parameterized constructor which takes all the attributes as an argument.

|  |  |
| --- | --- |
| Attributes | Methods |
| name | Account() |
|  | Account(name) |
|  | setAccountName() |
|  | getAccountName() |
|  | setAccountType() |
|  | getAccountType() |

## Transaction

The Transaction class is used for creating objects for each transaction. There will be several different attributes, each with accessor and mutator methods. There will also be two constructors for the class, a default constructor with no arguments that sets default values, and a parameterized constructor which takes all the attributes as an argument.

|  |  |
| --- | --- |
| Attributes | Methods |
| name | Transaction() |
| value | setName() |
| account | getName() |
| note | setValue() |
| date | getValue() |
|  | setAccount() |
|  | getAccount() |
|  | setNote() |
|  | getNote() |
|  | setDate() |
|  | getDate() |
|  | Income() |
|  | Expense() |

## **User**

The User class holds the information about the application User. The User class has two attributes, both of String type and with accessor and mutator methods. There is only one constructor method for the class, which is parameterized and takes both attributes as arguments. The User class will be finalized in Assignment 2.

|  |  |
| --- | --- |
| Attributes | Methods |
| username | User(username, pin) |
| pinNumber | setUsername() |
|  | getUsername() |
|  | setUserPin() |
|  | getUserPin() |

# Activities and Methods

## AccountBalancesActivity

The AccountBalancesActivity allows the user to view all their accounts on one view and shows the balance for each account. For Assignment 1, the empty activity will be present, with the list view pulling in data from the database (via an adaptor) to be implemented in Assignment 2.

## AddAccountActivity

The AddAccountActivity will be shown to the user when they want to add a new account to their accounts. On the AddAccountActivity the user will be able to specify the account’s name. Assignment 1’s release will only cover the basic layout and button interactions but will not create or store the Account. Creating and storing the Account will be finished in the following assignments when the database is implemented.

|  |  |
| --- | --- |
| **Method** | **Used for:** |
| public void createAccount() | This method will use the Account class to create an account and then eventually store the Account that has been created to the database. |
| public void cancelBtn() | This method is used to cancel the steps of creating an account and return back to the MainActivity class. |

## AddTransactionActivity

The AddTransactionActivity is show to the user when they want to add a transaction or edit an existing transaction. On the AddTransactionActivity the user will be able to specify whether the transaction is an expense, income or transfer between accounts. Assignment 1’s release will only cover the layout and some of the interaction functionality of the activity but will not create or store the transaction. This Activity and it’s features will be finished off in the following assignments when the database is implemented.

|  |  |
| --- | --- |
| **Method** | **Used for:** |
| public void showDatePickerDialog() | This method is used to create a DatePickerDialog and show it to the user to select a date for the transaction. |
| public void showTimePickerDialog() | This method is used to create a TimePickerDialog and show it to the user to select a time for the transaction. |
| public void returnHome() | This method is used to cancel the process of creating a transaction and take the user back to the MainActivity. |
| public void saveTransaction() | This method will be used to create a Transaction object and store it to the database when that functionality has been implemented. |
| public void selectAccountBtn() | This method will be used to show a floating menu to the user that has all their accounts listed and allows them to select one account to which the transaction should be associated. |

## AppCompatNavigationDrawerActivity

The AppCompatNavigationDrawerActivity class is an abstract class that extends the AppCompatActivity class and implements the functionality and connections for the Navigation Drawer. By creating a new class that implements the Navigation Drawer functionality, the Activities that require the Navigation Drawer functionality can inherit the AppCompatNavigationDrawerActivity class with the required code instead of the default AppCompatActivity class and then having to repeat (copy-paste) the same code in all the activities that need to use the Navigation Drawer.

|  |  |
| --- | --- |
| **Method** | **Used for:** |
| protected void onCreate() | This is a lifecycle that method is triggered when the activity is created and sets up and assigns values and references to the attributes. |
| protected void onPostCreate() | This is a lifecycle method which will sync the state of the navigation drawer. |
| public void onConfigurationChanged() | This method updates the toolbar toggle when the configuration has been changed. |
| public boolean onOptionsItemSelected() | This method returns a Boolean when an menu option item has been selected. |
| public boolean onNavigationItemSelected() | This method is triggered when the user clicks on one of the items in the Navigation Drawer and will then call the appropriate action. |
| public void onBackPressed() | This method is called when the back button on the device is pressed and will close the Navigation Drawer if it is open. |

## LoginActivity

The LoginActivity is the first activity shown to the user if they launch the application and have not been authenticated yet. The Login Activity will authenticate the User by either taking a username and pin or by utilizing Biometric Authentication.

For demonstration purposes, the LoginActivity is deployed with hardcoded username and pin with Assignment 1. This will be addressed in Assignment 2 with the addition of a Registration Page and storing the user credentials to the database.

|  |  |
| --- | --- |
| **Method** | **Used for:** |
| public void pinLoginBtn() | This method is called when the user clicks on the “Login with Pin” button and calls the pinLoginAuthentication method. |
| public void biometricLoginBtn() | This method is called when the user clicks the “Login with Biometrics” button and calls the biometric authentication process which has been generated in the onCreate() method. |
| public void pinLoginAuthentication() | This method checks whether the user has entered the correct username/pin combination to login, and if so calls the goToMainActivity() method. Otherwise it shows login failure messages to the user. |
| public void goToMainActivity() | This method takes the user to the MainActivity when it is called. |
| pubic Boolean onKeyUp() | This method overrides the default method so that if the user clicks on the Enter button on the number pad it calls the pinLoginAuthentication() method. |
| public void onBackPressed() | This method overrides the default method so that if the user clicks the back button on their device, the application is closed. |

## MainActivity

The MainActivity is the second activity that the user will encounter after having successfully been authenticated by the LoginActivity. The MainActivity will act as a dashboard with quick access to most of the other activities in the application.

|  |  |
| --- | --- |
| **Method** | **Used for:** |
| public void changeToTransaction() | When this method is called it takes the user to the AddTransactionActivity. |
| public void changeToViewTransaction() | When this method is called it takes the user to the ViewTransactionsActivity. |
| public void changeToAccountsView() | When this method is called is takes the user to the AccountBalancesActivity. |

## SettingsActivity

The SettingsActivity will allow the user to set their preferences for the application, including changing their Login details. With the Assignment 1 release, the Settings activity is currently just holding the default information that was provided by Android Studio when it was generated. This will be refined in Assignment 2 and Final Project with the abilities such as editing the username, pin and whether Biometric Authentication is enabled if available on the device.

## ViewTransactionsActivity

The ViewTransactionActivity lists all the transactions the user has entered into the application. The transactions are listed chronologically descending (latest activity is at the top). For Assignment 1, the empty activity will be present, with the list view pulling in data from the database (via an adaptor) to be implemented in Assignment 2.

## Moving between Activities Model:

Diagram

Description automatically generated

Figure 2 - Connections between Activities

Figure 2 demonstrates the connections between the different Activities, with LoginActivity being the entry point to the application.

Note that AppCompatNavigationDrawerActivity is an abstract class which manages the Navigation Drawer and is included in this Diagram to show which Activities can be accessed through the usage of the Navigation Drawer.

# Constraints Encountered

The Android emulator on my personal computer was very slow at times, making it painstaking to use it for testing the application. The solution to this issue was using my mobile phone in developer mode connected to my computer with a USB cable to run the application directly on the phone. This proved to work far better and faster than waiting for the emulator to run correctly.

# Application Screenshots

|  |  |  |
| --- | --- | --- |
| Graphical user interface, application, Teams  Description automatically generated  Figure 3 - Login Activity | Graphical user interface, application, Teams  Description automatically generated  Figure 4 - Invalid Username | Graphical user interface, application  Description automatically generated  Figure 5 - Invalid Pin Number |

These three figures demonstrate the Login Activity. Figure 3 shows the initial view of the Login Activity, Figure 4 and Figure 5 demonstrate the view when incorrect details are submitted.

Graphical user interface, application, Teams

Description automatically generated

Figure 6 - Biometric Login

Figure 6 demonstrates when the Login Activity is awaiting a Biometric Authentication attempt.

Graphical user interface, application

Description automatically generated

Figure 7 - MainActivity

Figure 7 shows the layout of the Main Activity view. This acts as a dashboard for the application and allows the user to quickly access functions.

|  |  |
| --- | --- |
| A picture containing shape  Description automatically generated  Figure 8 - AccountsBalanceActivity | A picture containing shape  Description automatically generated  Figure 9 - ViewTransactionsActivity |

These two figures (Figures 8 & 9) show the current scaffolding for the Accounts and Transactions views, which will be further developed when the database is implemented in Assignment 2 to provide data to two custom listviews.

Graphical user interface, application, Teams

Description automatically generated

Figure 10 - Add Transaction

Figure 10 demonstrates the Add Transaction Activity’s view. This is where a User inputs data to create a transaction.

Graphical user interface, application, Teams

Description automatically generated

Figure 11 - Add an Account

Figure 11 shows the view for the Add Account Activity, this is where the user will be able to create new accounts.

Graphical user interface, text, application

Description automatically generated

Figure 12 - Settings

Figure 12 shows the initial view for the Settings Activity. Currently it is still populated with the scaffolded settings from Android Studio, which will be replaced by BudgieCoin’s own settings later in the project as those features are added.

Graphical user interface, application

Description automatically generated

Figure 13 - Navigation Drawer

Figure 13 demonstrates the Navigation Drawer, which is in the abstract class AppCompatNavigationDrawerActivity and inherited by child Activities such as MainActivity, ViewTransactionsActivity and AccountBalancesActivity.

# References

*BiometricManager*. (2021, February 24). Retrieved April 14, 2021, from Android Developers: https://developer.android.com/reference/androidx/biometric/BiometricManager

*BiometricPrompt*. (2021, February 24). Retrieved April 14, 2021, from Android Developers: https://developer.android.com/reference/android/hardware/biometrics/BiometricPrompt

*DatePicker*. (2021, March 17). Retrieved from Android Developers: https://developer.android.com/reference/android/widget/DatePicker

*Navigation Drawer Activity in Android*. (2017, Oct 9). Retrieved from Android Mad: https://www.androidmads.info/2017/10/navigation-drawer-activity-in-android.html

*Show a biometric authentication*. (2021, February 24). Retrieved April 14, 2021, from Android Developers: https://developer.android.com/training/sign-in/biometric-auth#java

*SimpleDateFormat*. (2021, February 18). Retrieved from Android Developers: https://developer.android.com/reference/java/text/SimpleDateFormat

*TimePickerDialog*. (2021, February 24). Retrieved from Android Developers: https://developer.android.com/reference/android/app/TimePickerDialog