**Nicholas Webster**

**09/25/2017**

**ATM – Application:**

**MOST IMPORTANT NOTE FOR USING APPLICATION: Test data is generated for 4 users. Their user numbers are: 0, 1, 2, and 3. You will need these user numbers to login and transfer funds between users.**

**Overview of application features:**

This application attempts to emulate some of the basic functionality of a standard ATM. It allows users to log on, view their account balances, see a transaction log for each account and perform some basic functions such as Withdrawing Cash, Depositing Tender, and Transferring Funds.

**Application navigation and usage:**

**First Screen – The Logon Screen:**

Upon launch the application asks the user to enter their user number (valid user numbers are 0, 1, 2 and 3). The user is able to either click the input text box and type in their user number, or simply enter the user number with the on-screen digit pad.

If the user number was entered incorrectly and no user is found the application will alert the user and request the user attempt a different number.

**Second Screen – The User Information Screen:**

After logging in the user is greeted by the User Information Screen. This screen is divided into 4 sections.

1. The header section displaying details about the user who is logged onto the ATM
2. An Account Selection section that allows the user to see the current balance for each account and select the account they wish to work with.
3. The Transaction Log Table that shows all the transactions and transaction details for the selected account.
4. The Option Button section this section allows the user to click the process they wish the ATM to perform.

**The Withdraw Screen:**

If the user selects withdraw they will be taken to the withdraw screen. Here the user is asked to input the amount of cash they wish to withdraw. The amount to withdraw must be in increments of 20. Upon entering the desired amount, the user must click the enter button to perform the transaction.

**The Deposit Screen:**

If the user selects deposit they will be taken to the deposit screen. Here the user is asked to input the amount they wish to deposit. Upon clicking the enter button the transaction is completed and should show up in the transaction log.

**The Transfer Screens:**

If the user selects transfer they will be taken to a user selection screen. They must enter the user they wish to transfer funds to. Upon selecting the desired receiving user they will be taken to a second screen, where they are asked to enter the amount they wish to transfer and select the receiving user’s account that will receive the funds.

**Details about Code:**

The program is designed using the standard MVC model. The data model is as follows: The ATM has multiple users. Each user can have multiple accounts and each account has multiple transactions. The view model is split into multiple ui\_elements each making up a piece of each screen. The UIController then builds the required screen using these ui\_elements. There are two controllers, each designed to perform a specific function. The DataController holds all the users and their specific accounts, transactions and data. It can retrieve users and update/edit their information. The UIController handles the user interface elements for the program. It builds each screen using the individual ui\_elements and performs the proper actions depending on the user’s input.

Testing Plan:

I have thoroughly tested the program and am confident that each feature works as required. You can do the following in order to test and confirm the functionality.

**Test error message alerts and functionality:**

An error message is generated anytime the user input is not valid given the current situation. It will check to ensure that the user did not accidentally enter letters instead of numbers, decimal values instead of integers, negative numbers instead of positive ones, and confirm the existence of users and determine if sufficient funds exist to allow for the requested transaction.

To test for these messages simply attempt the following:

* Input a decimal, negative, or alphabetical value into the user logon or transfer user selection screen.
* Input a decimal, negative, alphabetical value or enter an integer that is not divisible by 20 or larger than the funds currently in the account on the withdraw screen.
* Input a negative or alphabetical on the deposit screen.
* Input a negative, alphabetical value or enter a decimal/integer value greater than the funds in the account on the transfer amount screen.
* Input the same number as the currently logged on user at the transaction user selection screen.

To test the systems, simply attempt each function and all buttons.

* The withdraw option deducts integers in increments of 20 from the account balance and logs the transaction.
* The deposit option adds integer or decimal amounts to the account balance and logs the transaction
* The transfer option allows you to select another user to transfer to, select the destination account attached to that user and input an integer or decimal value to transfer. The funds will be deducted from the user’s selected account and added to the receiving user’s selected account, creating a transaction for both the transferring and receiving accounts.
* The log off option takes you back to the login screen.
* You can log into a user that you transferred funds to and see the additional funds and transaction in the log. It states the amount, time of the transfer, and the number of the user who transferred money to the account.

**Limitations and possible future additions:**

I attempted to add as much functionality as possible to this application but it still has a few notable limitations:

* The data loaded to the DataController is all test data generated every time the application starts up. That’s why all the initial transactions have the same date and time. It would be nice to have an option to either serialize the data to a .json file or perhaps link it to a database.
* The user is unable to transfer funds between their own accounts. If the user attempts to enter their own user number on the transfer screen they are given an error message. It would be nice if instead it greyed out the account selected to transfer from but allowed the user to select another of their own accounts to transfer money to.
* The user is able to login simply by entering their user number. A password or PIN system would probably be a good idea to keep unwanted users from logging into other’s accounts.
* I built the layout to be more touch-screen friendly but never tested it with a touch device. Since most ATMs have touch screen expanding upon this feature would be an excellent option to increase functionality.
* From a code perspective there are several instances of suboptimal code. I notated each of these cases in great detail inside the code comments. I intend to finish up this project by correcting this suboptimal code, simplifying the design and uploading it to my GitHub to add to my professional portfolio.