

# **Financial Budgeting Mobile Application**

**KA FAI CHEUNG**

**September 2017**

**Dissertation submitted in partial fulfilment for the degree of  
Master of Science in Computing for Financial Markets**

**Computing Science and Mathematics  
University of Stirling**

## **Abstract**

This project, Financial Budgeting Mobile Application, is designed to assist participants in understanding how to save money, organise income and expense records, and have a plan for the future by mobile app. With the prevalent use of mobile devices, the app can also provide a convenient way to address the need for monitoring personal financial status.

The secondary objective of this project is to investigate the effects of regular monthly investments in stock. It provides an analysis tool to assess stock returns over the past ten years and whether budgeting requirements can be met by investment.

The third objective of this project is to report the findings of usability and quality of the app. Three volunteers were involved in this study. They were satisfied with the features of the app that could address their needs for tracking income and expense records. They were dissatisfied that there was not feature to buy stock. It would be convenient if they could buy stock by the app.

Future developments are discussed.

## **Attestation**

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I declare that I carried out this master thesis independently, and only with the cited sources, literature.

**Signature**

**Date: 04 September 2017**

## **Acknowledgements**

Foremost, I would like to express my sincere gratitude to my thesis advisor, Dr. Simon B. Jones, for his support of this research, for his patience, motivation, and immense knowledge. Without his guidance and support, I would not have been able to complete this dissertation.

Last, but not least, I would like to thank my wife, Zoe, my parents, Miu Han and Lap Chi, brothers and sisters, Kin Wah, Shuk Mei, Ka Ki, and Hon Man, for supporting me throughout my life.

I would like to dedicate this dissertation to my wife and my family.

# Table of Contents

Abstract .....	i
Attestation.....	ii
Acknowledgements .....	iii
Table of Contents.....	iv
List of Figures.....	vii
List of Tables .....	viii
1 Introduction .....	1
1.1 Background.....	3
1.1.1 Literature Review .....	3
1.1.3 Calculation of Monthly and Annual Return .....	4
1.2 Google Android [5].....	5
1.2.1 The Linux Kernel .....	5
1.2.2 Java Application Programming Interface (API) Framework.....	6
1.2.3 System Apps .....	6
1.2.4 Android Applications.....	6
1.2.5 Essential Components of Android Apps.....	6
1.2.6 Activities.....	6
1.2.7 Service .....	6
1.2.8 Content Providers .....	6
1.2.9 The Activity Lifecycle.....	6
1.2.10 onCreate().....	7
1.2.11 onStart().....	7
1.2.12 onResume().....	7
1.2.13 onPause().....	7
1.2.14 onStop().....	7
2 Overall System Design.....	9
2.1 Financial Budgeting Mobile Application Requirements and Features .....	9
2.2 Use Case Diagrams.....	9
2.3 Application Architecture.....	11
2.4 Graphical User Interface Design .....	13
2.5 The Technical View of the User Interface .....	14
2.5.1 Android ListView .....	14
2.5.2 Android TabLayout.....	14
2.5.3 LineChart.....	16

2.6	Database Design .....	16
2.6.1	The structure of DailyExpense table .....	17
2.6.2	The structure of IncomeTN table.....	17
3	Hardware and Software Requirements.....	18
3.1	Hardware and Software Requirements for Financial Budgeting Mobile Application .....	18
3.1.1	Operating System Requirements .....	18
3.1.2	Software Requirements .....	18
3.1.3	Program Language Requirements .....	18
3.1.4	Hardware Requirements .....	18
4	Implementation of the Financial Budgeting Mobile App.....	19
4.1	The overview of Income Monitoring.....	19
4.1.1	incomeMainPageFragment Activity .....	19
4.1.2	Add income information to the database.....	19
4.1.3	Display the income records stored in the database.....	20
4.1.4	Calculate the accumulated income .....	21
4.1.5	Export the income records to the external storage such as SD card.....	21
4.2	The overview of Stock Return Activity .....	22
4.2.1	savingStrategyFragment .....	22
4.2.2	historicalDataConstructor.....	23
4.2.3	historicalDataAdapter.....	23
4.2.4	HistoricalDataItem .....	23
4.3	The Operation of The Stock Return.....	23
4.3.1	Saving Strategy Fragment .....	24
4.3.2	Stock Return Activity .....	25
4.3.3	HistoricalDataAdapter.....	26
5	Mobile Application Testing and Quality Evaluation .....	29
5.1	Recruitment of Volunteers.....	29
5.2	Ethical Issues .....	29
5.3	Mobile Application Test with Volunteers .....	29
5.4	Evaluation of Test Results.....	31
6	Future Development.....	33
6.1	Randomized Control Trial .....	33
6.2	Widen the User Base .....	33
6.2.1	Cross Platform Mobile Development.....	33
6.3	Internationalising and Localising the App.....	33
6.4	Purchasing Stock Feature .....	33

7 Conclusion.....	34
8 References .....	35
Appendix 1 – User guide .....	39
Appendix 2 – Installation guide.....	46
Appendix 3 – Beta Test .....	49
Appendix 4 – Usability Test .....	51

## List of Figures

Figure 1.1 Android Activity Lifecycle [14] .....	8
Figure 2.1 Use Case Diagram for Financial Budgeting Mobile Application.....	10
Figure 2.2 Architecture of the Financial Budgeting Mobile Application .....	11
<b>Figure 2.3 Income monitor, expense monitor, and stock selection operation diagram...</b>	<b>13</b>
Figure 2.4 Flow Diagram of LsitView [18] .....	14
Figure 2.5 User Interfaces of Income, Expense, and Stock Return .....	15
Figure 2.6 Historical Monthly Returns in 2016 year .....	16
<b>Figure 4.1 The result of displaying records on the user interface .....</b>	<b>20</b>
Figure 4.2 The accumulated monthly income .....	21
Figure 4.3 The result of successful data export to SD Card .....	22
Figure 4.4 The main page of the stock return .....	23
Figure 4.5 Method Flow of the AsyncTask.....	25
Figure 4.6 The flow diagram of the Stock Return Activity .....	26
<b>Figure 4.7 Three different scenarios for analysing the desired return.....</b>	<b>28</b>





## List of Tables

Table 2.1 DailyExpense Table .....	17
Table 2.2 IncomeTN Table .....	17

# 1 Introduction

## **The importance of money saving**

The goal of this project is to demonstrate the usefulness of the Financial Budgeting Mobile Application that will assist mobile users to make sound decisions on financial budgeting. The project has a number of focus areas that indicate how the Financial Budgeting Mobile Application can be helpful in saving money. First, the app will help users identify daily income and expenses as they input the income and expense records to the app. Next, the user can calculate the net income to recognise how much money is paid for fixed costs such as rent, bills, utilities, and maintenance. Third, the user may have various saving goals for the future, for instance, purchasing a house in five years time or earning a doctorate degree within three to five years. By using the app, users will understand their personal financial status and have solid information on how much money they can save for the future.

This study has three major parts: Income Monitor, Expense Monitor, and Stock Return. Moreover, the records stored in the mobile device can be exported to an SD card. The app not only calculates the stock return but also provides historical stock returns for users.

The introduction includes background on financial apps and on Google Android. It discusses the strengths and limitations of the existing budgeting applications on the market, presents financial theory for money saving plans, and introduces the structure of Google Android. Chapter 2, Overall System Design, comprises Financial Budgeting Mobile Application requirements and features, application architecture, use case diagrams, graphical user interface design, and the technical view of the user interface and database design. It explores how to design the application from the point of view of application architecture, user interface design, and database structure. Chapter 3, Hardware and Software Requirements, lists those specific requirements for the Financial Budgeting Mobile Application. Next, Implementation of the Financial Budgeting Mobile App (Chapter 4), consists of the overview of the income monitoring, “incomeMainPageFragment Activity,” the overview of stock return activity, and the operation of the stock return. It briefly outlines the operation of the income, expense, and stock return features and then explicitly explains the activities of these features from a technical point of view. The Mobile Application Testing and Quality Evaluation chapter covers recruitment of volunteers, ethical issues, mobile application testing with volunteers, and evaluation of test results. It explores the testing procedures of the application and the evaluation of the result. Finally, the Future Development chapter refers to randomized control

trials, widening the user base, internationalising and localising the app, and the purchasing stock feature. In sum, this section reviews the future development of as well as the improvements to the mobile application.

## 1.1 Background

### 1.1.1 Literature Review

As the uses of mobile and mobile applications have grown exponentially in recent years, mobile applications have become a major segment of the digital world. People tend to complete everything by mobile device [1]. Thus, companies earnestly develop their applications to reach more potential users. It is reported that more than 120 mobile money applications have been developed in expanding markets. All classes of society can benefit from a wide range of mobile applications such as mobile transactions, mobile payments, mobile banking, and mobile money [2]. Statistics also suggests that tools and finance had 99.82% and 24.99% of global android mobile users [3]. Thus, people tend to complete daily and finance tasks by mobile device. For example, “Goodbudget” is an app for budgeting. It offers a lot of budgeting tools, but no investment features are provided to save money for the future. “Investing.com” is an app for stocks. It has numerous features regarding stock investment. However, very few mobile applications have integrated the budgeting and investment features together in one application.

Regarding those people who have poor money management skills, they are likely to spend more than they earn. Therefore, they cannot control their balance of income and expenses. In order to cope with this problem, Financial Budgeting Mobile Application can help to monitor the daily income and expense. Those who want to regularly track their income and expense records will look for the application that can help them analyse their income and expense trends. The mobile application newly developed in this present study is an attempt to assist the user to better manage budgets and investment.

The advantages of currently available financial budgeting applications

1. Categorize spending so users can analyse their spending on particular categories.
2. Track income and expenses in real time.
3. Visualize income and expense trends.
4. Automatically remind users if expenses are greater than income.

The disadvantages of currently available financial budgeting applications

1. Connectivity: unstable connectivity is often found on mobile devices. This results in the unreliability of the mobile application.

2. Goal tracking: most budgeting applications mainly track income and expenses, but they do not recommend that the goal can be achieved by investing in the stocks, bonds, and funds.

### **1.1.2 Financial Budgeting Mobile Application**

With the popularity of mobile devices such as Android phones and other smart phones, mobile apps have become an effective tool for businesses to directly satisfy the needs of their potential customers. A financial budgeting mobile app not only helps users to manage their finances in an effective way but also to actively monitor their financial status on a regular basis. This application can help people who particularly want to save money or those who spend more than they earn. People can keep track of their money spending and income any time by using this app. They can read every income and expense record stored in the app, edit records, delete records, and obtain total amounts of income and expense on the app. Moreover, people who have a long-term savings plan for a goal such as an educational fund for a child or a down payment may invest on a monthly basis in order to achieve their future goal. This app can help to calculate the stock's annual return and evaluate whether it is higher or lower than the expected return.

To evaluate the return and monitor the daily income and expenses in the mobile app, financial theory and mobile app technology are applied. The following sections explain how to obtain stock returns and introduce the architecture of the Google Android operating system.

### **1.1.3 Calculation of Monthly and Annual Return**

The stock performance in the past years can be assessed by calculating the historical stock monthly and annual returns. The steps of the calculation of the monthly and annual return are described below:

Firstly the monthly return is measured by the monthly change of the stock price.

$$\text{Monthly Return} = \frac{(\text{Current Monthly average Close Prices} - \text{Last Monthly average Close Prices})}{\text{Last Monthly average Close Prices}} * 100\%$$

For example, the following calculates the Google monthly return in June.

Google's MAY Monthly Stock Prices = 939.2841

Google's JUNE Monthly Stock Prices = 953.77

$$\text{Monthly Return} = \frac{(953.77 - 939.2841)}{939.2841} * 100\%$$

$$= 1.519134\%$$

After obtaining the monthly return, the stock annualised return can be calculated by the geometric means formula, which considers the compounding in the calculation. The formula is

$$1 + R(G) = \sqrt[T]{[1 + R(1)] \times [1 + R(2)] \times \dots \times [1 + R(T)]} \quad [4]$$

Where

R(G)	R(G) means geometric mean return
From period R(1) to R(T)	the monthly return in the period from Jan to Dec.
T	the number of periods, in this case 12

The annual return is a measure of gain or loss in the stock performance, and it is compared with the expected return.

For example, the following calculates the Google annual return in 2016.

Google Monthly Return in 2016 = 4.9% , -2.2% , 3.5% , 1.3% , -3.3% , -0.8% , 2.7% , 6.9% , -0.3% , 1.9% , -2.8% , 2%

2016 Annual Return =

$$\{ [(1+4.9\%) + (1-2.2\%) + (1+3.5\%) + (1+1.3\%) + (1-3.3\%) + (1-0.8\%) + (1+2.7\%) + (1+6.9\%) + (1-0.3\%) + (1+1.9\%) + (1-2.8\%) + (1+2\%) ] ^ {1/12} \} - 1$$

$$= 0.29\%$$

## 1.2 Google Android [5]

The Android operating system developed by Google is free, based on the Linux kernel. The Android platform comes with several major components:

### 1.2.1 The Linux Kernel

The Linux Kernel is the layer that controls the communication between hardware resource allocation and app in the Android. This layer offers the low-level services namely camera, networking, memory, CPU and Bluetooth. It helps to access the hardware components in the mobile device when the app requests to use the hardware components. [6].

## **1.2.2 Java Application Programming Interface (API) Framework**

This layer provides feature-sets to the Android operating system. Different levels of the API support different features. The keys services include: “libraries, interfaces and algorithms” [7].

## **1.2.3 System Apps**

The system apps layer provides the devices to perform the basic functions such as SMS messaging, clock and contact, Internet browsing, and more [8].

## **1.2.4 Android Applications**

Android applications are written mainly by Java. Their user interface is handled by XML language. The Android application is archived in a file named APK, which is an Android package.

## **1.2.5 Essential Components of Android Apps**

Android apps consist of the user or system to access the Android app.

## **1.2.6 Activities**

Activities handle interactions with the user. Each activity has its own function. For example, a phone call app may have different activities, one of them showing the contact list, the other providing an input page to enter the phone number.

## **1.2.7 Service**

Service is an application, which requires a longer processing time in the background thread without interrupting the other activities in the main thread.

## **1.2.8 Content Providers**

The content provider processes data operations; it manages how to obtain the data stored in local database (SQLite database) or remote database.

## **1.2.9 The Activity Lifecycle**

Activity can exist in different states such as onCreate, onStart, onResume, onPause and onStop when launching the activity. Android has a number of functions to handle the state of the activity and allow the activity to know if a state has changed. The lifecycle of the activity is shown in figure 1.1.



### **1.2.10 onCreate()**

When the activity is launched by the system, the onCreate callback function is called to create the activity or other objects. The activity is in the created state [9].

### **1.2.11 onStart()**

When the onStart function is called, the activity enters into the start state. Then the activity will be popped up on the screen [10].

### **1.2.12 onResume()**

The onResume is called as user returns to the previous activity. Then, the previous activity is popped up again on the screen [11].

### **1.2.13 onPause()**

The onPause is called as the activity is required to pause and is temporarily invisible on the screen [12].

### **1.2.14 onStop()**

While the onStop is called, it makes the activity permanently invisible on the screen because the activity is replaced by other activity [13].

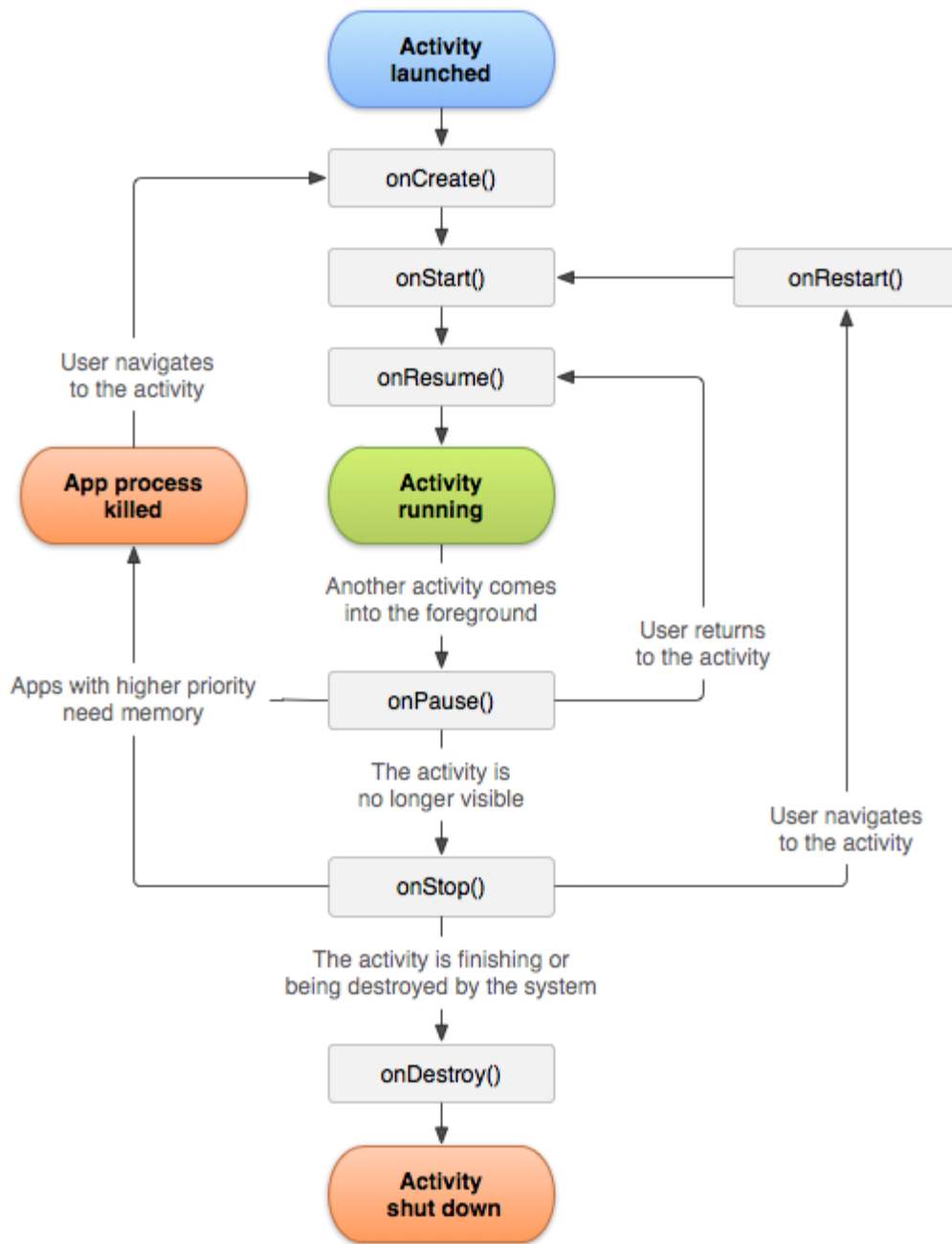


Figure 1.1 Android Activity Lifecycle [14]

## **2 Overall System Design**

This section will discuss the application architecture and design.

### **2.1 Financial Budgeting Mobile Application Requirements and Features**

The application's features and their requirements are discussed in this section. The application comprises three main features: the income monitor, expense monitor, and stock return analysis. The income and expense monitors track income and expense records. The requirements for the three features are listed below:

The requirements of the income and expense monitor features:

- The application can provide users with the capability to add income and expense records, filling out information with monthly salary, date, income from stock, other income, and remarks.
- The application can display the records stored in the database or database file directory.
- The application can calculate the accumulated records in the database.
- The records can be exported to an external SD Card.

The requirements of the stock return feature:

- Stock information can be shown such as bid, open, and last trade.
- The application is able to show the historical annual return to the users.
- The application helps to analyse the expected return.

### **2.2 Use Case Diagrams**

The architecture of the application will be discussed in the overall system design. This section focuses on the modelling of the application. In order to effectively model the application at the design level, unified modelling language (UML) diagram is employed to portray the functions of the application using its different graphical notations. UML diagram illustrates the logic connection of the application. Thus, UML facilitates the modelling and developmental process of the application, especially in the requirement specification and design level [15].

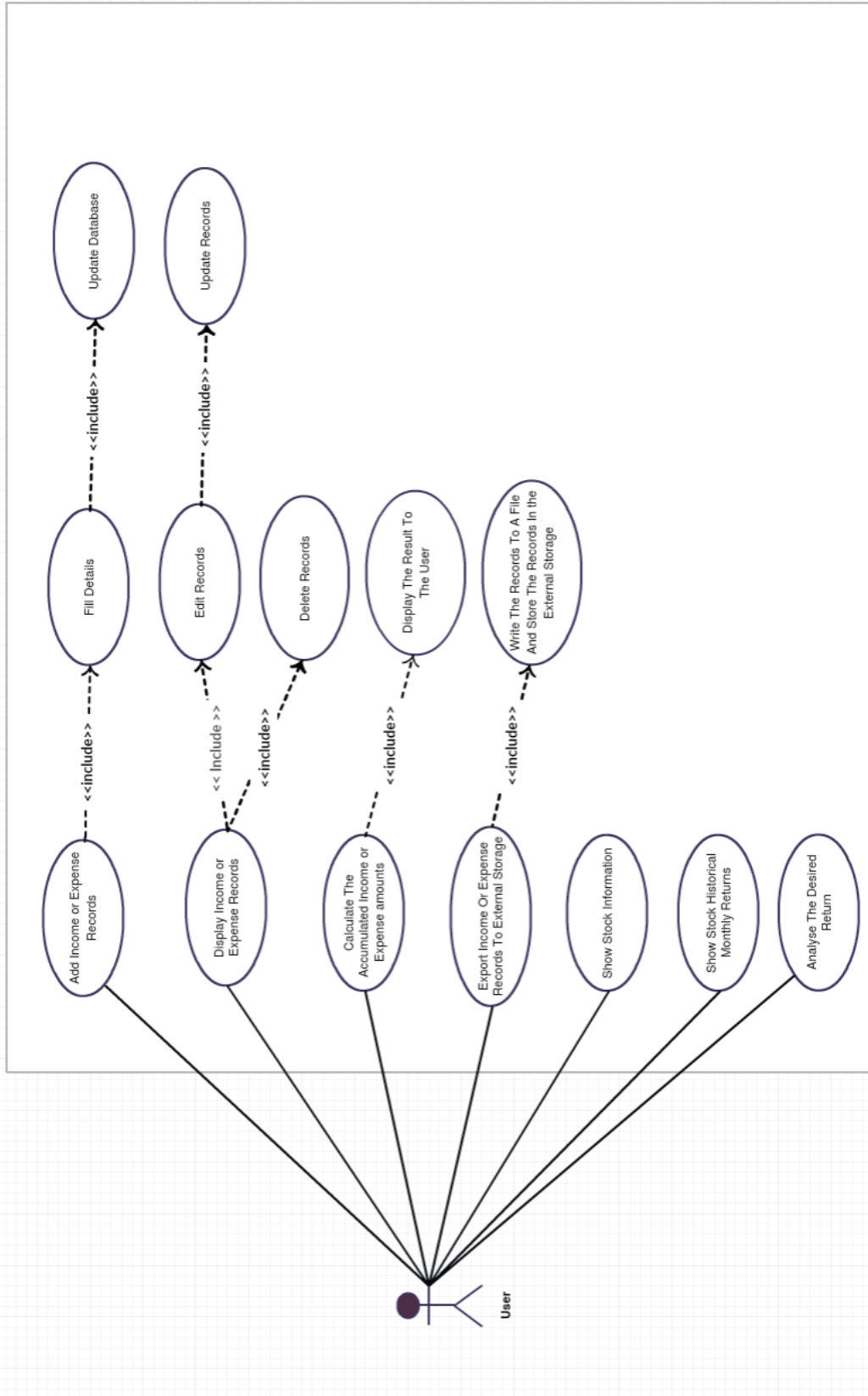
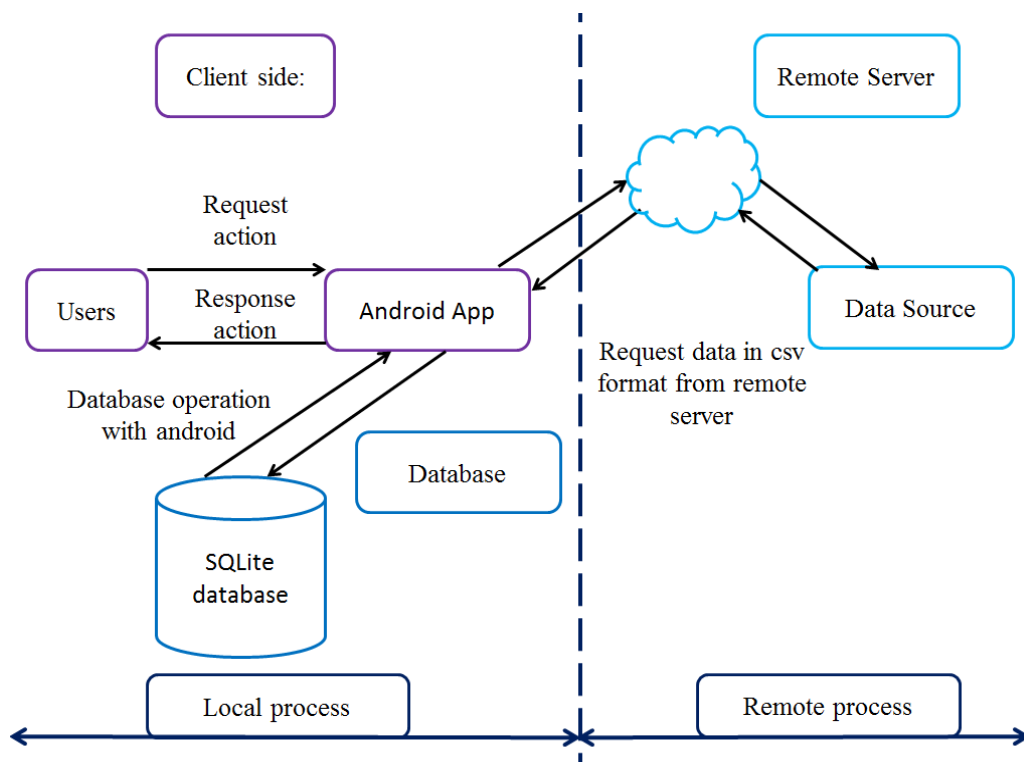


Figure 2.1 Use Case Diagram for Financial Budgeting Mobile Application

In the “use case” diagram, the user has seven different options: “add income or expense records,” “display income or expense records,” “calculate the accumulated income or expense amounts,” “export income or expense records to external storage,” “show stock information,” “show stock historical monthly returns,” and “analyse the desired return.” When the user selects “add income or expense records,” the user is requested to fill in all the details of the income or expense fields. After the user fills in the information, the database adds new records accordingly. The second case is “display income or expense records,” which presents three options: (1) fill the records stored in the database, (2) edit the selected record, and (3) delete the selected record. The database is updated correspondingly after the completion of the editing or after deleting the record. “Calculate the accumulated income or expense amounts” is the case that calculates total income or expense in the database. “Export income or expense records to external storage” exports all the income or expense records stored in the database to the external storage. “Show stock information” displays the stock information such as stock name, open, bid, or ask. “Show stock historical monthly returns” calculates the monthly and annual returns and presents the results of monthly and annual returns on the user interface. The final case is “analyse the desired return,” which helps to differentiate between desired returns and stock annual returns.

### 2.3 Application Architecture



**Figure 2.2 Architecture of the Financial Budgeting Mobile Application**

Figure 2.1 shows that the Financial Budgeting Mobile Application has two main operations. The key difference between the two operations is whether the data needs to be requested locally or remotely. The SQLite database is located within the private directory of the Android app; stock data is located on the Internet. The application accesses data from different databases depending on what activity is running. The following will examine the advantages and disadvantages of local and remote databases for income and expense data.

**The advantages of a local database for income and expense data are**

- Privacy: users are not willing to share their personal information with others.
- Control: users have complete control and the right to access and manage their data without restriction. Moreover; users are not required to enter credentials (username and password) as before accessing the remote database server.
- Security: the database server is protected by an internal network and separated from Internet access.
- Shorter data accessing time: The data accessing time from a local database is shorter than from a remote database.

**The disadvantages of local database for income and expense data are**

- Backup: the problem arises when the local storage is damaged. All of the local data are destroyed.
- Accessibility: the user cannot access the data from other devices such as a laptop or tablet.

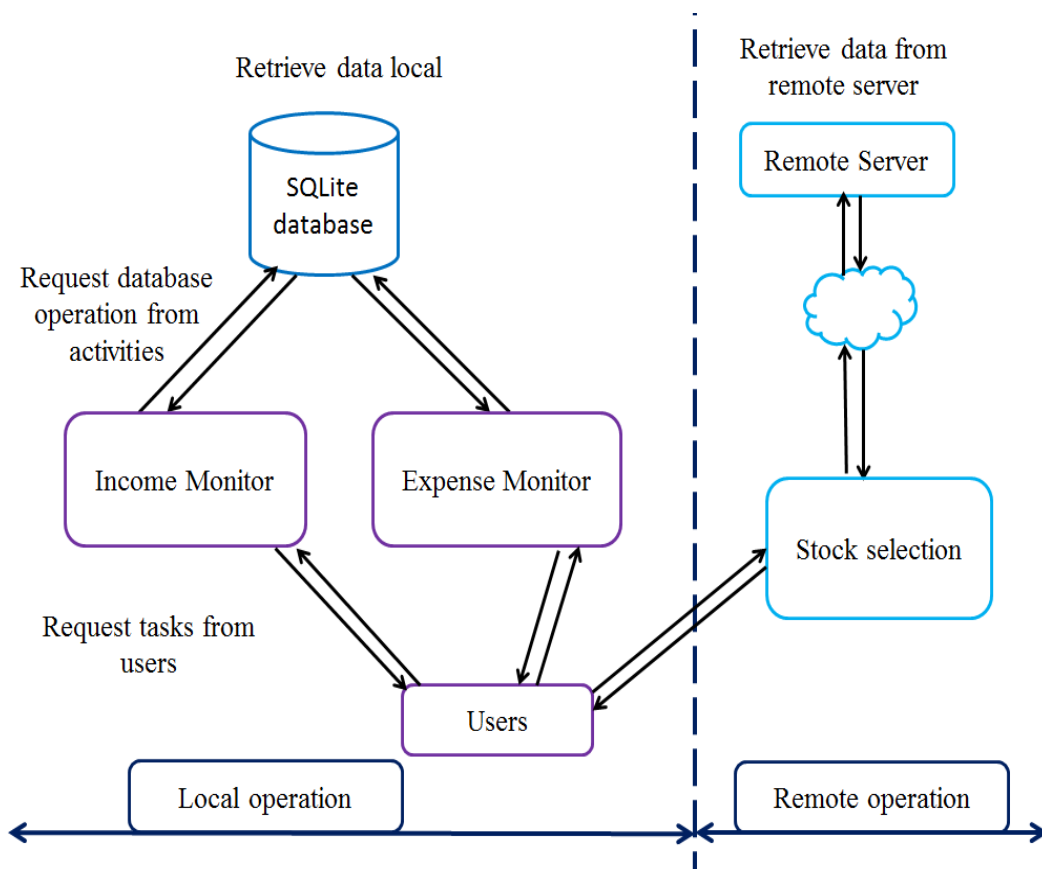
**The advantages of remote database for income and expense data are**

- Accessibility: the user can access the data from other devices via Internet.
- Flexibility: the scale of the database can be flexibly extended or reduced; the storage capacity and size of the database depends on the user demand.
- Backup: The data is stored in the virtual database server rather than a particular physical one. Therefore, the virtual database can exist in different servers. This setting allows the data to be restored faster if one of the servers is down.

**The disadvantages of remotes database for the income and expense data are**

- Security: The data traversing from mobile device to remote server cannot be guaranteed protection against hacker attacks or virus.

Figure 2.2 illustrates that the app retrieves data from the local database when users select the income monitor and expense monitor features. The app uses the SQL syntax to communicate with the SQLite database. Then the SQLite database returns data to the application based on the database query syntax. Therefore, only the income monitor and expense monitor features use the local database. On the other hand, the stock return feature of the app requests stock information from a remote database positioned on the Internet. After sending a HTTP request to the remote server, the server queries the database and obtains data. Next, the server sends back stock information to the app in a csv file. Then the app uses the stock data to calculate the returns and display the result to the user interfaces.



**Figure 2.3 Income monitor, expense monitor, and stock selection operation diagram**

## 2.4 Graphical User Interface Design

The user interface design follows the rules of User-Centered Design (UCD) [16] and Android user interface guidelines. Firstly, the design of the interface is based on the UCD, which focuses on what users want to achieve. UCD provides for the design of an app that is easy to follow and measures the usability of the app, so the app can provide a user-friendly and satisfying experience for users. Another design rule to follow is the best practices for user interface [17]. Since Android can be installed on different devices with different screen sizes, in order to overcome the various screen sizes, Google recommends using the wrap\_content

and `match_parent` attributes. They allow the app to automatically adjust the layout dimension to either fit the content size or match the size of the parent view. Therefore, the development of the user interface is much more flexible.

## 2.5 The Technical View of the User Interface

### 2.5.1 Android ListView

“ListView” is good at processing repetitive and numerous tasks [18]. Displaying the records stored in the database and the historical return calculated by stock return activity are presented in the same format and pattern. Thus, ListView is chosen to show the income, expenses, and stock return to the user. Figure 2.4 shows that ListView has three modules [19] – (1) ListView, (2) adapter, and (3) database source. ListView asks the adapter to obtain a new item and automatically insert the data to the item from the database source. Afterwards, the item is displayed on a vertical scrollable list.

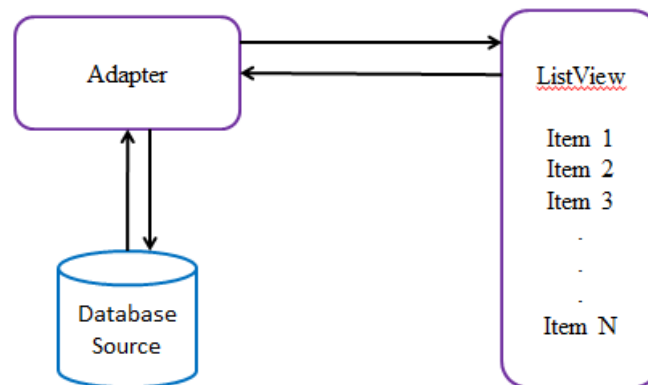


Figure 2.4 Flow Diagram of ListView [18]

### 2.5.2 Android TabLayout

The app has three main features: income, expense, and stock return. TabLayout can offer a simple user interface to manage these features. It allows all three features to appear on a single screen via different tabs [20]: “INCOME,” “EXPENSE,” and “STOCK RETURN.” Each tab represents an independent feature. For example, users select the “INCOME” tab to input new income records, the “EXPENSE” tab to input expense records, and the “STOCK RETURN” tab to analyse the stock returns. Figure 2.5 demonstrates that each tab represents the “INCOME,” “EXPENSE,” and “STOCK RETURN” interfaces.



Please add the expense information below.

Salary:

Date:

Investment :

Other Income :

Enter Remark :

Current Monthly Income :

Income user interface

Please add the expense detail information below.

Item Name:

Purchased Date:

Enter Price :

Enter Category :

Enter Remark :

Current Monthly Expense :

Expense user interface

Expected Return:

GOOG

GOOG

Stock Name:	Alphabet Inc.
Last Trade:	934.09
Previous Close:	947.80
Open:	951.78
Bid:	930.00
Ask:	931.99
Trade Time:	4:00pm
Change:	-13.71 - -1.45%
Symbol:	GOOG

Stock Return Interface

Figure 2.5 User Interfaces of Income, Expense, and Stock Return

### 2.5.3 LineChart

MPAndroidChart is a third party library developed by Philipp Jahoda [21]. It provides various graph types to display information in graphical view. The line chart is chosen to display the historical monthly returns of stock. Figure 2.6 exhibits a line graph of the “Google monthly returns” in 2016.

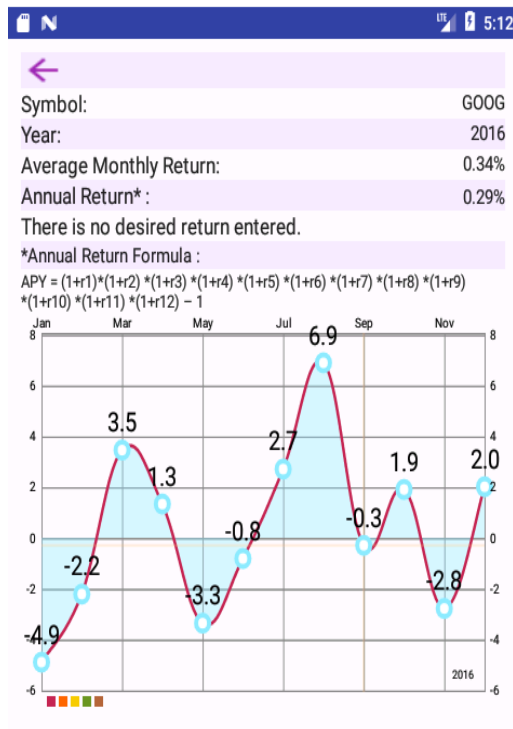


Figure 2.6 Historical Monthly Returns in 2016 year

## 2.6 Database Design

SQLite exists in the form of a library in the Android. The use of SQLite requires simply importing it from the SQLite library to the application, and thus database installation is not needed. SQLite directly links to the application. Android SQLite database classes provide a set of methods that help users manage database operations. The key methods of SQLite database classes assist users to create tables and delete, edit, and insert data into the application [22].

There are two databases named “DailyExpenseDB” and “IncomeDB” in the app. These databases are independent of each other. “DailyExpenseDB” has one table called “DailyExpense” with six attributes, namely, groceries\_Id, item, prices, category, purchaseDate, and Remarks. The DailyExpense table is shown below.

### 2.6.1 The structure of DailyExpense table

**Table 2.1** DailyExpense Table

Primary Key	String	String	String	String	String
groceries_Id	Item	Prices	Category	purchaseDate	Remarks

The following example shows how to insert a new record in the IncomeTN table.

1. The new record is Apples, 100 pounds, category is food, the input date is 24-07-2017, and remarks contain “I brought apples for 100 pounds on July 24.”
2. After filling out all the attributes, the primaryKey is automatically incremented by 1.

The table is updated accordingly as below:

Primary Key	String	String	String	String	String
groceries_Id	Item	Prices	Category	purchaseDate	Remarks
1	Apples	100	Food	24-07-2017	I brought apples for 100 pounds.”

### 2.6.2 The structure of IncomeTN table

“IncomeDB” has one table called “IncomeTN” with six attributes. They are income\_Id, salary, investment, otherIncome, Date, and Remarks. The IncomeTN table is shown below.

**Table 2.2** IncomeTN Table

Primary Key	String	String	String	String	String
income_Id	Salary	Investment	otherIncome	Date	Remarks

The following example shows how to insert a new record in the IncomeTN table.

1. The new record is 3000 salary, 300 earned from investment, otherincome is 200, the input date is 25-07-2017 and remarks contain “My July salary.”
2. After filling out all the attributes, the primaryKey is automatically incremented by 1.

The table is updated accordingly as below:

Primary Key	String	String	String	String	String
income_Id	salary	Investment	otherIncome	Date	Remarks
1	3000	300	200	25-07-2017	My July salary.

## **3 Hardware and Software Requirements**

### **3.1 Hardware and Software Requirements for Financial Budgeting Mobile Application**

#### **3.1.1 Operating System Requirements**

- Windows 10 (32 bit or 64 bit) [23]

#### **3.1.2 Software Requirements**

- Android version 3.2 or higher [24]
- Minimum SDK Version API Level: 13 or higher [25]
- Android Studio IDE v2.3.0 [26]
- Java SE Development Kit [27]
- USB Driver for Windows

#### **3.1.3 Program Language Requirements**

- XML [28]
- JAVA [29]
- SQL [30]

#### **3.1.4 Hardware Requirements**

- Android Mobile or Android Emulator
- Computer

## **4 Implementation of the Financial Budgeting Mobile App**

The Financial Budgeting Mobile Application is implemented with SQL, Java, and XML languages and various libraries in Android. XML is used to implement the user interface, SQL syntax with SQLite database classes is used for information storage, and Java is the language used in the majority of the this application. As mentioned in previous chapter, the app has three main features to help users monitor the status of income and expenses and analyse the stock return.

### **4.1 The overview of Income Monitoring**

The income monitoring feature is mainly written in the three languages: SQL, Java, and XML. This feature has seven activities: `incomeMainPageFragment`, `DisplayIncomeActivity`, `EditIncomeRecordActivity`, `IncomeConstructor`, `IncomeDBHelper`, `IncomeListAdapter`, and `ShowIncomeSingleRecordActivity`.

#### **4.1.1 `incomeMainPageFragment` Activity**

The `incomeMainPageFragment` activity is the gateway to the income feature. It provides users several services: (1) adding income information to the database, (2) displaying the income records stored on the database, (3) calculating the accumulated income, and (4) exporting the income records to external storage such as an SD card.

#### **4.1.2 Add income information to the database**

After users input the income data in the `EditText` fields, users press the “add income records” button. Then the program checks if the salary, date, and investment fields are empty. If one of the fields is empty, the program pops up a toast box to inform users: "Please enter the Salary, Purchased Date, Investment." The data are only stored to the database if those fields are filled. The following codes are to verify that the salary, date, and investment fields are not empty.

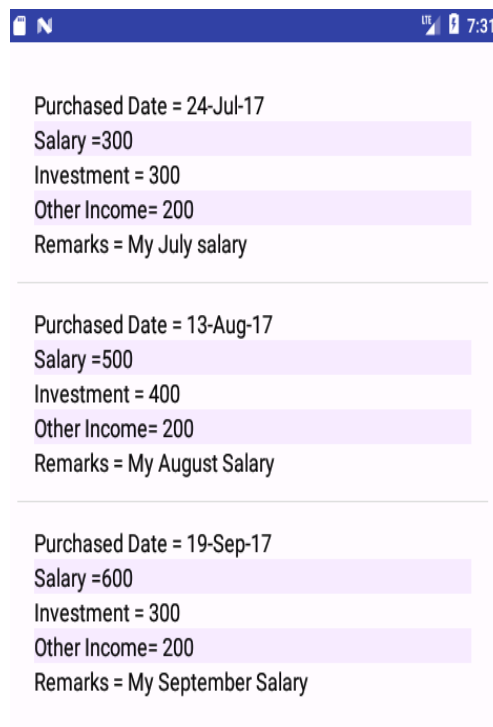
```

if (TextUtils.isEmpty(EDSalary) || TextUtils.isEmpty(EDPurchasedDate) || TextUtils.isEmpty(EDInvestment)) {
    Toast.makeText(getActivity(), "Please enter the Salary, Purchased Date, Investment.",
Toast.LENGTH_LONG).show();
} else {
    Toast.makeText(getActivity(), "inserting record", Toast.LENGTH_LONG).show();
    DataBaseBuild();
    DBHelper.onCreate(dailyExpenseDB);
    DBHelper.AddIncomeRecord(DailyIncomeRecord);
    clearInputField();
}

```

### 4.1.3 Display the income records stored in the database

This simply informs the app to change the current activity to “DisplayIncomeActivity”. Next, DisplayIncomeActivity retrieves the records from the database and displays the records on the user interface. Figure 4.1 shows the result of displaying records on the user interface.



**Figure 4.1** The result of displaying records on the user interface

#### 4.1.4 Calculate the accumulated income

The accumulated income is equal to the sum of the investment and other income. In order to obtain the accumulated income, most of SQL syntax and objects are used to retrieve the data of investment and other income from the database. First it queries the table with "SELECT \* FROM "+ IncomeDBHelper.TABLE\_NAME." Next, the cursor moves to the first row and searches data in the investment and other income attributes. After reading through every attribute, the cursor moves to the next row and repeats the same procedures until it is at the last row in the database. The total accumulated income then is returned from the calculateMonthAmount() method. Figure 4.2 demonstrates the result of accumulated income.

The screenshot shows the 'Smart Saving Strategy' app with the 'EXPENSE' tab selected. The interface prompts the user to 'Please add the expense information below.' and includes the following fields and buttons:

- Monthly Salary:  Please enter salary.
- Date:  Date
- 
- Investment :  Please enter Investment
- Other Income :  Enter other income
- Enter Remark :  Enter Remark
- Accumulated Monthly Income : **1600.00**
- 
- 
- 
- 

Figure 4.2 The accumulated monthly income

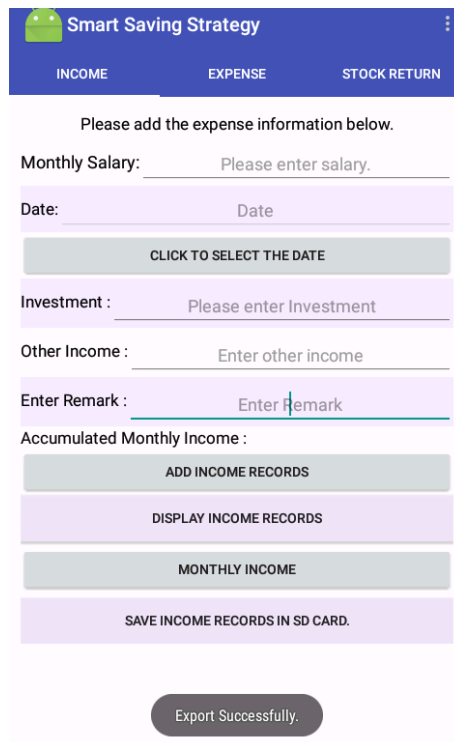
#### 4.1.5 Export the income records to the external storage such as SD card

It is assumed that the users have given permission to write and read a file on a SD card. The permission to write and read is granted by following syntax.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>  
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>[31]
```

When the users click the button “save income records” to SD card, the “CSVExport” “setOnClickListener” event is triggered and the app calls the “exportDB” method from “incomeDBHelper.” The “exportDB” method first checks if an SD card is present on the

mobile device. After successful validation is completed, the method starts to read the attributes and write them to a csv file located in the SD card. After writing the attributes to the file, the method reads all of the records and writes them to the same file until it has read all of the records from the database. After completion of writing records into file, a TRUE value is passed to “CSVExport” in “IncomeMainPageFragment”. After obtaining the TRUE value, the application displays a message “Export Successfully.” Figure 4.3 shows the result of a successful records export to a SD Card. On the other hand, if there are any problems during the process of exporting records to the SD card, the “exportDB” method returns a FALSE value. The application displays a message of “Export unsuccessful” to the users.



**Figure 4.3 The result of successful data export to SD Card**

## **4.2 The overview of Stock Return Activity**

The overview of the stock return activities is described in the following list, and the details will be discussed later.

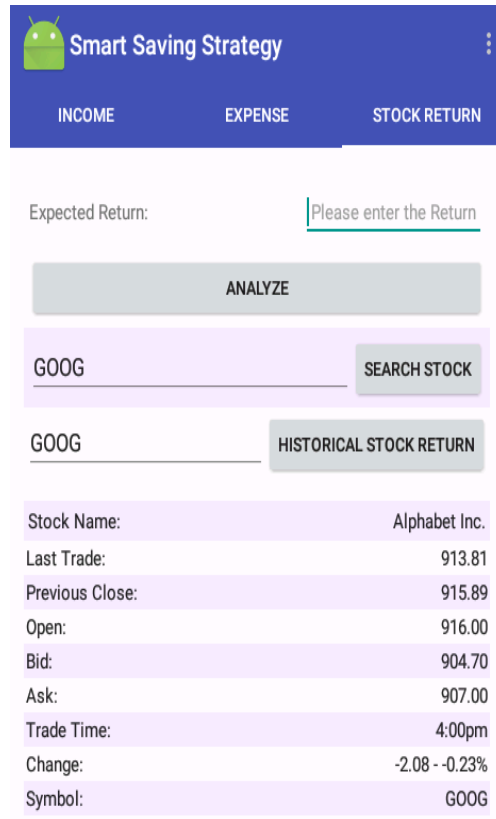
### **4.2.1 savingStrategyFragment**

Figure 4.4 shows that “savingStrategyFragment” provides three services to the users in the main page of the stock return.

1. It analyses the expected return if the stock annual returns are greater or less than the expected return.



2. It searches and displays the stock information.
3. It displays historical stock returns.



**Figure 4.4** The main page of the stock return

#### **4.2.2 historicalDataConstructor**

This constructor activity provides initial values to the parameters and helps to get data from arrays and set send data to the user interface [32].

#### **4.2.3 historicalDataAdapter**

This adapter activity converts data items into views and displays the converted data in the user interface [33].

#### **4.2.4 HistoricalDataItem**

The “HistoricalDataItem” activity provides initial values to the parameters and helps to supply stock information to the “savingStrategyFragment” activity.

### **4.3 The Operation of The Stock Return**

This is the core part of the stock return feature. It starts to use http protocol to communicate with the data source. Afterwards, the data source sends historical stock data in csv file format.

The program reads the csv file and stores the data in different arrays. Those arrays are passed to the “historicalDataAdapter” to display in the user interface and calculate the returns.

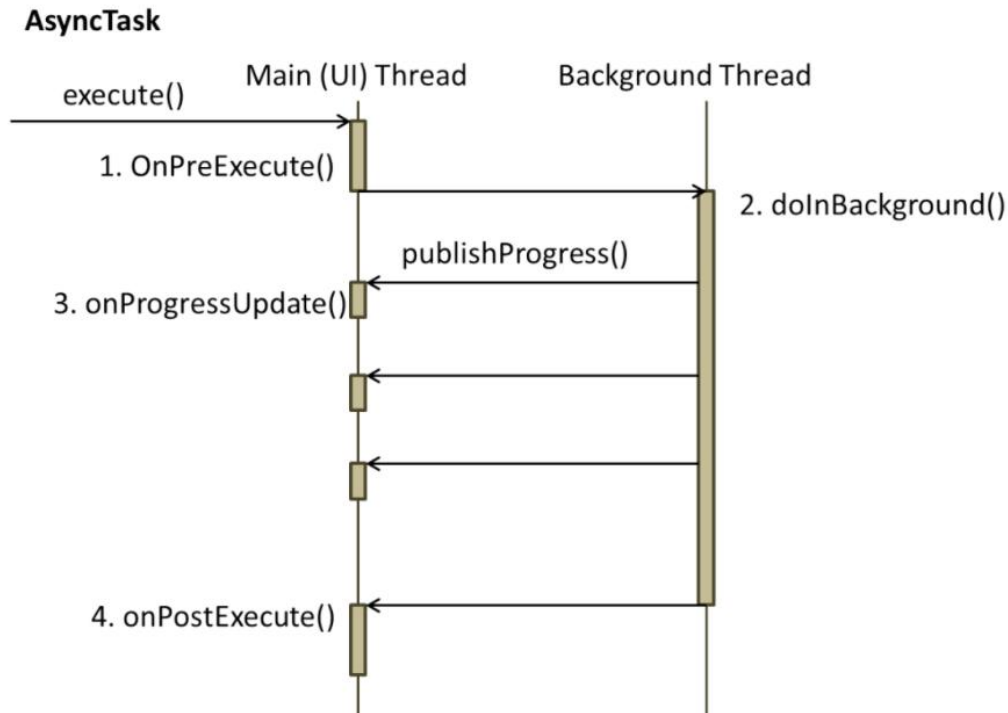
#### **4.3.1 Saving Strategy Fragment**

As mentioned in “stock return overview,” analysing the stock returns and displaying the stock returns are the most important features to the users. All of the features in stock return activity are required to obtain stock information from the remote database via Internet access, use the data to display on the user interface, and analyse the stock return. When the users request to display the stock information, the app starts to download data from the data source in the background thread. In the downloading case, three steps are required to collect the data. Firstly, the app starts to check the network availability before downloading the data. Secondly, it sends an http request to the remote server and waits for the server response. After successful communication with the server, it sends back a stock information csv file. If network connectivity is not available on the mobile device, the app pops up a toast box with text: “No Internet service.” The app will not perform any tasks until Internet service is available. After the validation of network availability, the question comes as to how to download the data. Users would lose control and be blocked on the main thread if the downloading task is run on the main thread. The “asyncTask” class is introduced to solve this situation. “AsyncTask” allows those tasks that interrupt the services in the main thread (UI thread) and require long calculation time to perform in a separated thread called the background thread. “Asynctask” allows users to retain control in the UI thread without interrupting the user’s proper operation. The user can freely dismiss the task and move back to the previous page.

The Figure 4.5 demonstrates the method flow of the “asyncTask.”

There are four main methods in the “asyncTask” process [34].

1. “onPreExecute()” is called on the main thread (UI thread). This method is used to initialise tasks.
2. “doInBackground()” is called on the background thread right after the “onPreExecute” method and is used to process time-consuming tasks.
3. “onProgressUpdate()” is called when it displays the progress on the main thread while the task is still running.
4. onPostExecute () is called after the task is completed on the background thread. The result on the background thread is returned to the main thread.

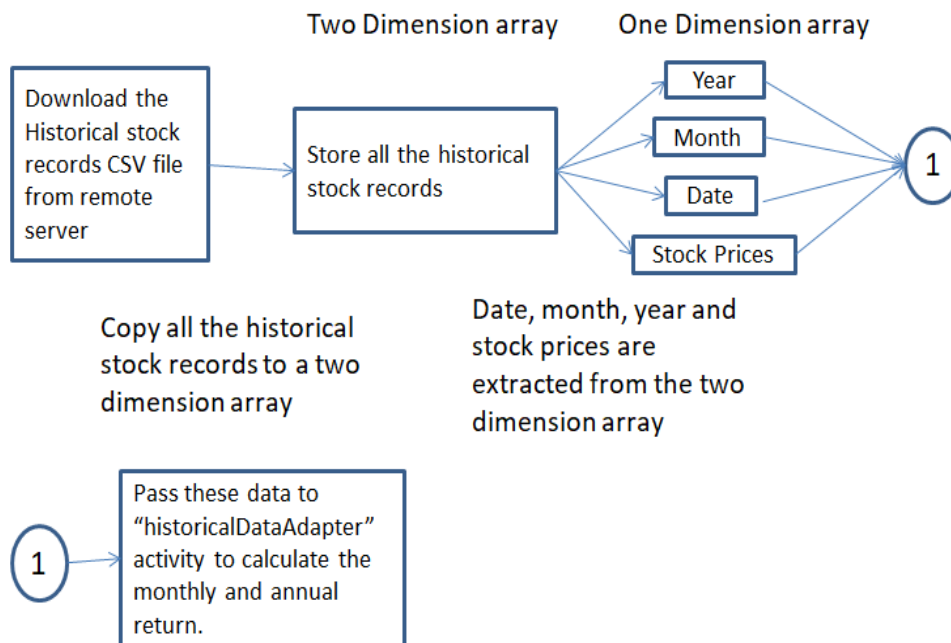


**Figure 4.5 Method Flow of the AsyncTask**

When users ask for the stock detail information in the Saving Strategy Fragment activity, “asynTask” delegates the URL connection operation to the background thread. While the download task is executing, the “asynTask” uses the “progressDialog” class to indicate the progress on the user interface by displaying the progress bar or other indicator. Once the task is completed in the background thread, the “progressDialog” is dismissed, and the result is passed to the “OnPostExecute” method, which is then called from the main thread [35]. After completion of the “asynTask” to download the stock information from the background thread, data cleaning and data filtering are applied to remove and retrieve the meaningful data from the raw data.

### 4.3.2 Stock Return Activity

The stock return activity is mainly used to obtain historical stock information from the data provider via Internet access and store those data to different arrays, such as one-dimensional arrays and two-dimensional arrays, in preparation for the work of calculating the monthly and annual returns. The data providers send the historical data file in a csv format. This file has six attributes, which are date, open, high, low, close, and volume, and generally contains over a thousand data sets. Thereafter, all data are copied to the two-dimensional arrays as a large data table, which is also a source for the lookup table. Then date, month, and year are extracted from the date attribute; stored in separate one-dimensional arrays; and passed to the “historicalDataAdapter” activity for the calculation of the monthly and annually return.



**Figure 4.6 The flow diagram of the Stock Return Activity**

As the general process of the stock return activity has been discussed, the following will describe the process of this activity in detail. Figure 4.6 demonstrates the process of the stock return activity. The stock return activity starts when the user enters the symbol and presses the "historical stock return" button in the "saving strategy fragment," the symbol is passed to the stock return activity to download the stock data. As discussed in the saving strategy fragment section, "asyncTask" is utilised to download the historical data in a separate thread. It ensures that the user can still retain control in the main thread. In "asyncTask," the historical data is copied row-by-row, splits the attributes by comma, and stores the data to the two-dimensional array called "items." This array is used to calculate the returns. After the completion of the "asyncTask," the "dateArrayConversion" method is called. This method simply chops the date attributes into date, month, and year and puts them into three different one-dimensional arrays. At the end, the symbol, item, date, month, and year arrays are transferred to the "historicalDataAdapter" activity for displaying the stock information - monthly and annual returns - to the user interface.

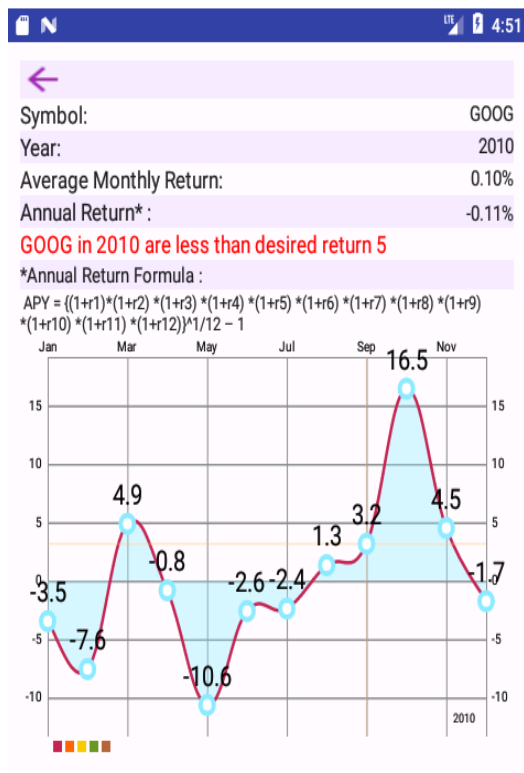
### 4.3.3 HistoricalDataAdapter

After organising the historical data in the stock return activity, "historicalDataAdapter" uses those data to calculate the monthly and annual returns. This activity also compares the

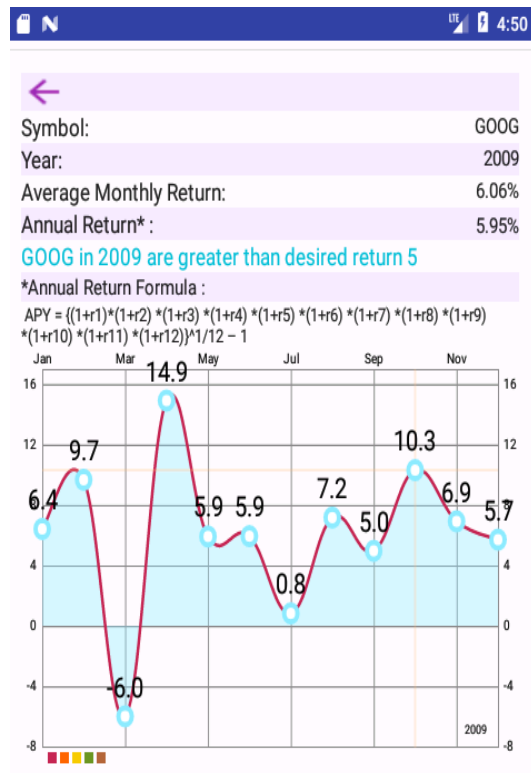
expected return entered by the user with the stock annual return. There are three different scenarios in which desired returns are analysed. Figure 4.7 illustrates all the scenarios.

Scenario one occurs when the stock annual return is less than the desired return (expected return). Then the app displays a text with the format of “Symbol” in “historical year” are less than desired return “desired return.” Scenario two happens when the stock annual return is greater than the desired return (expected return). Then the app displays a text with the format of “Symbol” in “historical year” are greater than desired return “desired return.” Scenario three pertains when the annual return does not cover the whole period of the year. For example, the 2017 annual return covers the period from January 2017 through December 2017. At the time of this writing, there are only seven monthly returns available in 2017. The app displays a text with “For those investments less than one year long, we will provide monthly return calculation for the App users.” The last scenario occurs if there is no desired return input from the users. Then, the app displays a text with “There is no desired return entered.”

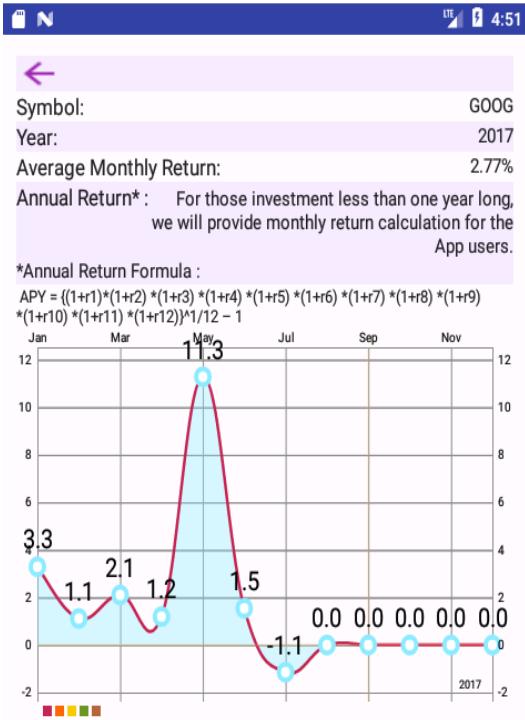
After the app completes the calculation of historical returns and comparison between desired return and stock annual return, the activity passes those data to a “listView” displaying the results on the user interface.



Scenario one: stock annual return is less than the desired return.



Scenario two: stock annual return is greater than the desired return.



Scenario three: annual return does not cover the whole period of the year

**Figure 4.7 Three different scenarios for analysing the desired return**

## **5 Mobile Application Testing and Quality Evaluation**

In order to evaluate the usability and quality of the financial budgeting mobile application, volunteers were invited to participate in a pilot test of the mobile app. During the test, data and user feedback were collected to measure the performance of the app. The feedback was taken into account by the developer for future improvements to the app. The other purpose of the testing is to discover what users are thinking when they are testing the app.

### **5.1 Recruitment of Volunteers**

A convenience sampling technique [36] was applied to collect data in the initial stage of the study regarding the specifications of the Financial Budgeting Mobile Application and in the pilot testing for the mobile application in terms of usability and to gather user opinions. Participants were asked to join the study because they were conveniently available to the developers/researchers. Friends of mine were accessed via WeChat or WhatsApp as the convenience sample to participate in this study.

### **5.2 Ethical Issues**

According to the guidance for Internet-mediated research (IMR), when collecting research data via the Internet, the researcher is required to obtain student dissertation ethical approval from the University of Stirling. The ethical approval form was already submitted and approved by the University.

After the approval from the University, potential participants were invited to engage in a brief (10 minute) telephone briefing session. This contact provided the potential participants with the opportunity to ask questions about the trial. After the telephone briefing session, a software evaluation information sheet, a requirement specification information sheet, and an informed consent form were provided to the participants via email. Then the informed consent of the participants for collecting data (including demographic data) was obtained via email prior to the beginning of the study.

### **5.3 Mobile Application Test with Volunteers**

During development of the Financial Budgeting Mobile Application, a beta test was conducted with a focus group. In this beta test, three users were recruited via a convenient sampling method to give feedback concerning this app. The information collected was used to facilitate the redesign of this app to better address the users' needs concerning financial budgeting. After improvement, a usability test was conducted to test the beta version of the financial budgeting

mobile app. Another three volunteers were recruited via a convenient sampling method to join this usability test.

In the beta test (refer to appendix 3 for the test plan of the Financial Budgeting Mobile Application), the volunteers made comments on three aspects of the application.

1. The text in grey colour on a white background was hard to read. For example, stock information on the stock return user interface was in grey colour on a white paint background.
  - 1.1 In response to this comment, the alternating row colours for background technique will be applied to the user interface to increase the readability of the app and recognisability of the text. Odd and even rows will be set to white and purple colour respectively. Plus, the text's colour will be changed to black.
2. The volunteers also requested that the app provide a backup option for the income and expense records. They expected the records could be saved to external storage.
  - 2.1 To fulfil their expectation, the backup feature is now added to the app accordingly.
3. The historical stock monthly returns should be displayed on a graph such as a bar chart or line chart.
  - 3.1 To rectify this issue, the monthly returns result can now displayed on a graph in the app.

After the adjustment of the mobile app according to the users' comments in the beta test, the app was ready for the "usability test." Three volunteers participated in the "usability test." They followed the instructions on the usability test plan (refer to appendix 4 for the usability test plan of the Financial Budgeting Mobile Application) to test the app functions, performance, and quality. After testing the app, they answered the questions on the questionnaires regarding the app functions, performance, and quality. The summary of the test plan is listed below:

1. At the beginning of the test, the volunteers were requested to install the application to their mobile device.
2. Then, the income feature was tested first. In this section, they were required to add, delete, edit, and update the records to the app. The aim was to check whether the database operation implemented correctly.



3. After completion of the database operation, the function of “calculate the accumulated income” was tested. This function should calculate and display the total of the income to the users.
4. As per the recommendation from the beta test, “the export records to SD card” were added to the application. The backup file should be stored in the SD card.
5. The testing procedures for expense features were the same as the income’s testing procedures.
6. The test of stock return was different from the other tests as it was not related to a database operation. The test focused on displaying stock information, showing the historical monthly return, and analysing the desired return.

#### **5.4 Evaluation of Test Results**

The findings of the aforementioned “usability test” indicated that the three participants were satisfied with the app in terms of function, user interface, usability, and performance. All of the tasks in the test plan were successfully completed. The findings showed that the features relating to stock returns helped the users to understand stock performance by presenting the historical performance. The line graphical view gave them an overview of stock performance by providing the past monthly return, annualized return, and the movement of the stock, which also indicated the stock’s variance. Hence, participants could analyse the stock in a more comprehensive way.

The findings also demonstrated that participants felt comfortable with utilising the app to perform database operations such as adding, deleting, editing, and updating. They added their records to the database within seven minutes in general. Displaying all the records from the database was done within one minute. The longest process of the database operation was “edit the existing records” because it involved several steps. The participants first displayed all the records and then selected the records they wanted to edit. After selection of the record, they clicked the “edit record” button to edit the record and update the database accordingly. The whole process normally took approximately eight to ten minutes. The processing time varied depending on how many records they wanted to edit.

To summarise the above findings, the app was able to assist people to save money in different ways.

1. Firstly, the app could help people monitor the income and expenses in their daily lives. Thus, they could measure their net income, save that money in the bank, and earn interest on it.

2. Secondly, the app could give people another alternative to saving money if they wanted to take the risk of investing in stocks. For the people who were willing to take higher risks, the stock savings plan could provide a better return than banks. However, it requires investors to invest in stock on a regular monthly basis. The app can provide stock information to users so that they can analyse the stock in a more objective way.

It is hoped that this essay would help developers to create applications suitable for effective money saving in the future.

## **6 Future Development**

### **6.1 Randomized Control Trial**

Despite the above findings, there are some limitations to this study. Firstly, the small sample size of the two tests (beta test and usability test) may not reflect the needs of the majority from different backgrounds and cultures as the opinions of the small groups of participants may only reflect their personal needs.

In order to measure the usability and effectiveness of the app in a more accurate way, it is recommended to increase the sample size, which enhances the accuracy of the testing. Certainly, a randomised control trial (RCT) is also recommended to be conducted in the above two testing stages. RCT requires volunteers randomly assigned into two separate groups—a test group and a control group respectively. The test group will use the app; whereas, another group will not. Thus, we can compare the data from these two groups [37]. The outcome will demonstrate more validly and reliably the effects of this app.

### **6.2 Widen the User Base**

#### **6.2.1 Cross Platform Mobile Development**

The mobile application is only designed for Android phones. It can also be extended to other mobile operating system (OS) such as the Blackberry OS, the Windows Phone OS, and the iPhone OS. Therefore, the popularity of this app will be enhanced.

### **6.3 Internationalising and Localising the App**

As to further development in the future, it is recommended that the app support different languages and cultures for local users in order to increase the app's user base. Eventually, the app can overcome language barriers, and local users in different countries can utilise the app confidently.

### **6.4 Purchasing Stock Feature**

The feature of purchasing stock is considered a further value to the app and to be added in the future. After evaluating the income and expense records of the users' daily life and identifying the desired return for their savings, users may be interested in purchasing stock through the app. The users could, therefore, purchase the stocks they desired immediately.

## **7 Conclusion**

To conclude, the goal of this project is to assist people to track income and expenses in their daily lives and provide stock information for their investment analysis if they are interested in investing in the stock market on a monthly basis. The beta test and the usability test have indicated that people have positive feedback for using this app to save money. As mentioned in the future development section, the improvement of this app will come from using randomized control trials in conducting the tests, unlocking the user base, and adding the stock purchasing feature. Therefore, the usability and quality of this app can be further enhanced. It is hoped that this thesis will motivate other developers to further study this area.

## 8 References

- [1] Personal financial planner: A mobile application that implementing forward chaining technique for notification mechanism. (2014). 2014 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), Computer Applications and Industrial Electronics (ISCAIE), 2014 IEEE Symposium on, 65. doi:10.1109/ISCAIE.2014.7010211
  
- [2] Beshouri, Chris; Chaia, Alberto; Cober, Beth; Gravråk, Jon (2010). *Banking on mobile to deliver financial services to the poor. Global Financial Inclusion*. McKinsey & Company's Social Sector office.
  
- [3] Market reach of the most popular Android app categories worldwide as of March 2017. Retrieved July 20, 2017 from <https://www.statista.com/statistics/200855/favourite-smartphone-app-categories-by-share-of-smartphone-users/>
  
- [4] Mark J. P. Anson, Frank J. Fabozzi and Frank J. Jones. (2011). *The Handbook of Traditional and Alternative Investment Vehicles: Investment Characteristics and Strategies*. Copyright © 2011 John Wiley & Sons, Inc.
  
- [5] What is android? Retrieved July 20, 2017 from <http://developer.android.com/guide/basics/what-isandroid.html>
  
- [6] System And Kernel Security. Retrieved July 26, 2017, from <https://source.android.com/security/overview/kernel-security>.
  
- [7] Platform Architecture - Java API Framework. Retrieved July 26, 2017, from <https://developer.android.com/guide/platform/index.html#linux-kernel>.
  
- [8] Platform Architecture - System Apps. Retrieved July 26, 2017, from <https://developer.android.com/guide/platform/index.html#linux-kernel>.
  
- [9] Learning Android Application Programming: Creating an Android User Interface - onCreate, retrieved from 26/07/2017, <http://www.informit.com/articles/article.aspx?p=2168079&seqNum=4>
  
- [10] Learning Android Application Programming: Creating an Android User Interface - onStart, re-trieved from 26/07/2017,

<http://www.informit.com/articles/article.aspx?p=2168079&seqNum=4>

- [11] The Activity Lifecycle - onResume, retrieved from 26/07/2017,  
<https://developer.android.com/guide/components/activities/activity-lifecycle.html>
- [12] The Activity Lifecycle - onPause, retrieved from 26/07/2017,  
<https://developer.android.com/guide/components/activities/activity-lifecycle.html>
- [13] The Activity Lifecycle - onStop, retrieved from 26/07/2017,  
<https://developer.android.com/guide/components/activities/activity-lifecycle.html>
- [14] The activity lifecycle, retrieved from 26/07/2017,  
<https://developer.android.com/guide/components/activities/activity-lifecycle.html>
- [15] Fahad, A. (2012). A Critical Analysis and Treatment of Important UML Diagrams Enhancing Modelling Power. *Intelligent Information Management*, 4, 231-237.
- [16] Introduction to User-Centered Design  
<http://www.usabilityfirst.com/about-usability/introduction-to-user-centered-design/>
- [17] Best Practices for User Interface Available :  
<https://developer.android.com/training/best-ui.html> [8 August 2017]
- [18] Android List View, retrieved from  
27/07/2017,[https://www.tutorialspoint.com/android/android\\_list\\_view.htm](https://www.tutorialspoint.com/android/android_list_view.htm)
- [19] Introduction to Listview in Android, 14/07/2015 retrieved from 27/07/2017,  
<https://www.slideshare.net/technoguff/introduction-to-listview-in-android>
- [20] Android Tab with Swipe Views, ViewPager, FragmentPagerAdapter in Android Studio  
retrieved from 27/07/2017,<https://inducesmile.com/android/android-tab-with-swipe-views-viewpager-fragmentpageradapter-in-android-studio/>
- [21] MPAndroidChar , retrieved from  
27/07/2017,<https://github.com/PhilJay/MPAndroidChart>
- [22] Android.database.sqlite, retrieved from 27/07/2017,  
<https://developer.android.com/reference/android/database/sqlite/package-summary.html>

- [23] Windows 10 Specifications & Systems Requirements. Retrieved from July 20, 2017, <https://www.microsoft.com/en-us/windows/windows-10-specifications>
- [24] The Android Story. Retrieved from July 20, 2017, <https://www.android.com/history/#/marshmallow>
- [25] What is API Level? Retrieved from July 20, 2017, <https://developer.android.com/guide/topics/manifest/uses-sdk-element.html>
- [26] Android Studio. The Official IDE for Android. Retrieved from July 20, 2017, <https://developer.android.com/studio/index.html>
- [27] Java SE Downloads. Retrieved from 29/07/2017, <http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html>
- [28] XML. Retrieved from 29/07/2017, <https://www.xml.com/>
- [29] JAVA. Retrieved from 29/07/2017, <https://java.com/en/>
- [30] MYSQL. Retrieved from 29/07/2017, <https://dev.mysql.com/>
- [31] <uses-permission>. Retrieved from 29/07/2017, <https://developer.android.com/guide/topics/manifest/uses-permission-element.html>
- [32] Using an ArrayAdapter with ListView. Retrieved from, <https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView> 30/07/2017
- [33] Adapter Tutorial With Example In Android Studio. Retrieved from, <http://abhiandroid.com/ui/adapter> 30/07/2017
- [34] AsyncTask, Retrieved from 30/07/2017, <https://developer.android.com/reference/android/os/AsyncTask.html>
- [35] AsyncTask usage summary for asynchronous classes in Android. Retrieved from, <https://www.codeday.top/2017/03/04/19812.html> 30/07/2017  
AsyncTask usage summary, <http://corochann.com/async-task-usage-summary-341.html> 30/07/2017

[36] Explorable.com. (2009, Sep 16). Convenience Sampling. Retrieved Nov 13, 2015, from <https://explorable.com/convenience-sampling>

[37] MacGill, M. (2016, January 29). "What is a randomized controlled trial in medical research?." Medical News Today. Retrieved from <http://www.medicalnewstoday.com/articles/280574.php>.



# Appendix 1 – User guide

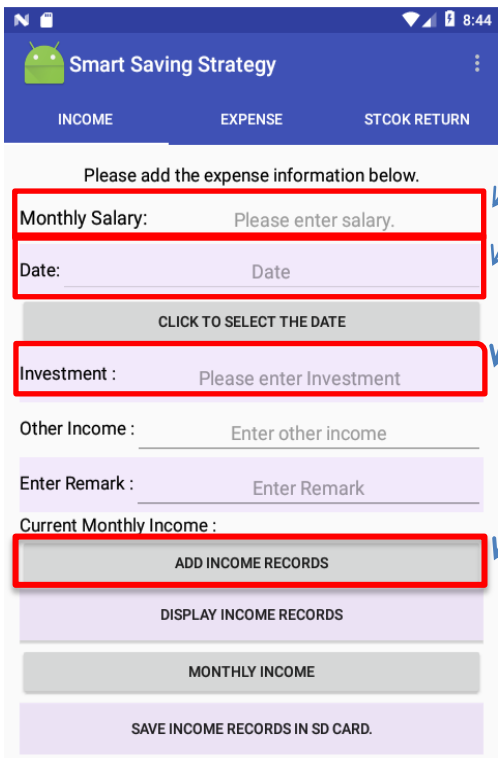
## 1. Introduction

The Financial Budgeting Mobile Application is not only an expense monitoring (no Internet connection required) mobile application, but it also provides information for users to analyse a stock for their stock saving plan.


## Getting Started

After the app is launched, the income page is displayed as below, ready to add income records to the app.

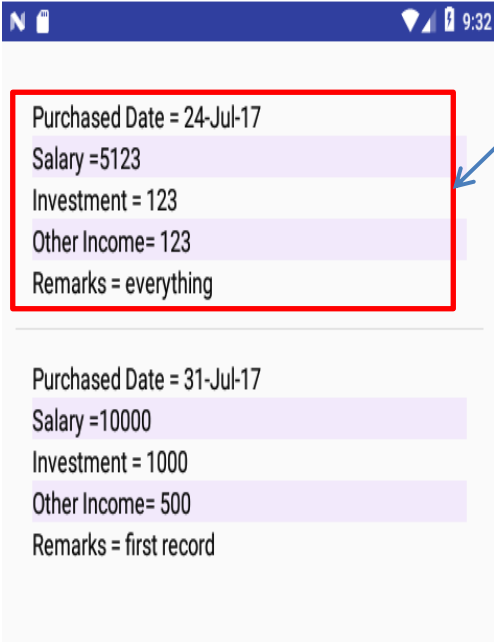
## Adding Income Items

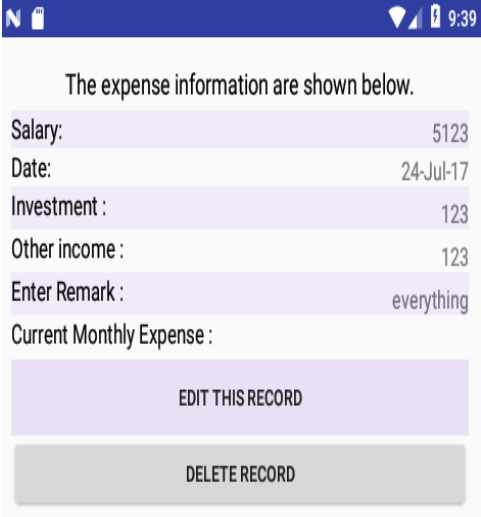
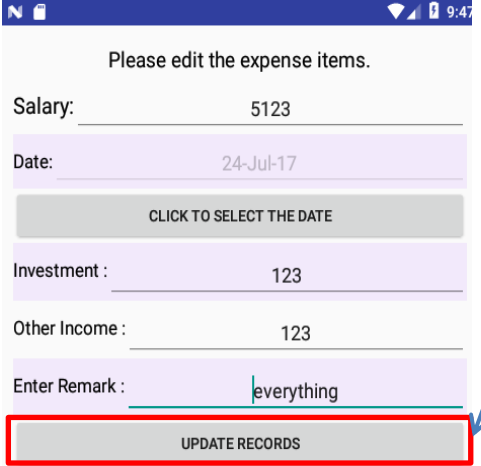


1. When inputting the data to the app, Monthly Salary, Date, and Investment fields are mandatory, which means the fields cannot be empty.
2. After filling all the fields, press the “ADD INCOME RECORDS” button to add the income records to the app.

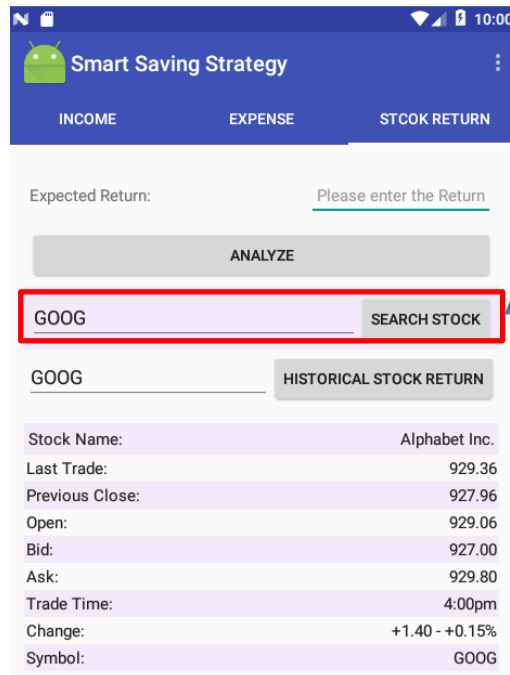
 <p>Purchased Date = 24-Jul-17 Salary =5123 Investment = 123 Other Income= 123 Remarks = everything</p> <hr/> <p>Purchased Date = 31-Jul-17 Salary =10000 Investment = 1000 Other Income= 500 Remarks = first record</p>	<p>The added items are shown on the left hand side of the screen.</p>
---	---

Editing or deleting an income items

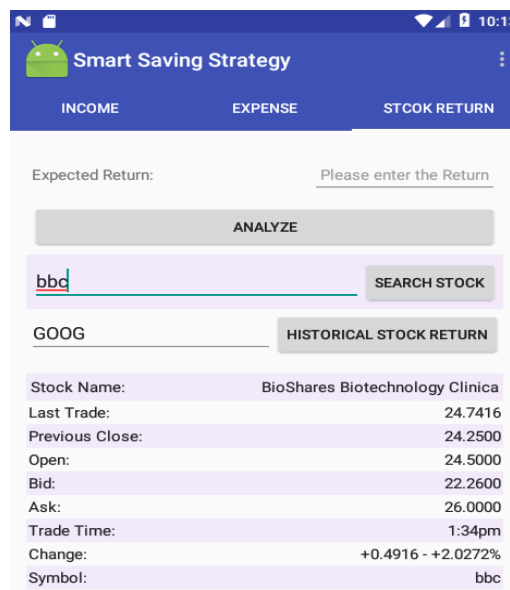
 <p>Purchased Date = 24-Jul-17 Salary =5123 Investment = 123 Other Income= 123 Remarks = everything</p> <hr/> <p>Purchased Date = 31-Jul-17 Salary =10000 Investment = 1000 Other Income= 500 Remarks = first record</p>	<p>1. To edit or delete items, users simply click on the item.</p>
---	--

 <p>The expense information are shown below.</p> <p>Salary: 5123  Date: 24-Jul-17  Investment : 123  Other income : 123  Enter Remark : everything  Current Monthly Expense :</p> <p>EDIT THIS RECORD</p> <p>DELETE RECORD</p>	<ol style="list-style-type: none"> <li>2. Before deleting the items, it is recommended to check the record again and delete the record.</li> <li>3. Click the “EDIT THIS RECORD” button to edit the records or click the “DELETE RECORD” button to delete this record.</li> </ol>
 <p>Please edit the expense items.</p> <p>Salary: 5123  Date: 24-Jul-17  CLICK TO SELECT THE DATE  Investment : 123  Other Income : 123  Enter Remark : everything</p> <p>UPDATE RECORDS</p>	<ol style="list-style-type: none"> <li>4. Click the “UPDATE RECORDS” button to update the record.</li> </ol>

## Displaying the stock information

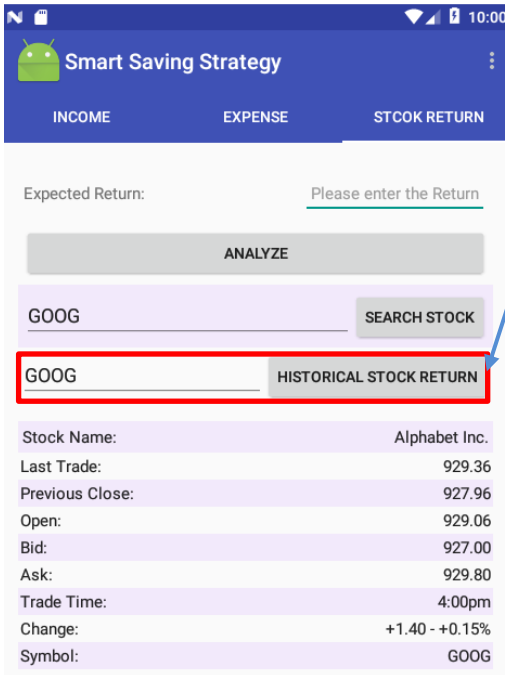
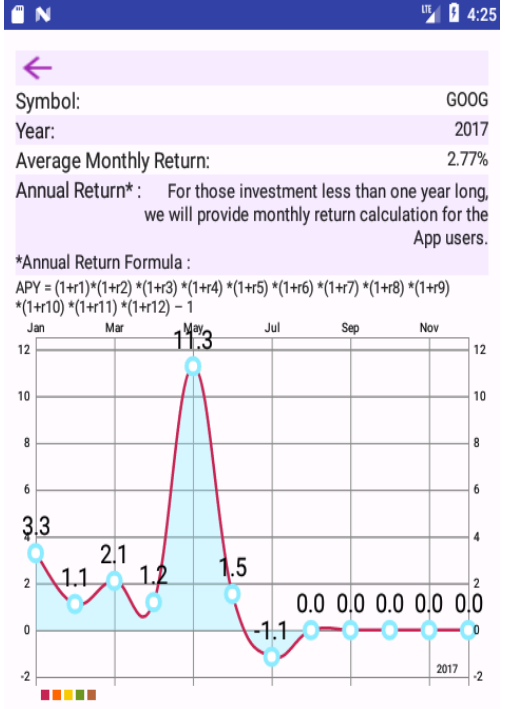


1. The app will automatically load the Google stock information on the screen because it is a pre-entered input field.
2. Remove the GOOG text and enter a stock symbol (e.g. BBC) and then press the “SEARCH STOCK” button.

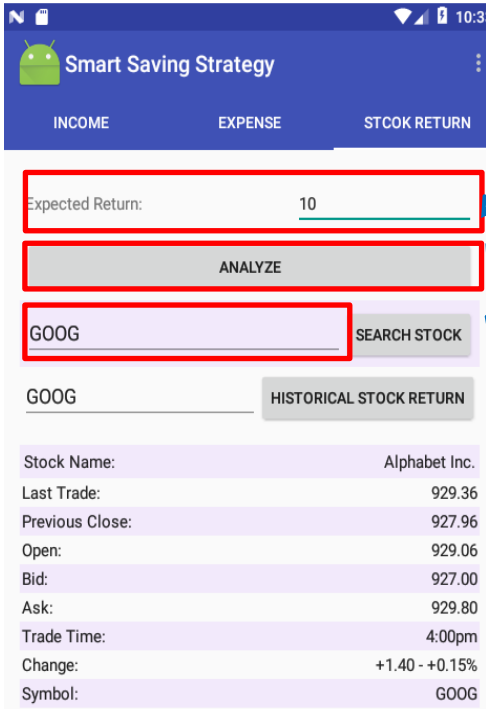
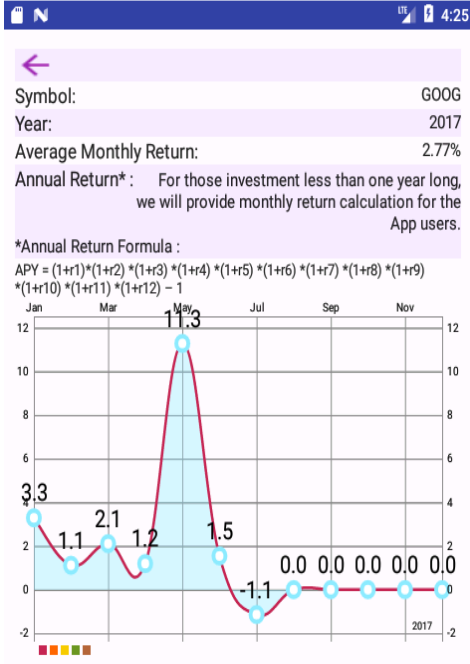


3. The BBC's information is displayed on the screen.

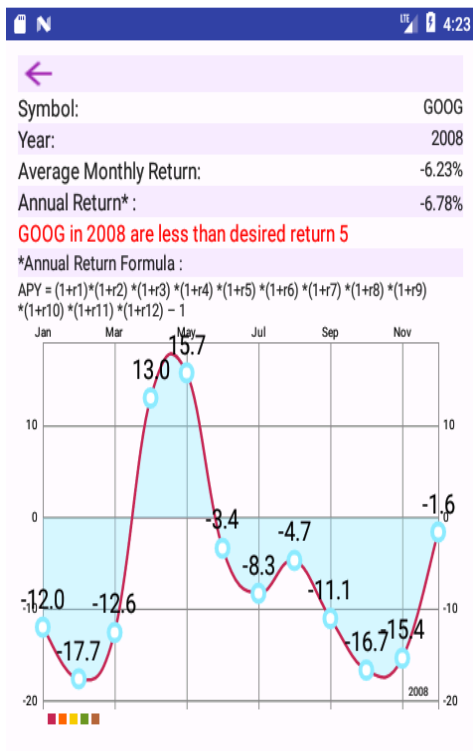
## Showing the historical stock return

 <p>Smart Saving Strategy</p> <p>INCOME EXPENSE <b>STOCK RETURN</b></p> <p>Expected Return: <input type="text" value="Please enter the Return"/></p> <p><b>ANALYZE</b></p> <p>GOOG <b>SEARCH STOCK</b></p> <p><b>GOOG</b> <b>HISTORICAL STOCK RETURN</b></p> <p>Stock Name: Alphabet Inc.          Last Trade: 929.36          Previous Close: 927.96          Open: 929.06          Bid: 927.00          Ask: 929.80          Trade Time: 4:00pm          Change: +1.40 - +0.15%          Symbol: GOOG</p>	<p>1. Enter the stock symbol (GOOG) and press the button “HISTORICAL STOCK RETURN” to display the historical return of the stock.</p>																										
 <p>Symbol: GOOG          Year: 2017          Average Monthly Return: 2.77%          Annual Return*: For those investment less than one year long, we will provide monthly return calculation for the App users.</p> <p>*Annual Return Formula :  <math>APY = (1+r_1) * (1+r_2) * (1+r_3) * (1+r_4) * (1+r_5) * (1+r_6) * (1+r_7) * (1+r_8) * (1+r_9) * (1+r_{10}) * (1+r_{11}) * (1+r_{12}) - 1</math></p> <table border="1"> <thead> <tr> <th>Month</th> <th>Return (%)</th> </tr> </thead> <tbody> <tr><td>Jan</td><td>3.3</td></tr> <tr><td>Feb</td><td>1.1</td></tr> <tr><td>Mar</td><td>2.1</td></tr> <tr><td>Apr</td><td>1.2</td></tr> <tr><td>May</td><td>11.3</td></tr> <tr><td>Jun</td><td>1.5</td></tr> <tr><td>Jul</td><td>-1.1</td></tr> <tr><td>Aug</td><td>0.0</td></tr> <tr><td>Sep</td><td>0.0</td></tr> <tr><td>Oct</td><td>0.0</td></tr> <tr><td>Nov</td><td>0.0</td></tr> <tr><td>Dec</td><td>0.0</td></tr> </tbody> </table>	Month	Return (%)	Jan	3.3	Feb	1.1	Mar	2.1	Apr	1.2	May	11.3	Jun	1.5	Jul	-1.1	Aug	0.0	Sep	0.0	Oct	0.0	Nov	0.0	Dec	0.0	<p>After pressing the button “HISTORICAL STOCK RETURN,” information including year, monthly return, and annual return are displayed on the screen. The example displays the stock monthly return in 2017. Zero means there is no monthly return available at this moment.</p>
Month	Return (%)																										
Jan	3.3																										
Feb	1.1																										
Mar	2.1																										
Apr	1.2																										
May	11.3																										
Jun	1.5																										
Jul	-1.1																										
Aug	0.0																										
Sep	0.0																										
Oct	0.0																										
Nov	0.0																										
Dec	0.0																										

## Comparing the stock return and the expected return

 <p>Expected Return: 10</p> <p>ANALYZE</p> <p>GOOG SEARCH STOCK</p> <p>GOOG HISTORICAL STOCK RETURN</p> <p>Stock Name: Alphabet Inc.</p> <p>Last Trade: 929.36</p> <p>Previous Close: 927.96</p> <p>Open: 929.06</p> <p>Bid: 927.00</p> <p>Ask: 929.80</p> <p>Trade Time: 4:00pm</p> <p>Change: +1.40 - +0.15%</p> <p>Symbol: GOOG</p>	<ol style="list-style-type: none"> <li>1. Enter stock symbol and expected return in the text fields and press the “ANALYZE” button. In this example, expected return is 10 percent and the stock symbol is GOOG.</li> </ol>																								
 <p>Symbol: GOOG</p> <p>Year: 2017</p> <p>Average Monthly Return: 2.77%</p> <p>Annual Return*: For those investment less than one year long, we will provide monthly return calculation for the App users.</p> <p>*Annual Return Formula :</p> $APY = (1+r_1) * (1+r_2) * (1+r_3) * (1+r_4) * (1+r_5) * (1+r_6) * (1+r_7) * (1+r_8) * (1+r_9) * (1+r_{10}) * (1+r_{11}) * (1+r_{12}) - 1$ <table border="1"> <thead> <tr> <th>Month</th> <th>Return (%)</th> </tr> </thead> <tbody> <tr><td>Jan</td><td>3.3</td></tr> <tr><td>Feb</td><td>1.1</td></tr> <tr><td>Mar</td><td>2.1</td></tr> <tr><td>Apr</td><td>1.2</td></tr> <tr><td>May</td><td>11.3</td></tr> <tr><td>Jun</td><td>1.5</td></tr> <tr><td>Jul</td><td>-1.1</td></tr> <tr><td>Aug</td><td>0.0</td></tr> <tr><td>Sep</td><td>0.0</td></tr> <tr><td>Oct</td><td>0.0</td></tr> <tr><td>Nov</td><td>0.0</td></tr> </tbody> </table>	Month	Return (%)	Jan	3.3	Feb	1.1	Mar	2.1	Apr	1.2	May	11.3	Jun	1.5	Jul	-1.1	Aug	0.0	Sep	0.0	Oct	0.0	Nov	0.0	<p>There are three cases that can result.</p> <ol style="list-style-type: none"> <li>1. There is no result shown on the screen if the period does not cover the whole year.</li> <li>2. The result will be printed in red colour if the stock return is lower than the expected return.</li> <li>3. The result will be printed in green colour if the stock return is higher than the expected return.</li> </ol> <p>The result of case one is shown on the left hand side.</p>
Month	Return (%)																								
Jan	3.3																								
Feb	1.1																								
Mar	2.1																								
Apr	1.2																								
May	11.3																								
Jun	1.5																								
Jul	-1.1																								
Aug	0.0																								
Sep	0.0																								
Oct	0.0																								
Nov	0.0																								

The screen below is case 2.



The screen below is case 3.



The end of the user guide

## Appendix 2 – Installation guide

### Financial Budgeting Mobile Application Installation Requirement

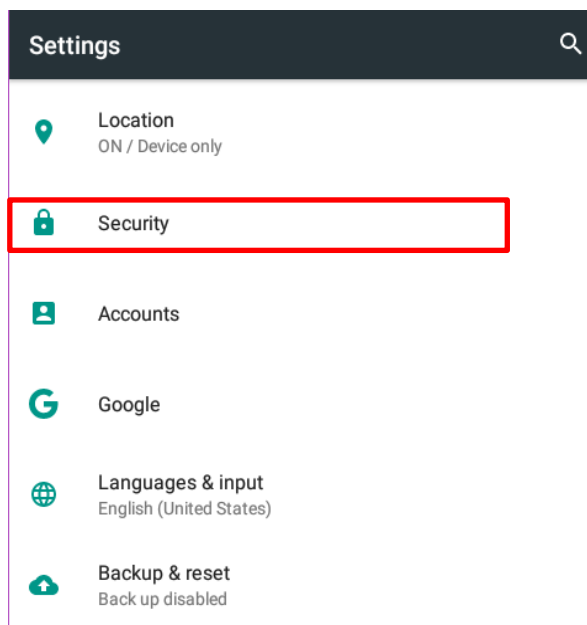
- The Android device must be running Android 3.2 or above.
- The unknown sources option is enabled on your Android device.

#### Step 1

Copy the APK file to your Android device.

#### Step 2

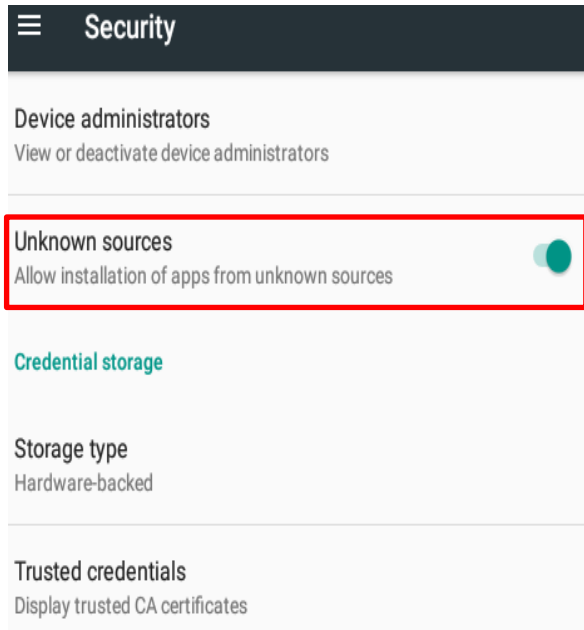
Go to the Settings page and select “Security.”





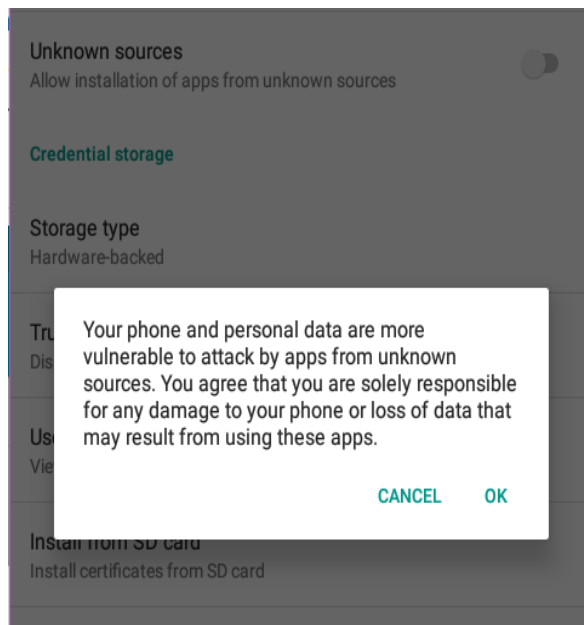
### Step 3

Toggle the switch button to enable the device to install apps outside the Google Play Store.



### Step 4

A pop-up warning message is shown on screen that installing apps from unknown sources increases the risk of being attacked by the apps.



### Step 5

Search for the APK of the Financial Budgeting Mobile Application, select the file, and press the "INSTALL" button.

The End of The Installation Guide

## Appendix 3 – Beta Test

This user test presents you with a number of tasks to perform using the Financial Budgeting Mobile Application. For each task, please read the description and then attempt to use the app to complete it. You may use any help facility the app provides. When you have completed the task, or if you find you cannot complete it, please answer the questions that follow the task description.

### **Task 1: Add daily income / expense to the app**

You are invited to input the item, date, prices and save them to the app.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. "Item, date and price fields cannot be empty").
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

### **Task 2: Edit / Delete item from the app.**

You are invited to delete item, date and prices from the app.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. The selected item cannot be deleted.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

### **Task 3: View the income / expense records.**

You are invited to view the incomes / expense by clicking the display monthly income /expense records button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot display the income / expense records.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

#### **Task 4: View historical stock return records**

You are invited to view the historical stock returns by clicking the historical stock return button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot display the historical stock return records.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

#### **Task 5: View stock information**

You are invited to view stock information by clicking the search stock button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot display the stock information.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

#### **Task 6: Analyse the desired return if the stock annual returns are greater / less than the desired return.**

You are invited to evaluate the result of analysis the desired return by clicking the analyse button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot correctly analyse the desired return.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

## Appendix 4 – Usability Test

This user test presents you with a number of tasks to perform using the Financial Budgeting Mobile Application. For each task, please read the description and then attempt to use the app to complete it. You may use any help facility the app provides. When you have completed the task, or if you find you cannot complete it, please answer the questions that follow the task description.

### **Task 1: Add daily expense to the app**

You are invited to input the item, date, prices and save them to the app.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. "Item, date and price fields cannot be empty").
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

### **Task 2: Edit / Delete item from the app.**

You are invited to delete item, date and prices from the app.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. The selected item cannot be deleted.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

### **Task 3: View the income / expense records.**

You are invited to view the incomes / expense by clicking the display monthly income /expense records button.

Feedback

4. Were you able to complete the task? Yes / No
5. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot display the income / expense records.)
6. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

#### **Task 4: Export incomes / expense records**

You are invited to export the income / expense records by clicking the save income /expense records in SD card button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot export the income / expense records to SD Card.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

#### **Task 5: Calculate accumulated total expenses**

You are invited to calculate accumulated total expenses by clicking the monthly expense button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot calculate accumulated total incomes / expenses.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

#### **Task 6: View historical stock returns**

You are invited to view the historical stock returns by clicking the historical return button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot display the historical stock returns.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

#### **Task 7: View stock information**

You are invited to view the stock information by clicking the search stock button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot display the stock information.)

3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}

**Task 8: Analyse the desired return if the stock annual returns are greater / less than the desired return.**

You are invited to evaluate the result of analysis the desired return by clicking the analyse button.

Feedback

1. Were you able to complete the task? Yes / No
2. If not, please describe how far you got and any error messages that were displayed (e.g. the app cannot correctly analyse the desired return.)
3. How do you evaluate the difficulty level of the task? {From 1(=difficult) to 5 (=very easy)}