**Wells Fargo Mobile Banking**

Exercise D, Group 9

Prepared for:

Dr. Harry S Delugach

CS 650

April 19, 2017

Prepared by:

Adithi Madhwesh

Nethravathi Sathnur Nagaraj

Mohammad Shojaeshafiei

Table of Contents

**Introduction3**

**Reviews Conducted**3

**Test Cases**3

Test Cases3

Transfer Funds3

Bill Payment4

View Checking Transactions5

View Saving Transactions5

**Traceability Table**6

**Function Point Estimate**12

Complexity Multiplier12

Function Point Estimate Table13

Discussion of Function Point13

User Input13

User Output13

User Inquiries13

Files13

External Interfaces13

**Evaluation of Our Exercise14**

**Evaluation of Overall Set of Exercises**14

**Appendix15**

**Appendix: Team member contributions15**

Team meetings**15**

**References16**

**Introduction**

We were assigned the set of documents dealing with the Wells Fargo Banking application. Wells Fargo is a bank, that provides amenities such as transferring funds among account, paying bills, viewing bank statements, and activities on the account. The phone application, allows the user to partake in these amenities anywhere. Within this exercise there are evaluations of the originating documents, as well as this document, test cases, a revised traceability table, a function point estimate, and an appendix. The evaluations of the originating documents were made with respect to usefulness and understandability. The evaluation of our document was made with respect to completeness, correctness, and quality. The test cases were based on the requirements from the use case specifications in the originating document Exercise A, and the sequence diagrams from the originating documents of Exercise B. The traceability table has been revised to include a trace from and to all the test cases. The Function point estimate includes how we determined our complexity multiplier, and justifications for our weights in the function point estimate, as well as how we determined other values. The appendix gives a breakdown of the work load per group member, as well as information on our group meetings in this deliverable, we derived the test cases based on the use case specification of the first deliverable to test if the app works properly. It also contains the traceability table that shows how the artifacts in the test cases trace back to the use case specifications and sequence diagrams. Additionally, we have captured the Function Point as a unit of measurement to address the amount of functionality our specific product provides to a user.

**Reviews conducted**

After thorough and careful consideration in exercise A deliverable, we realized the quality for some of the use cases were not mentioned in a correct manner and consequently deliverables B and have not addressed these shortcomings. For that reason, some of the use case specifications were incomplete.

**Test Cases**

**Test Case Name: Transfer Funds**

**Description:** This test case covers the mainstream of the transfer money which is mentioned in the first use case. The goal is to test transfer money from one account to another one.

Setup:

1. The user’s account balance should be adequate to transfer money from the account.
2. The user should have two accounts; one as a source and one for destination.
3. The application is already installed and the user has logged in and is on the homepage.
4. The user has internet connection.

|  |  |  |
| --- | --- | --- |
| Steps | Action | Expected Output |
| 1 | Select Transfer money | Pop up a list of approved accounts |
| 2 | User taps on the account to transfer money from, for example Wells Fargo | Displays the details (To, Frequency, Date and Amount) which need to be filled by user. |
| 3 | User enters the details (To, Frequency, Date and Amount) to transfer for example $20 from Wells Fargo Checking to Redstone Checking and taps on Continue button. | Transfer Money Page presents all the detailed entered by user and the “Continue” and “Cancel” buttons will be displayed |
| 4 | User approves the transaction | Displays the “transaction confirmation” page to the user including for example:  From: Everyday checking  To: Redstone FCU  Frequency: Once  Send On: 04/15/17  Delivery:2 business days  Fee: $ 0.00  Expected Delivery: 04/19/17  Amount: $ 20.00  Verification code: 208963469 |
| 5.1 | Verify transfer was successful | The balance is $ 20.00 less |
| 5.2 | Transfer Failure | The balance is the same |

**Test Case Name: Bill Payment**

**Description:** This test case test’s whether the user can make a bill payment to recipients account as per the second use case specification mentioned in the first deliverable. The goal of this test case is to make a payment from user’s account to the recipients account.

Setup:

1. The Application is already installed and the user has login credentials and has logged in to the system and is on the home page of the application.
2. The User should have Recipients account details and User should have a checking account and the account balance should be adequate to pay the bill from his/her account to Recipient account.
3. The user has internet connection.

Exceptions: The touch screen is not sensitive enough to capture the user's input

|  |  |  |
| --- | --- | --- |
| Step | Test Steps | Expected output |
| 1 | User Taps on “Bill Payment” option | Application should display a screen with “Make Payment” and “Manage Bill Payment” options. |
| 2 | User Taps on “Make Payment”  option | Application should display a screen with details (Frequency, Date and Amount) which needs to be filled by the user and “Pay To” button |
| 3 | User Taps on “Pay To” button | Application should display these options  - Add a new account  - select from existing account |
| 4 | User Taps on “select from existing account” option | Application should display a list of user’s existing external accounts |
| 5 | User selects the account from the displayed list to make the payment and enters the required inputs on the screen | Application displays “Make Payment” or “Cancel” button on the screen for the user to select |
| 6 | User Taps on “Make payment” button | Application should display payment was successful with payment details and confirmation number. |

Note: The Use case specification of first deliverable is completely different form the existing mobile application of Wells Fargo bank and these errors have not been notified in second or third deliverable evaluation. Checking account transfer has not been mentioned in the pre- condition but has been written in the use case specification. We have developed test cases based on the use case specification provided in the first deliverable.

**Test Case: View Transactions; Checking**

**Description:** For a banking application, the ability of the user to check their transactions is crucial, especially if they have lost their bank card. This test case check to see if the user is able to view the transactions made on both their checking account. There is a separate test case for the savings account. This test case will be completely successful when the user can is brought to the screen that allows them to view all of the transactions that have been made on their checking account. If no transactions have been made, then a message will appear where the transactions should be stating “No transactions have been recorded for this account.”

Set-Up:

1. The application should have been successfully installed onto the testing device.
2. The user should have an account on the application.
3. The user should have internet connection.

|  |  |  |
| --- | --- | --- |
| Step | Input Data | Expected Output |
| 1 | Tap on the checking account icon. | Either a list of saved transactions for made for the checking account, or a message stating “No transactions have been recorded for this account.” |

**Test Case: View Transactions; Saving**

**Description:** For a banking application, the ability of the user to check their transactions is crucial, especially if they have lost their bank card. This test case check to see if the user is able to view the transactions made on their savings account. There is a separate test case for the checking account. This test case will be completely successful when the user can is brought to the screen that allows them to view all of the transactions that have been made on their saving account. If no transactions have been made, then a message will appear where the transactions should be stating “No transactions have been recorded for this account.”

Set-Up:

1. The application should have been successfully installed onto the testing device.
2. The user should have an account on the application.
3. The user should have internet connection.

|  |  |  |
| --- | --- | --- |
| Step | Input Data | Expected Output |
| 1 | Tap on the savings account icon. | Either a list of saved transactions for made for the savings account, or a message stating “No transactions have been recorded for this account.” |

**Traceability Table**

This Traceability Table is grouped by kind of item, and each kind of item is then alphabetized.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item No. | Traceable Item | Kind of Item | Item traced from | Item traced to |
| 1 | accountBalance | Attribute | Account class | accountBalance attribute in either checking or saving subclass |
| 2 | accountNumber | Attribute | Account class | accountNumber attribute in Account class |
| 3 | accountNumber | Attribute | Account class | accountNumber attribute in Account class |
| 4 | accountType | Attribute | Account class | accountType attribute in Account class, select account type event in sequence diagram |
| 5 | accountType | Attribute | Account class | accountType attribute in Account class |
| 6 | Address | Attribute | Implicit requirement for Customer in use case diagram | Customer class |
| 7 | m\_branchName | Attribute | Bank class | m\_branchName attribute in Bank class |
| 8 | m\_password | Attribute | Customer class, Login use case | Customer class, Authenticate class |
| 9 | m\_password | Attribute | Implicit requirement for Customer in use case diagram | Customer class, authenticate method in Authenticator class |
| 10 | m\_routingNumber | Attribute | Bank class | m\_routingNumber attribute in Bank class |
| 11 | m\_userID | Attribute | Customer class, Login Use Case | Customer class, Authenticate class |
| 12 | m\_userID | Attribute | Implicit requirement for Customer in use case diagram | Customer class, authenticate method in Authenticator class |
| 13 | phoneNumber | Attribute | Implicit requirement for Customer in use case diagram | Customer class |
| 14 | Account | Class | Implicit requirement, account in use case diagram | Account class, sequence diagram |
| 15 | Authenticator | Class | Authenticate in use case diagram | Authenticator class |
| 16 | Bank | Class | Actor in use case diagram | Bank class |
| 17 | Checkings | Class | Checking account in use case diagram | Checking subclass of Account class |
| 18 | Customer | Class | “Customer” actor in use case diagram | Customer class, Customer in sequence diagram |
| 19 | Savings | Class | Saving account in use case diagram | Savings subclass of Account class |
| 20 | Acknowledgement | Message | Sequence Diagram | Database class in sequence diagram |
| 21 | Display Existing Account | Message | Bank class | Bank class in sequence diagram, Bank class |
| 22 | Display Updated Message | Message | Sequence Diagram | Account class in sequence diagram, Account class |
| 23 | Enter Payment Details | Message | Customer class | Customer class in sequence diagram |
| 24 | Initiate Deposit | Message | Deposit checks use case | Customer class, Customer class in sequence diagram |
| 25 | Send Check Details | Message | Pay Bills use case | Account class, Account class in sequence diagram |
| 26 | Update Database | Message | Sequence Diagram | Database class in sequence diagram |
| 27 | authenticate | Method | Authenticator class, Authenticate use case | Authenticate method in Authenticate class |
| 28 | depositCheck | Method | deposit check in use case diagram | depositCheck method in Customer class |
| 29 | login | Method | login in use case diagram | login method in Customer class |
| 30 | logout | Method | logout in use case diagram | logout method in Customer class |
| 31 | manageAlerts | Method | manage alerts in use case diagram | manageAlerts in Customer class |
| 32 | payBills | Method | pay bills in use case diagram | payBills method in Customer class, payBills event in sequence diagram |
| 33 | receiveCash | Method | implicit requirement for Bank in use case | receiveCash method in Bank class |
| 34 | register | Method | initial requirement for customer | register method in Customer class |
| 35 | selectAccountType | Method | Account class, Account Summary use case | selectAccountType method in Customer class, Select Account Type message in sequence diagram |
| 36 | selectedAccDetails | Method | Account summary Use Case | selectedAccDetails in Account class |
| 37 | showTransactions | Method | view statements in use case diagram | showTransactions in account subclasses |
| 38 | transferCash | Method | implicit requirement for Bank in use case | transferCash method in Bank class |
| 39 | transferFunds | Method | transfer funds in use case diagram | transferFunds method in Customer class, transferFunds sequence diagram |
| 40 | updateAccountInfo | Method | implicit requirement for Bank in use case | updateAccountInfo method in Bank class |
| 41 | updateBalance | Method | implicit requirement for Bank in use case | updateBalance method in Bank class |
| 42 | viewAccountSummary | Method | account summary in use case diagram | viewAccountSummary in class diagram, Display Account summary in sequence diagram |
| 43 | viewAccountSummary | Method | Account class | viewAccountSummary method in Account class |
| 44 | viewAccountSummary | Method | view statements in use case diagram | viewAccountSummary method in account subclasses |
| 45 | viewRecentTransactions | Method | recent transactions in use case diagram | showTransactions in Account class |
| 46 | viewRecentTransactions | Method | View statements use case | viewRecentTransactions in Account class |
| 47 | Alerts | State | Manage alert use case | manageAlerts method in Customer class |
| 48 | Retrieve Checking Balance | State | Account summary use case | accountBalance attribute in checking class |
| 49 | Retrieve Saving Balance | State | Account summary use case | accountBalance attribute in saving class |
| 50 | Retrieving Customer Info | State | Manage profile use case | m\_userID, m\_userName, m\_address, m\_phoneNumber attributes in Customer class |
| 51 | Bill Payment | Test Case | Bill Payment Use Case Specification, Bill Payment Sequence Diagram |  |
| 52 | Transfer Money | Test Case | Transfer Funds Use Case Specification, Transfer Funds Sequence Diagram |  |
| 53 | View Checking Transactions | Test Case | View Statements Use Case Specification, View Statements Sequence Diagram |  |
| 54 | View Savings Transactions | Test Case | View Statements Use Case Specification, View Statements Sequence Diagram |  |
| 55 | Account Summary | Use Case | Account Summary Use Case | viewAccountSummary in Account class, send Account details event in sequence diagram |
| 56 | Authenticate | Use Case | Authenticate Use Case | Authenticate method in Authenticate class, authenticate event in sequence diagram |
| 57 | Checking Account | Use Case | Recent Transactions Use Case | Checking class |
| 58 | Checking Total | Use Case | Account Summary Use Case | viewAccountSummary in Account class (superclass) |
| 59 | Checks | Use Case | View Statements Use Case | Checks use case |
| 60 | Deposit | Use Case | View Statements Use Case | deposit method in Customer class |
| 61 | Deposit Checks | Use Case | Deposit Checks Use Case | depositChecks method in Customer class, deposit check event in sequence diagram |
| 62 | External Account | Use Case | Transfer Funds Use Case | External account use case |
| 63 | Fingerprint Login | Use Case | Login Use Case | login method in Customer class |
| 64 | Login | Use Case | Login Use Case | Login method in Authenticator class, login event in sequence diagram |
| 65 | Logout | Use Case | Logout Use Case | logout method in Customer class |
| 66 | Manage Alert | Use Case | Manage Alert Use Case | manageAlerts method in Customer class |
| 67 | Manage Profile | Use Case | Manage Profile Use Case | Account class, updateAccountInfo in Bank class |
| 68 | Pay Bills | Use Case | Initial Requirement, Use Case Diagram | payBills method in Customer class, Pay Bills event in sequence diagram, Bill Payment Test Case |
| 69 | Recent Transactions | Use Case | Recent Transactions Use Case | viewrecenttransactions in Account class |
| 70 | Saving Account | Use Case | Recent Transactions Use Case | Savings class |
| 71 | Saving Total | Use Case | Account Summary Use Case | viewAccountSummary in Account class (superclass) |
| 72 | Transfer Funds | Use Case | Initial Requirement, Use Case Diagram | transferFunds in Customer class, transferFunds event in sequence diagram, Transfer Money Test Case |
| 73 | Update Info | Use Case | Manage Profile Use Case | Bank class |
| 74 | View Statements | Use Case | Initial Requirement, Use Case Diagram | viewRecentTransactions method in Account class, View Checking Transactions Test Case, View Savings Transactions Test Case |
| 75 | Withdrawal | Use Case | View Statements Use Case | Withdrawal use case |

**Function Point Estimate:**

**Complexity Multiplier [1]**

|  |  |  |  |
| --- | --- | --- | --- |
| Generic System Characteristics | | Value\* | Description |
| 1 | Data Communication | 5 | There are multiple different communication facilities, such as the database, the user, the administrator, etc. so there is a strong influence throughout. |
| 2 | Distributed Data Processing | 1 | Since this application will try to access a single database that hold all the information, there will need to be some type of distributed data processing, so there is an incidental influence. |
| 3 | Performance | 5 | The application requires quick response time and throughput, so there is a strong influence throughout. |
| 4 | Heavily Used Configuration | 2 | The phone’s controls and touchscreen are used by the user to interact with the application, so there is a moderate influence. |
| 5 | Transaction Rate | 3 | Transactions are executed at a moderate rate, so there is an average influence. |
| 6 | Online Data Entry | 1 | The application mirrors an online version of the application, so there is an incidental influence. |
| 7 | End-User Efficiency | 5 | The application must provide value to the end user, therefore efficiency is very important, so there is a strong influence throughout. |
| 8 | Online Update | 5 | The influence of internal logical files would be high, since banking requires logic, so there is a strong influence throughout. |
| 9 | Complex Processing | 5 | Since this is a banking application, there will be mathematical processing for actions such as transferring funds, bill payment, interest, etc. so, there is a strong influence throughout. |
| 10 | Reusability | 3 | This application provides many reusability features, so there is an average influence. |
| 11 | Installation Ease | 5 | There is a high influence on making conversion and installation easy for the end user, so there is a strong influence throughout. |
| 12 | Operational Ease | 5 | There is a high influence on making features such as start-up, back-up, and recovery procedure easy and automated for the end user, so there is a strong influence throughout. |
| 13 | Multiple Sites | 3 | The application should be able to be installed at multiple sites, since users have several types of mobile devices, and banking applications will also be able to allow the user to make payments for things like online purchases, bill payment, etc. so there is an average influence. |
| 14 | Facilitate Change | 3 | Since change is inevitable, the application that is developed should support change, so there is an average influence. |
| Total Degree of Influence(Fi) | | 51 | |
| Value Adjustment Factor\*\* | | 1.16 | |
| \*The value is ranged from 0-5, 0 indicating not present or no influence, and 5 indicating a strong influence throughout.  \*\*The Value Adjustment Factor was found using the following formula: (0.65+(0.01(Fi))) | | | |

**Function Point Estimate Table[2]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Weighing Factor | | |  |
| Measurement parameter | Count | Simple | Average | Complex | FP |
| Number of user inputs | 28 |  | 4 |  | 112 |
| Number of user outputs | 25 |  |  | 7 | 175 |
| Number of user inquiries | 3 |  | 4 |  | 12 |
| Number of files | 0 |  |  |  | 0 |
| Number of external interfaces | 4 |  | 7 |  | 28 |
| Count-total |  | | | | 327 |
| Complexity multiplier | 0.65+0.01(Fi) = 0.65+0.01(51) | | | | 1.16 |
| Function points | 327\*1.16 | | | | 379.32 |

Assuming the android version of the mobile application. The language chosen is Java and

one function point = 53 LOC,

Estimated LOC=373.52\*53 ~ 19796

**Discussion of Function Point Estimate:**

As per the discussion with Professor Dr. Delugach, the following estimates were made and the values of the weighting factor are derived from the table from the Software engineering slides.

**User inputs:** user inputs have been taken based on class diagram and data dictionary from previous deliverables. On estimating user’s inputs were 28.

We chose the average weighting factor because we feel that the user inputs fall between simple and complex cases.  The user inputs like username or password have an internal check for the size of the inputs. Tapping on various options seems to be one input for the user. As there are several inputs provided to each function of the application as parameters we chose it to be average.

**User outputs:** On an average most of the user inputs have common outputs and so the count is 25. The weighting factor is taken complex with respect to this application considering database operations, formatting of the output and several parameters to display.

**User inquiries:** The possible user inquiry in case of this application is 3, This is because user inquiries retrieve details from the database such as loading saved accounts to pay bills and managing personal account details . We chose average weighting factor because we feel that the user inquiries fall between simple and complex cases considering the connection of database server and getting the records from the database.

**Files:** We have not counted on files as there are no files considered in this application.

**External interfaces**: The estimated external interfaces used for this application are Database, User, Administrator and camera device. So, the estimated count is 4. We chose average weighting factor because internal bank transfers use an internal bank database which is simple and for the external bank transfers uses the connection to the external payment gateway and several operations which are complex.

The estimate has been calculated by calculating the most number of possible entries for each parameter with respect to the provided use case diagram. For each parameter, the highest factor of the maximum number of entries and lines of code has been considered for estimating the function point estimate. Hence, we consider it to be a fairly reasonable attempt to calculate the estimate.

**Evaluation of Overall Set of Exercises**

Upon writing the test cases for the 3 use case specifications, our group discovered that the specifications for the application, did not match the flow of command of the application the requirements were based on. Moreover, we realized these differences have not mentioned in exercises B and C that provides various ambiguous results in the next deliverables.  Some steps in use case “Transfer Fund” does not match the transfer fund in the real world actual application. For instance, the user in step 1 selects Transfer and pay then from drop down list they choose the appropriate option which is Transfer Money. In the next step, it is also mentioned that application provides the list of accounts to the user but the fact is that the procedure is completely different and the user is supposed to choose the source and the destination account in two separate procedures along with filling some other information such as Date, amount, Frequency, etc. in the transfer money page. Moreover, in order to approve the transaction of Transfer Money, it is necessary the application require the user for at least twosteps confirmation which the second one is for the user the review the information and check if they are correct the approve it, but the use case in deliverable A just mentioned one step confirmation. As a group, and with the help of Dr. Delugach, we decided to use the use case specifications and the sequence diagrams from the previous deliverables to form our test cases. In some of the use case specifications, an action would have been described, but the precondition for that action would not have been stated; for example, in Exercise A, in the “Bill Payment” use case specification, step 8 uses a checking account, but the preconditions for that use case specification do not require that the user have a checking account. While a paragraph explaining what exactly is going on in each sequence diagram would have helped in our understanding of the sequence diagrams; they were helpful in reinforcing the what was stated in the use case specifications. The class diagrams, the traceability tables, state diagrams, and the data dictionaries were useful in the determination of our function point estimate. If there had been a paragraph explaining the organization of the class diagrams and the traceability tables, that would have aided us in understanding them. If there had been a paragraph describing the state diagrams, that would have also been helpful. The traceability tables of the Exercise B and C were both useful and understandable in developing our own traceability table.

*Understandability*: Use Case diagram and consequently use case specifications are not clear and it is hard to understand the exact functionality of them in each step.

*Usefulness:* In fact the use case diagram should provide the necessary features of the system and helped us in understanding how the user interacts with the system. It could be much better that the available ones to be more useful.

**Evaluation of Our Exercise**

In this deliverable, we were asked to provide a test case for each use case specification in exercise A, which we have done. We were asked to revise the traceability table, which we have done. We were asked to provide a function point estimate, which we have done. We were asked to evaluate the overall set of exercises, which we have done. Finally, we were asked to provide an appendix, which we have done. Based on the previous claims, our deliverable is complete. We each reviewed and analyzed the work completed by each group member to make sure no step would be missing, and to make sure all the work was consistent. We even consulted the professor upon completing our first draft of the deliverable. Based on the claims, we strived to deliver a deliverable that was correct. The traceability table is organized based on kind of item, and then alphabetized to make looking items up in the table more efficient. We have numbered out pages, and provided a description or an example where necessary. Based on these claims, the quality of our deliverable is high. Therefore, we as a group have delivered on a standard of completeness, correctness, and quality.

**APPENDIX**

Team Member Contributions

1. Nethra Sathnur Nagaraj

* Test Case – Bill Payment
* Function Point Estimate
* Evaluation of Function point
* Evaluation of previous deliverable

1. Adithi Madhwesh

* Test Case – View Account Statements
* Traceability Table
* Complexity Multiplier
* Evaluation of previous deliverable

1. Mohammad Shojaeshfiei

* Test Case – Transfer Funds
* Evaluation of existing deliverable
* Traceability Table
* Final Report

Team Meetings:

* April 14th, 2017  
  Attendees: Adithi, Nethra, Mohammad

Location: Tech Hall at 11 AM – 12 :30 PM

Agenda: Introduction and allotment of work  
We met in Tech hall and introduced each other and discussed about the exercise and allotted the work for each team member.

* April 16th, 2017

Attendees: Adithi, Nethra, Mohammad

Location: Tech Hall at 11 AM – 3 PM

Agenda: Test Cases

We met in Tech hall and discussed about the test cases each one had worked on and had few doubts and decided to clarify with Dr. Delugach. Discussed on functional point.

* April 18th , 2017

Attendees: Adithi, Nethra, Mohammad

Location: Tech Hall at 2 PM – 2:30PM

Agenda: Functional point and test cases

We met in Tech hall and clarified doubts with Dr. Delugach on test cases and functional point. Made changes as per his suggestions.

* April 19th , 2017

Attendees: Adithi, Nethra, Mohammad

Location: Tech Hall at 2 PM – 3:30PM

Agenda: Final Document

We met in Tech hall and worked on final document

**References:**

[1] ["Value Adjustment Factor - Sharing the knowledge." Google Sites. N.p., n.d. Web. 19 Apr. 2017.](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/locks/ReentrantLoc)

[2] Admin. "Function Point Languages Table." QSM SLIM-Estimate. N.p., 11 Aug. 2016. Web. 19 Apr. 2017.