

**Bachelor of Information Technology**

**BIT302**

**Software Engineering**

**ASSIGNMENT 4**

**FUNDFAIR: VIRTUAL STOCK GAME**

**PROJECT PERIOD: 11 January 2016 – 22 April 2016**

**Student Name: Go Wen Xi Student ID: B1300649**

**Student Name: Student ID:**

**15 Jalan Sri Semantan 1, Off Jalan Semantan, Bukit Damansara, 50490 Kuala Lumpur, Malaysia.**

**Tel : 603-2716 2000, Fax : 603 - 2095 7100**

Contents

[Updated Analysis and Design documentation 2](#_Toc479539259)

[Identification of problems and solutions 2](#_Toc479539260)

[Screen Shots 3](#_Toc479539261)

[Updated Test Plan 3](#_Toc479539262)

[Test objectives 3](#_Toc479539263)

[Test Case 4](#_Toc479539264)

[Integration Test Plan 8](#_Toc479539265)

[System Test Plan 13](#_Toc479539266)

[Test Analysis Report 17](#_Toc479539267)

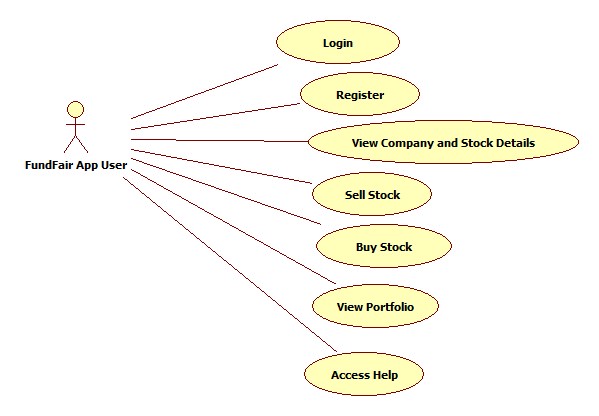
[Conclusion 17](#_Toc479539268)

[Updated Gantt Chart 18](#_Toc479539269)

[References 19](#_Toc479539270)

# Updated Analysis and Design documentation

We had added one more use case which is “Access Tutorial”. This is use case is user access to Tutorial. Tutorial function is to let user understand some important information about stock investment, and user can access Tutorial to learn about it.



Functional Requirement

|  |  |
| --- | --- |
| Use Case | Access Help |
| Actor(s) | FundFair App User |
| Description | FundFair App User wants to access help. |

# Identification of problems and solutions

**Problems**

During iteration 1, we planned to complete 3 use cases: Login, Register and View Company and Stock Details. However, we only managed to complete Login and Register for iteration 1. We got an issue when we try to perform Login after we have done registration.

**Solutions**

We have solved the issue by using Firebase instead of the traditional way. Previously, we used MySQL and PHP to connect and update the registered users to the database. Thus, it needed some times to debug both side when there is any error occurs. After that we switched to the Firebase Authentication, it helps a lot by easing the process of coding for login and register. Firebase Authentication provides easy-to-use SDKs and backend services to authenticate users to the app. Authentication using password and other popular identity providers such as Facebook, Google and Twitter is also available. Using this method, password reset emails will also be handled and sent by the Firebase Authentication (Google Firebase, 2017). There are few other sign in methods available in Firebase Authentication. We are using Email providers as the sign in methods. Firebase is not only providing Authentication, there are many other features such as database, storage, hosting, functions, test lab, crash reporting and so on which will benefit our development process afterwards.

# Iteration 2

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Number | Use Case Name | Use Case Description | Assigned to |
| 3 | View Company and Stock Details | Add a list into Trading Centre and details of the stock. | Lee Xuin Hean & Go Wen Xi |
| 4 | Buy Stock | Implement a buy button to let user buy stocks. | Go Wen XI |
| 5 | View Portfolio | View the stock that buy before in portfolio | Lee Xuin Hean |
| 6 | Sell Stock | Implement a sell button to let user sell stocks | Go Wen Xi |
| 7 | Access Help | List of keyword and description to help user | Lee Xuin Hean |

# Screen Shots

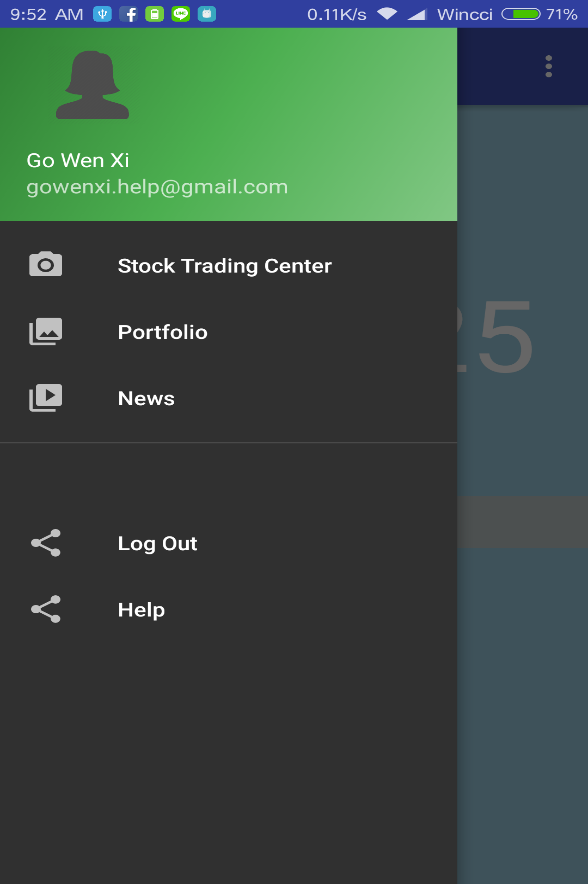
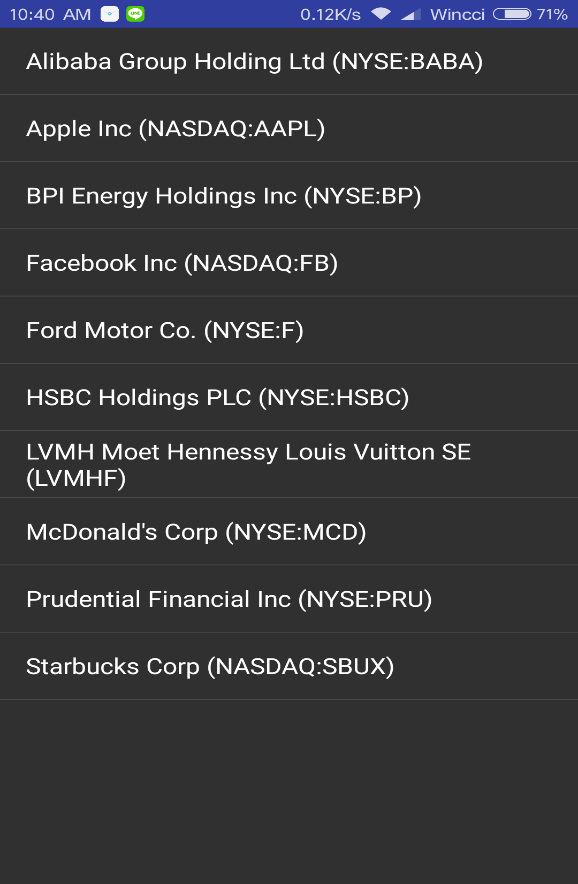
 

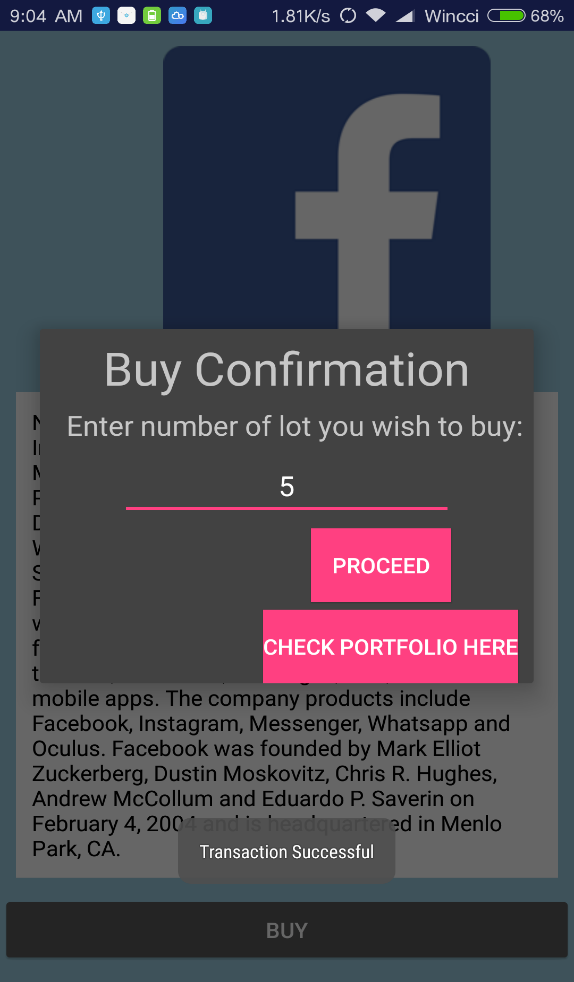
Figure 1(Side Menu Screen) Figure 2 (Stock Trading Center)  

Figure 3 (Stock Details Screen) Figure 4 (Buy Stock at Stock Details Screen)

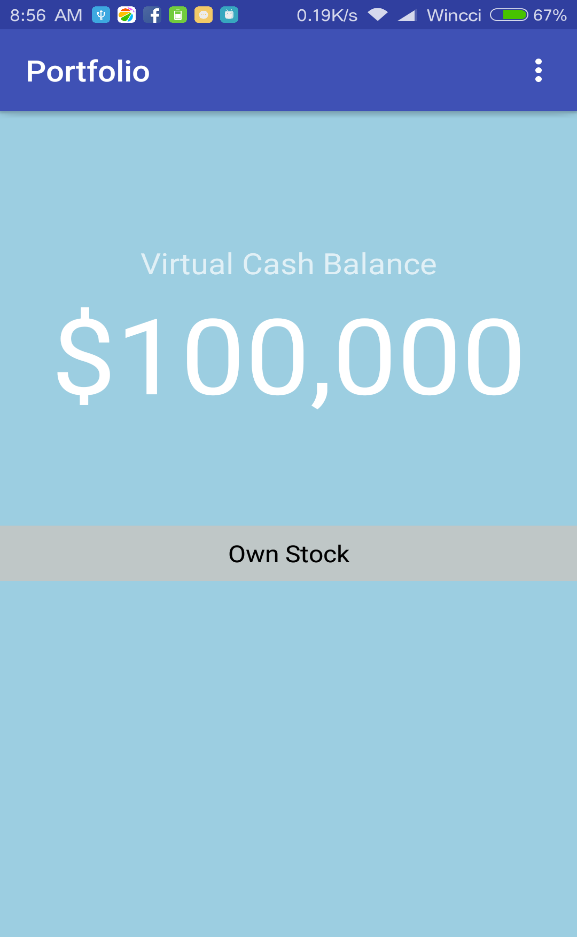
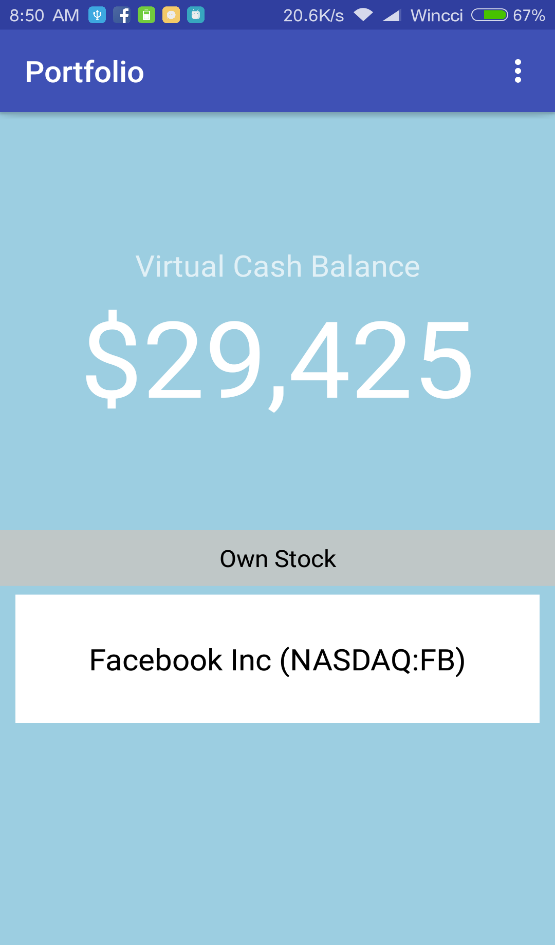
 

Figure 5 (Portfolio- before buy stock) Figure 6 (Portfolio- after buy stock)

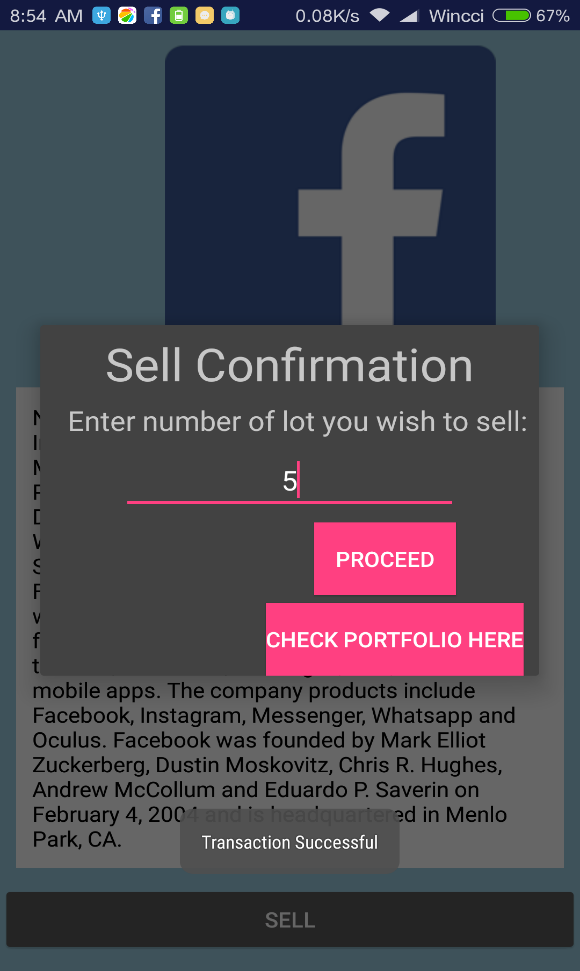
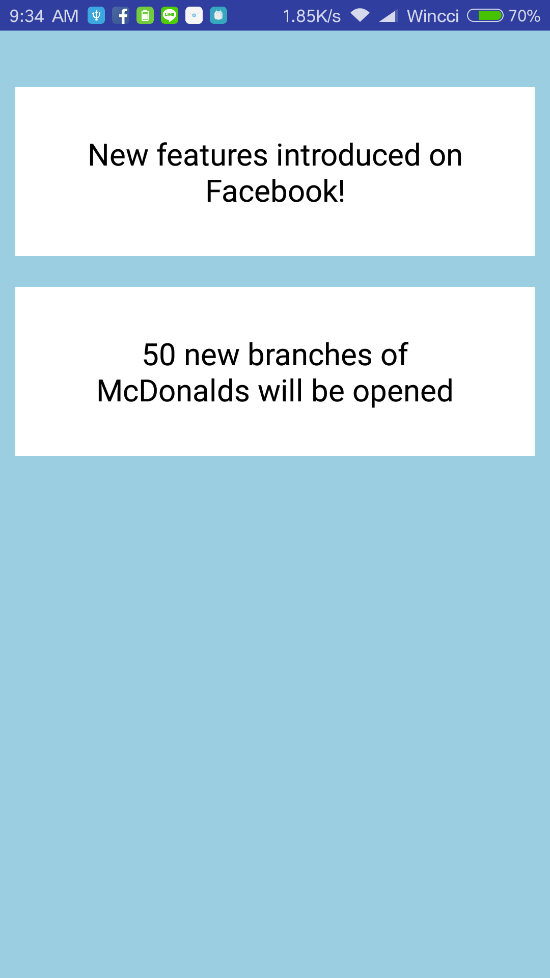
 

Figure 7 (Sell Stock at Portfolio Screen) Figure 8 (News Screen)

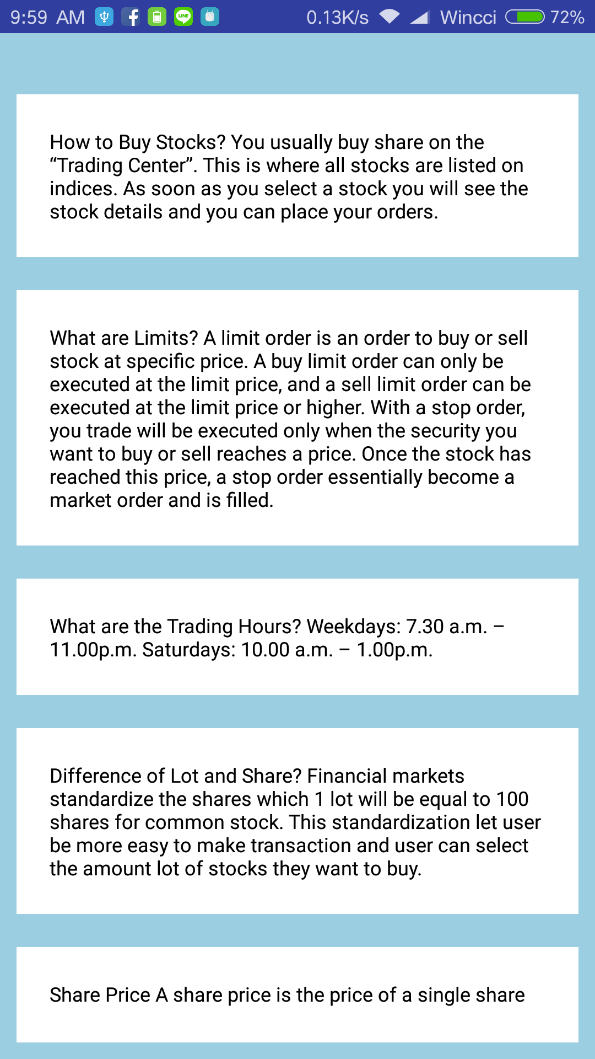


Figure 9 (Help Screen)

# Updated Test objectives

1) To resolve as many errors, bugs or faults as possible in the source code

2) To ensure non-functional performance requirement able to achieved at system testing.

3) To verify functional requirement and design are align with the specification and design description

4) To provide clear documentation like analysis report and unit, integrated and system test results.

5) To test new use function that didn’t implement during Iteration 1.

6) To find defect which may create by developer during developing the software.

# Updated Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Technique | Carried out | Aim | Start Date | End Date |
| Unit Testing | When component of the source code is complete | Used white box testing strategy to test the source code to find out any errors. Code inspection is carry out to make sure the module can meet the requirement | 8 April 2017 | 8 April 2017 |
| Integration Testing | When each component is ready to be integrated, and testing by gather all components that tested in Unit Testing | Used top-down integration technique. To test the overall structure of system to ensure modules of Unit testing able to works well after combining. | 8 April 2017 | 8 April 2017 |
| System Testing | When all modules integrated together | Used black box testing strategy to test the system functionality and performance. All use case will be tested over here to ensure able to meet the requirement. Non-functional requirement is also tested as well. | 9 April 2017 | 9 April 2017 |

# Test Method Specification Evaluation

We use android studio to do white box testing. We use code inspection of android studio to debug is that any error for the source code, if there is error android studio will stop debugging and show error at which line. For system testing, we use android studio to generate APK and we send to our android phone which is LG G3 and Redmi 3. We install FundFair in our phone and apply black box testing for every functionalities and non-functionalities.

Test CaseUnit Testing

|  |
| --- |
| **Unit Test Plan** |
| **Test Case-1.0 Unit Testing -1** |
| Validate whether user can buy stock when cash not enough |
| **Test Case-1.1 Description** |
| To test if the system response to the user if the calculation of total price of share that exceed the cash that user owns. |
| **Test Case -1.2 Source Code** |
| public void onClick(View v) {  int lot = Integer.parseInt(number.getText().toString());  int exceed = lot \* 100 \* currentPricePerShare  if(totalCash < exceed))  {  Toast.makeText(Facebook.this,"Insufficient cash",Toast.LENGTH\_SHORT).show();  }  else  {  Toast.makeText(Facebook.this,"TransactionSuccessful",Toast.LENGTH\_SHORT).show();  mCheckPortfolio.setVisibility(View.VISIBLE);  } |
| **Test Case-1.3 Test Data** |
| |  |  | | --- | --- | | **Number of Test** | **Number of Lot User Entered** | | 1 | 2 | | 2 | 99999 | |
| **Test Case-1.4 Expected Result** |
| |  |  | | --- | --- | | **Number of Test** | **Result** | | 1 | “Transaction Successful” text displayed | | 2 | “Insufficient Cash” text displayed | |
| **Test Case-1.5 Test Outcome** |
| |  |  | | --- | --- | | **Number of Test** | **Result** | | 1 | “Transaction Successful” text displayed | | 2 | “Insufficient Cash” text displayed | |
| **Test Case-1.6 Remark** |
| The buy stock validation work as expected. |

|  |
| --- |
| **Unit Test Plan** |
| **Test Case-2.0 Unit Testing-2** |
| Validate whether user cash amount will be deducted after buy stock |
| **Test Case-2.1 Description** |
| To validate the amount of cash user owned will be deducted after purchase the stock. |
| **Test Case-2.2 Source Code** |
| int lot = Integer.parseInt(number.getText().toString());  int totalPrice = lot \* 100 \* currentPricePerShare;  remaining = totalCash - totalPrice |
| **Test Case-2.3 Test Data** |
| |  |  | | --- | --- | | **Number of Test** | **Number of Lot User Entered to Buy** | | 1 | 5 for currentPricePerShare of 130.15 | | 2 | 8 for currentPricePerShare of 130.15 | |
| **Test Case-2.4 Expected Result** |
| |  |  | | --- | --- | | **Number of Test** | **Result** | | 1 | Cash Amount become 65,075 | | 2 | Cash Amount become 104,120 | |
| **Test Case-2.5 Test Outcome** |
| |  |  | | --- | --- | | **Number of Test** | **Result** | | 1 | 65,075 | | 2 | 104,120 | |
| **Test Case-2.6 Remark** |
| The Amount of Current Cash deducted and displayed correctly after user buy a stock. |

|  |
| --- |
| **Unit Test Plan** |
| **Test Case-3.0 Unit Testing -3** |
| Validate whether user cash amount will be added after sell stock |
| **Test Case-3.1 Description** |
| To validate the amount of cash user owned will be added after sell the stock. |
| **Test Case -3.2 Source Code** |
|  |
| **Test Case-3.3 Test Data** |
| |  |  | | --- | --- | | **Number of Test** | **Number of Lot User Entered to Sell** | | 1 | 2 | | 2 | 99999 | |
| **Test Case-3.4 Expected Result** |
| |  |  | | --- | --- | | **Number of Test** | **Result** | | 1 | Cash Amount become | | 2 | Cash Amount become | |
| **Test Case-3.5 Test Outcome** |
| |  |  | | --- | --- | | **Number of Test** | **Result** | | 1 | Cash Amount become | | 2 | Cash Amount become | |
| **Test Case-3.6 Remark** |
| The Amount of Current Cash added and displayed correctly after user sell a stock. |

### Integration Test Plan

|  |
| --- |
| **Integration Test Plan** |
| **Test Case--4.0 Integration Testing-1** |
| Lead to trading centre when trading centre button is clicked |
| **Test Case--4.1 Description** |
| To test whether will be directed lead to the trading centre when clicked the trading centre button. |
| **Test Case--4.2 Diagram** |
| **C:\Users\Wen Xi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\17837712_1169528246503861_656349357_o.png C:\Users\Wen Xi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\17838630_1169530103170342_371457857_o.png** |
| **Test Case--4.3 Test Data** |
| Stock Trading Centre button is clicked |
| **Test Case--4.4 Expected Result** |
| Navigate to Stock Trading Centre |
| **Test Case--4.5 Test Outcome** |
| Navigate to Stock Trading Centre |
| **Test Case--4.6 Remark** |
| Successful navigate to Stock Trading Centre Screen |

|  |
| --- |
| **Integration Test Plan** |
| **Test Case--5.0 Integration Testing-2** |
| Lead to Portfolio when Portfolio Button is clicked. |
| **Test Case--5.1 Description** |
| To test whether will be directed lead to the portfolio when clicked the portfolio button. |
| **Test Case--5.2 Diagram** |
| C:\Users\Wen Xi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_2017-04-10-08-56-15-377_android.gowenxi.fundfair.png**C:\Users\Wen Xi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\17837712_1169528246503861_656349357_o.png** |
| **Test Case--5.3 Test Data** |
| Portfolio button is clicked |
| **Test Case--5.4 Expected Result** |
| Direct lead to Portfolio Screen |
| **Test Case--5.5 Test Outcome** |
| Direct lead to Portfolio Screen |
| **Test Case--5.6 Remark** |
| Successful lead to Portfolio Screen |

|  |
| --- |
| **Integration Test Plan** |
| **Test Case--6.0 Integration Testing-3** |
| Lead to Help when Help Button is clicked. |
| **Test Case--6.1 Description** |
| To test whether will be directed lead to the help when clicked the Help button. |
| **Test Case--6.2 Diagram** |
| **C:\Users\Wen Xi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\17837712_1169528246503861_656349357_o.png** |
| **Test Case--6.3 Test Data** |
| Help button is clicked |
| **Test Case--6.4 Expected Result** |
| Direct lead to Help Screen |
| **Test Case--6.5 Test Outcome** |
| Direct lead to Help Screen |
| **Test Case--6.6 Remark** |
| Successful lead to Help Screen |

|  |
| --- |
| **Integration Test Plan** |
| **Test Case--7.0 Integration Testing-4** |
| Buy button in buy stock screen |
| **Test Case--7.1 Description** |
| To test whether the buy stock screen will able to displayed a pop out window to confirm the purchase and show the status of purchase when ‘Proceed’ button is clicked. |
| **Test Case--7.2 Diagram** |
|  |
| **Test Case--7.3 Test Data** |
| Buy button is clicked |
| **Test Case--7.4 Expected Result** |
| A pop out window to confirm the purchase will be displayed and status will be shown. |
| **Test Case--7.5 Test Outcome** |
| Pop out window displayed and status updated based on the condition. |
| **Test Case--7.6 Remark** |
| Buy Button able to work as expected. |

|  |
| --- |
| **Integration Test Plan** |
| **Test Case--8.0 Integration Testing-5** |
| Sell button in sell stock screen |
| **Test Case--8.1 Description** |
| To test whether the sell stock screen will able to display a pop out window for confirmation and status based on the condition when ‘Proceed’ button is clicked. |
| **Test Case--8.2 Diagram** |
|  |
| **Test Case--8.3 Test Data** |
| Sell button is clicked |
| **Test Case--8.4 Expected Result** |
| A pop out window will be displayed and after that ‘Transaction Successful’ will be displayed if the transaction is valid. |
| **Test Case--8.5 Test Outcome** |
| ‘Transaction Successful’ displayed when the ‘Proceed’ button is clicked |
| **Test Case--8.6 Remark** |
| Sell Button able to work as expected. |

### ystem Test Plan

|  |
| --- |
| **System Test Plan** |
| **Test Case-9.0 System Testing-1** |
| View Stock |
| **Test Case--9.1 Description** |
| To test whether the user will be able to view stock details when clicked on the stock button within 3 second (Non-functional Performance Constraint). |
| **Test Case--9.2 Diagram** |
| C:\Users\Wen Xi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\17858756_1169566399833379_1715105772_o.pngC:\Users\Wen Xi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\17858876_1169566476500038_10333012_o.png |
| **Test Case-9.3 Test Data** |
| |  |  | | --- | --- | | **Test number** | **Stock Name** | | 1 | Alibaba | | 2 | Apple | |
| **Test Case-9.4 Expected Result** |
| Stock Details of Alibaba and Apple will be displayed |
| **Test Case-9.5 Test Outcome** |
| Stock Details of Alibaba and Apple will be displayed |
| **Test Case-9.6 Remark** |
| Stock Details displayed correctly. |

|  |
| --- |
| **System Test Plan** |
| **Test Case-10.0 System Testing-2** |
| Buy Stock |
| **Test Case-10.1 Description** |
| To test whether the buy stock screen is able to display success message and will add the stock into user portfolio when ‘Buy’ button is clicked. |
| **Test Case-10.2 Diagram** |
|  |
| **Test Case-10.3 Test Data** |
| Select Amount of Lot and Buy button is clicked |
| **Test Case-10.4 Expected Result** |
| ‘Transaction successful’ displayed, the stock will be added to portfolio and the cash will be deducted. |
| **Test Case-10.5 Test Outcome** |
| ‘Transaction successful’ is displayed, the stock is added to portfolio and the cash has been deducted. |
| **Test Case-10.6 Remark** |
| Buy Stock able to works well. |

# Test Analysis Report

This is the second iteration for developing FundFair mobile application. Every use case of FundFair application able to achieve the functional requirement. After the unit testing, integration testing and system testing, the test results are good because the test result are same as our expected results.

# Conclusion

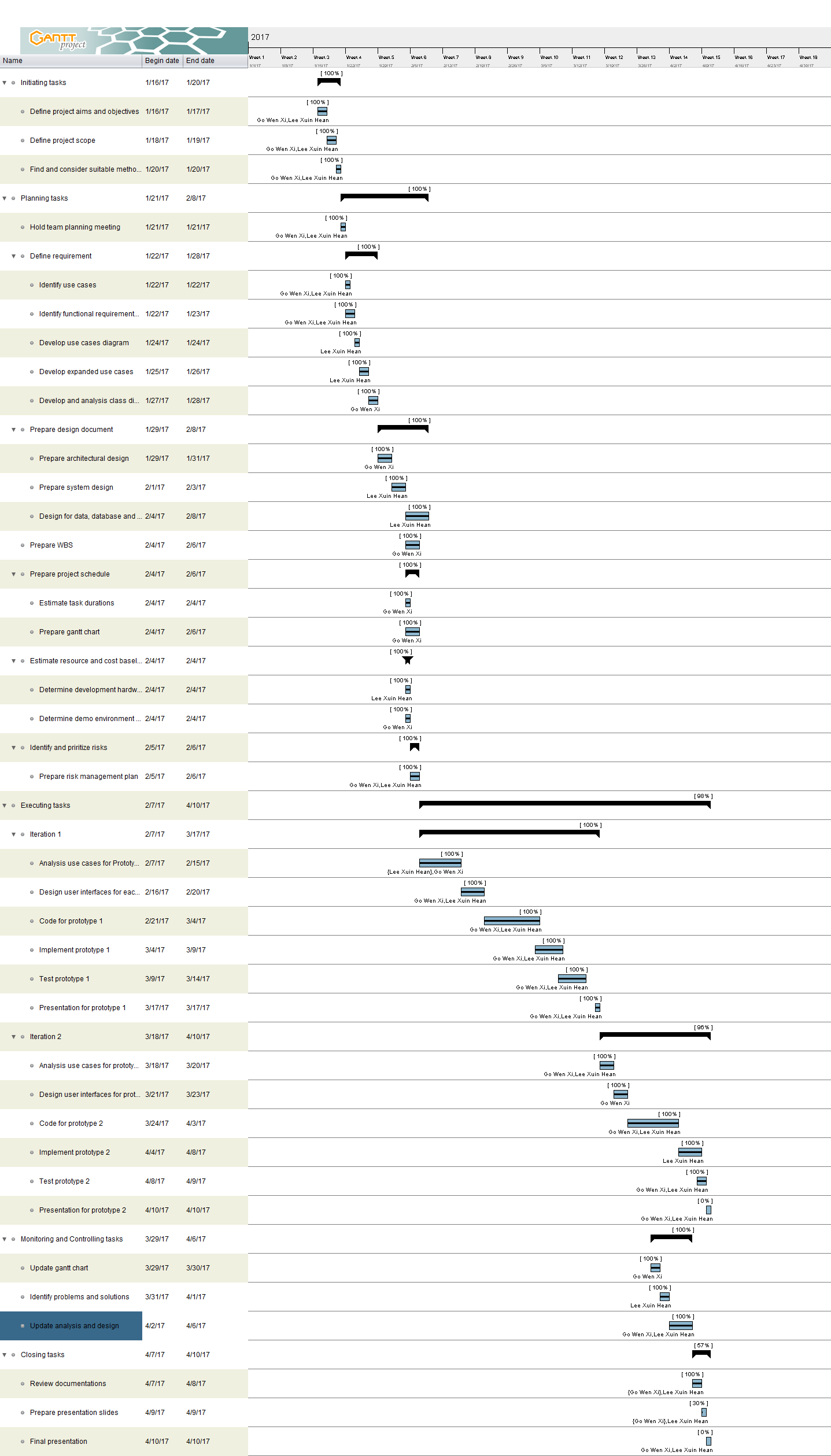
We manage to develop our Fund Fair mobile application prototype on time which are stated inside the Gantt Chart. Our aim to develop FundFair is to let novice learn the way to make stock investment and FundFair is a platform that provides stepping stone to let them experience, so our FundFair application is not complicated as others existing stock investment application.

We had make some differences for our FundFair application, at first we decided to make into multiplayer game and the results will be based on fictitious stock and randomize fictitious news. After our group discussions, we found that is not necessary to make a multiplayer game and this is only a prototype, the key feature for stock investment is learn how to see the information and know how to buy and sell stock. So, we make some changes by removing the multiplayer and fictitious news, that’s why we unable to achieved 3 objectives which are develop a stock game based on fictitious stock that will generate fictitious, allow 5 players to participate in a game and show the winner based on net profit. Since we did not make it as a multiplayer game so these objectives unable to achieved. What we had achieved our objective is providing a novice friendly game user interface, show the net profit that they earned and we are using fictitious stock information. Another difference is we added one more use case which is “Access Help”, as we think it is very important to let user know some key knowledge about stock investment and the meaning of the keywords that show in the Stock information.

There are few things went wrong, one of it is that we did not plan properly for our FundFair application before developing, so in the end we change to single player game, these affected our project objectives. Next thing that went wrong is the non-functional requirement, we did not understand the purpose of this, that is the reason why we put the non-functional requirement standard very high as unable to be tested, example like system should be able to hold a 50000 user records initially and a record should be fully available on the system for at least 5 years. Another thing that did wrong is too late to start developing our application, as we are new in doing mobile application so we faced some problems during developing especially in iteration 1 login and register, we unable to connect with our database end up we unable to finish another use case which is view company and stock details during iteration 1.

Things that we did right is our test result can match with our expected results, and another one is we add in the “access help” use case. Providing hints is very important to guide novice and provide them knowledge about it, luckily at the end we manage to add this use case inside our prototype.

# Updated Gantt Chart



# References

Google Firebase. (2017, March 10). *Firebase Authentication*. Retrieved from Firebase: https://firebase.google.com/docs/auth/