Hassan Shahid - 10155809
Niv Ben Shabat - 20048720
Caleb Aikens - 10105630

Assignment 4: Back End Design Document

The structure of the Back Office we have created is very similar to that of our Front End. Our back end consists of a 2-layer architecture, along with a launcher and a shared classes package.

- **Business Layer (BL)** - Similar to the front end, this layer initializes the Back Office. It consists of one class:
    - **Core.java** - starts the back end, taking in the transaction file and making the appropriate actions according to it.
      Methods:
        - ➢ start() - initializes the system; reads in accounts file, transaction file and writes data according to the contents
        - ➢ handleDeposit(Transaction t) - Adds to the amount to deposit to the user's account balance. Takes one parameter of type Transaction. The parameter is one line of input of a transaction; it provides all the information needed to perform the action.
        - ➢ handleWithdraw(Transaction t) - Takes away the amount to withdraw from the user's account balance.
        - ➢ handleTransfer(Transaction t) - Takes away the amount to transfer from one user's account balance and adds it to the other user's account balance.
        - ➢ handleNew(Transaction t) - Creates new user in accounts file

        - ➢ handleDelete(Transaction t) - Deletes a user from the accounts file.
        - ➢ findUser(String accountNumber) - Finds a user in the accounts file. Takes 1 parameter of type string - the account number to find in the file.
- **Data Access Layer (DAL)** - this layer is responsible for data fetching and updating
    - **Data.java** - reads transaction and accounts files, parses lines, and writes to file.
        - ➢ readTransactionFile() - reads the transaction summary file. Returns a linked list of transactions
        - ➢ parseLineToTransaction(String line) - reads in a line, splits it, and returns a Transaction object with the line's info.
        - ➢ readAccountFile() - reads the valid accounts list file. Returns a linked list with all the users.
        - ➢ parseLineToUser(String line) - returns an object of type User; reads the info from the line.
        - ➢ writeUsers(LinkedList<User> users) - writes all users into the valid accounts file
        - ➢ setTransactionFilePath(String path) - Sets the path where the transaction file is stored
        - ➢ setMasterAccountFilePath(String path) - Sets the path where the accounts file is stored
- **Shared Classes** - shared data structures needed by the other layers
    - **Transaction.java** - defines a Transaction object; stores the transaction code, the account number, an amount (if

applicable), another account number (if applicable), and the name on the account.

    - **User.java** - defines a User object; stores the account number, current balance, and name of the user.

- **Launcher** - starts the Back Office; initializes the system to begin analyzing data. Handled by one class, containing the main method to start the system.

Visually, the Back Office looks like this

: