**C++ Assignment: The Banking Application**

**Vivian Lam**

Student No. 4210

[Vlam54@uwo.ca](mailto:Vlam54@uwo.ca)

October 2016

Computer Science 3307A - Object-Oriented Design & Analysis

University of Western Ontario, London ON

Canada N6A 3K7

**Table of Contents**

**Deliverable 1: Minimum Requirements**

**Deliverable 2: Enhancement Requirements**

**Deliverable 3: System code and executable**

**Deliverable 4: Scenarios**

**Deliverable 5: Implementation of the enhanced requirements**

**Deliverable 6: What I learned**

**Deliverable 1: Minimum Requirements**

* Ability to open/close accounts
* Give warning when not sufficient funds when withdrawing
* Ability to transfer between chequing and savings account
* Manager/customer Obtain account balances
* Give warning message to client on his/her “chequing” account will drop below the threshold of $1,000 – prior to him/her executing the operation. If s/he decides to go ahead despite the warning message, a charge of $2.00 is levied on the client for each such transaction. This levy is applied only once, when there is a crossover from above to below the threshold.
* Client is able to open multiple accounts
* The bank manager can display the account details of any given customer, or all customers, and obtain aggregate data on the funds in the bank, etc., categorised appropriately.
* Funds in the account transcend the duration of a particular user session with the bank.
* Maintenance person can login and turn on and off execution trace
* Execution trace printed to external file

**Deliverable 2: Enhancement Requirements**

* Implemented a login system
  + The system automatically knows what type of account (manager, customer, maintainer) from your account login ID

**Deliverable 3: System code and executable**

In the Debug folder open the “vleeambank.exe” to run the executable.

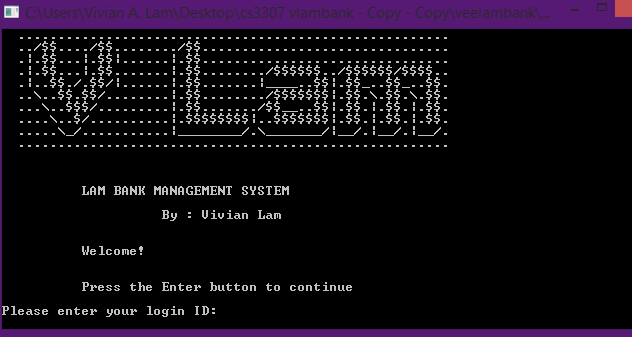
This application is for Windows 8.1

Please check the README before running and details regarding compiling

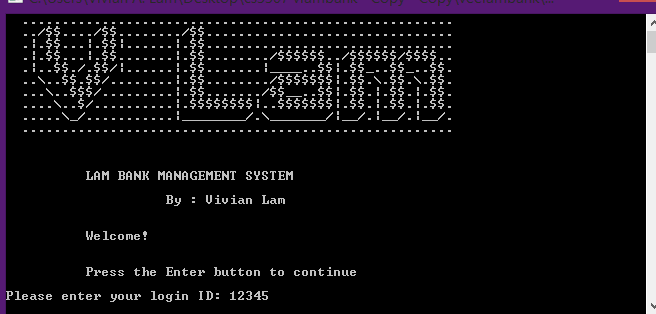
**Deliverable 4: Scenarios**

Scenarios (keep scrolling for screenshots):

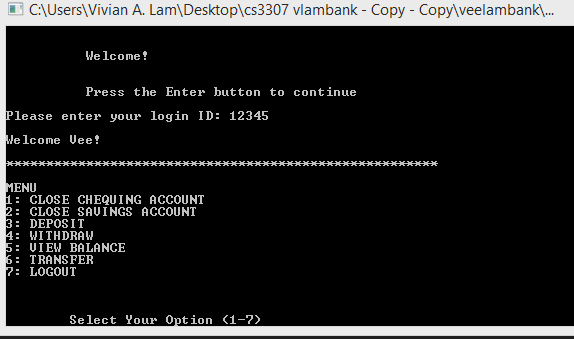
1. Client deposits funds
   * Client logs into the ATM.
   * Selects his/her Deposit account.
   * Checks the balance.
   * Deposits CAD 20.
   * System deposits the funds.
   * System displays the resultant balance.
2. Manager creates an account
   * Manager logins
   * Manager selects create an account
   * Enter in details
   * System creates a new account
   * System displays the new account data
3. Maintenance operator turns on execution trace and views the traced files
   * Maintenance operator logs in
   * Selects execution trace to be on
   * Print out execution trace
   * View the .txt file (note that it will only show the trace for this scenario because that’s when execution trace was turned on)
4. Manager deletes an account
   * Manager logs in
   * Manager selects delete an account
   * Enters in the account to be deleted
   * System deletes the account
   * System shows that the account no longer exists
5. Customer transfers funds
   * Customer logs in
   * Customer selects transfer funds
   * Customer selects which account to transfer from
   * Customer selects amount
   * System transfer funds
   * System displays the new account data
6. **Client deposits funds**

Upon loading 

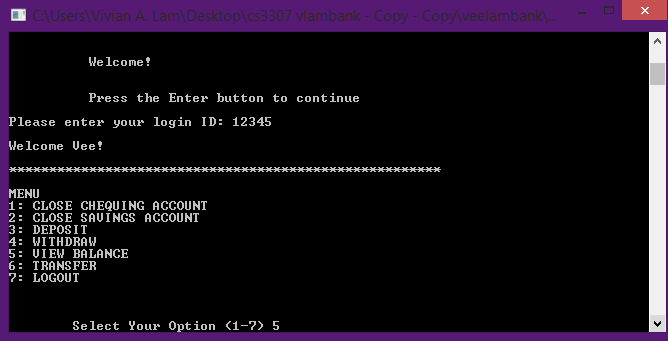
Client logs into the ATM.

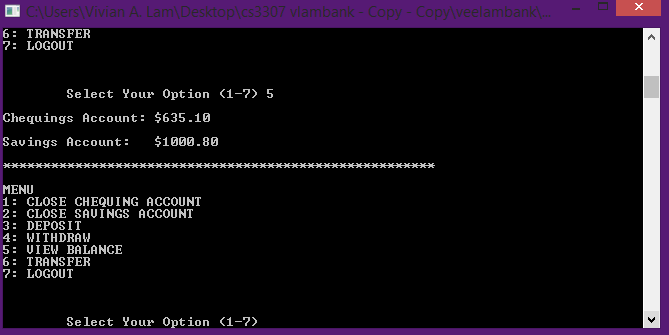


Selects his/her Deposit account.

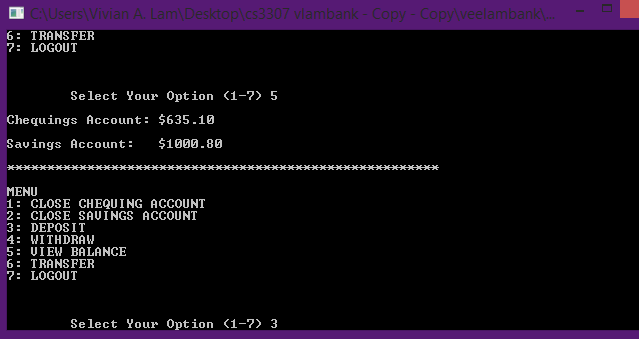


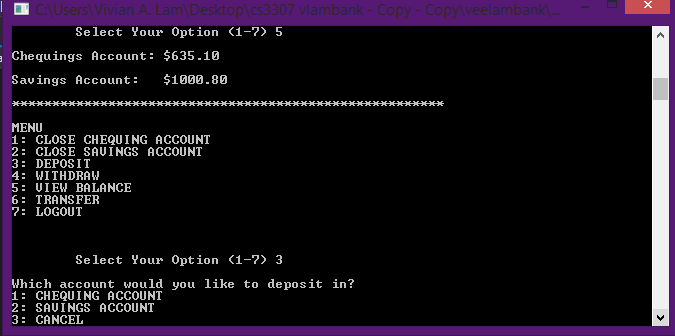
Checks the balance.

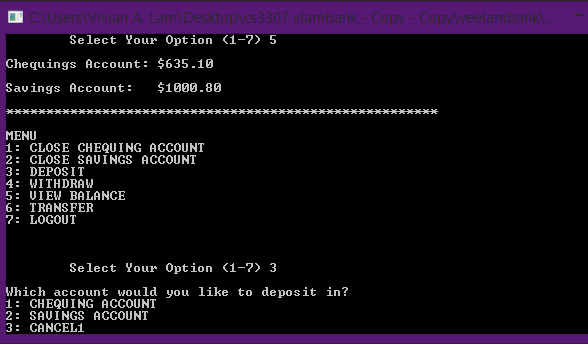


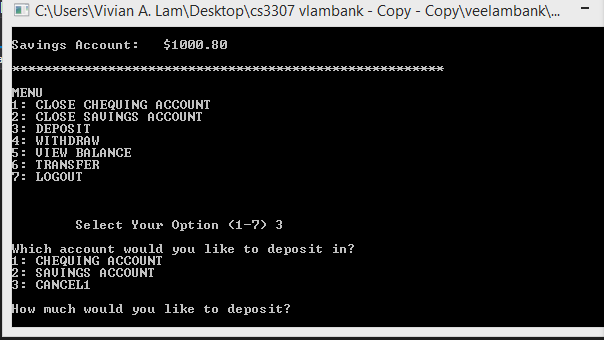


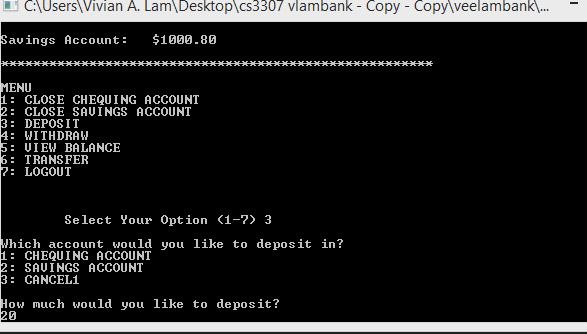
Deposits CAD 20.



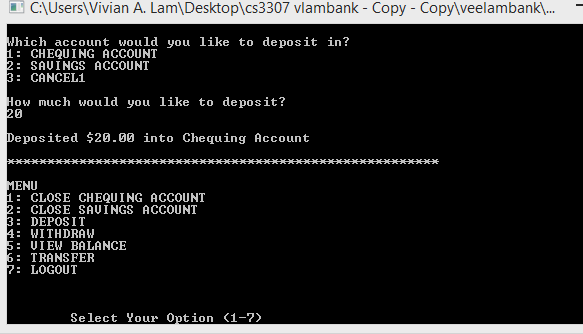




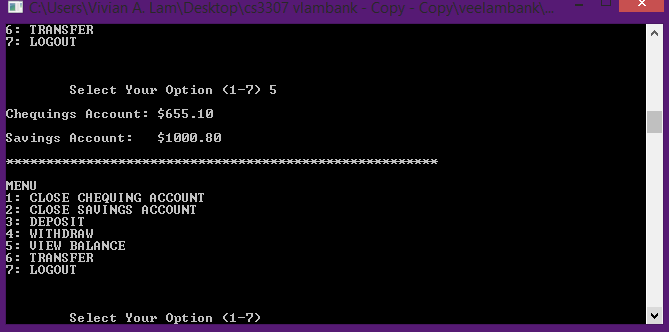




System deposits the funds.

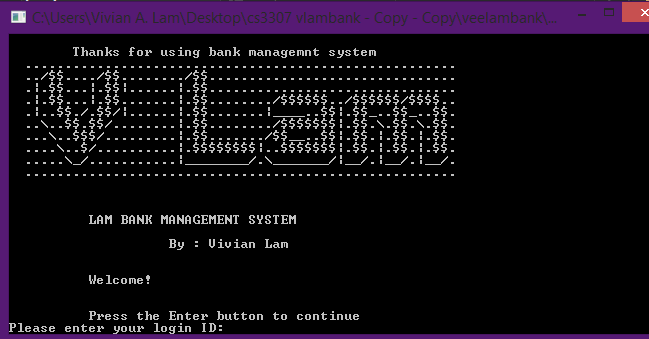


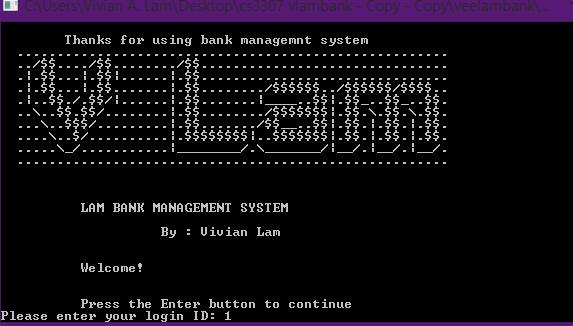
System displays the resultant balance. (select VIEW BALANCE from the menu)



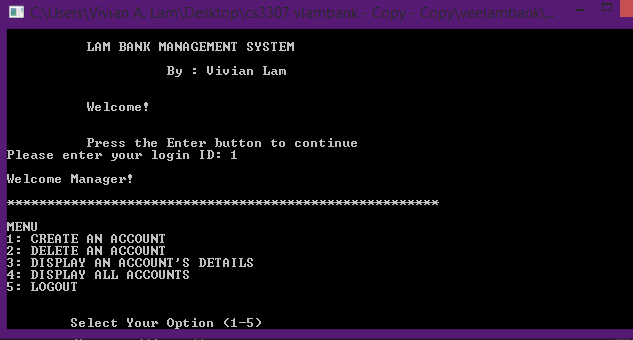
**2. Manager creates an account**

Manager logins



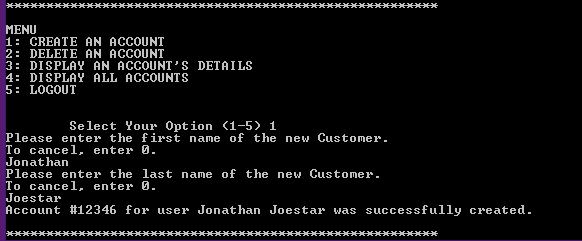


Manager selects create an account



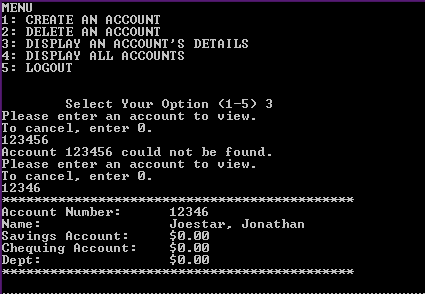


Enter in details: first and last name (in this example: Jonathan; Joestar. The account number is auto generated)



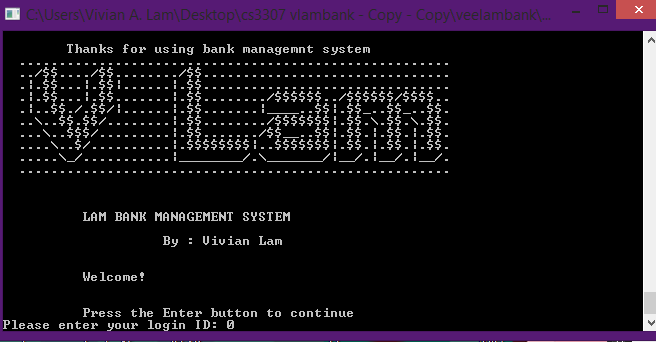
System creates a new account

System displays the new account data (select option 3 from main menu and then enter in the right account number. Below I have accidentally entered a non-existant account number, followed by the correct account number)



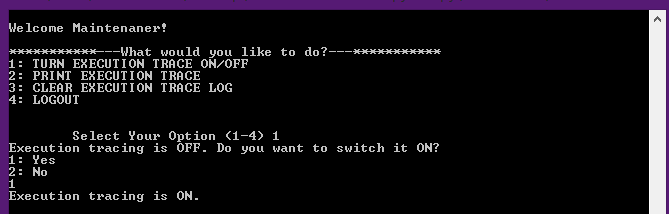
**3. Maintenance operator turns on execution trace and views the traced files**

Maintenance operator logs in

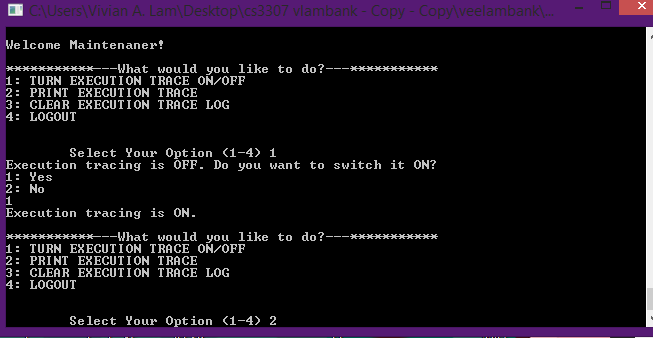




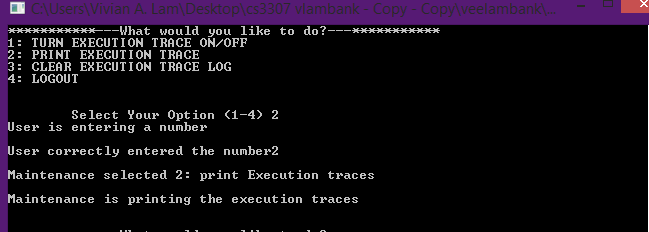
Selects execution trace to be on (select option 1 from main menu, then select 1 again to turn it on)



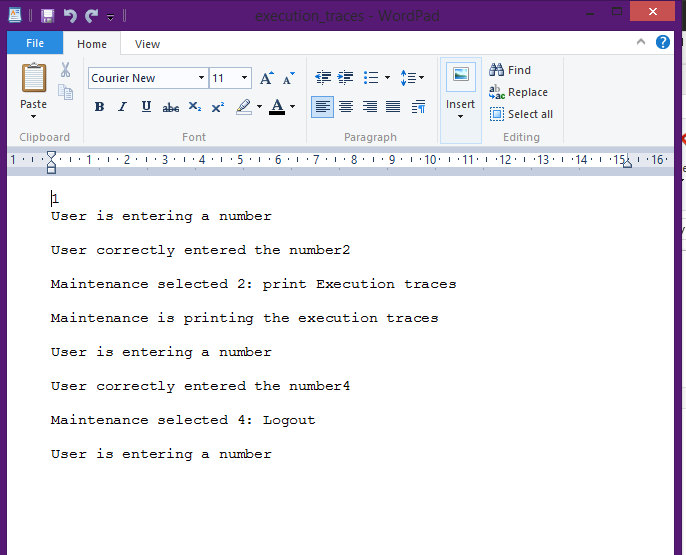
Print out execution trace (option 2 from main menu)



Note that it also shows on screen

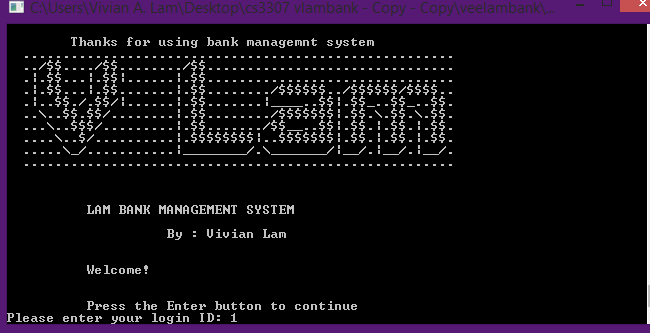


View the .txt file (note that it will only show the trace for this scenario because that’s when execution trace was turned on)

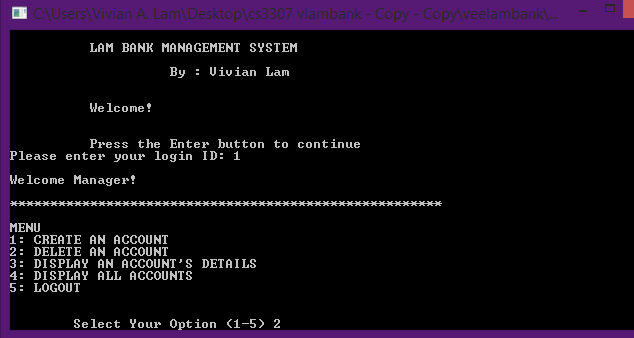


**4. Manager deletes an account**

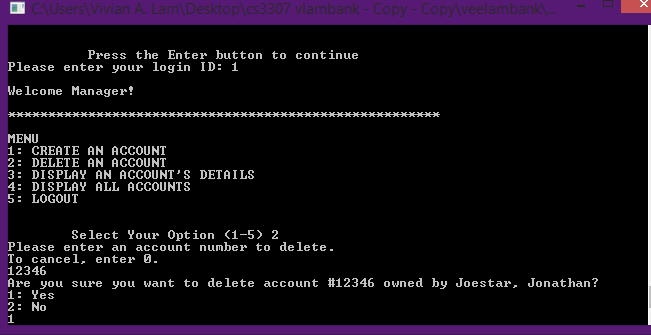
Manager logs in



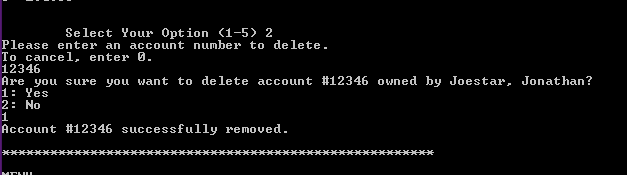
Manager selects delete an account



Enters in the account to be deleted (and enters 1 to confirm)



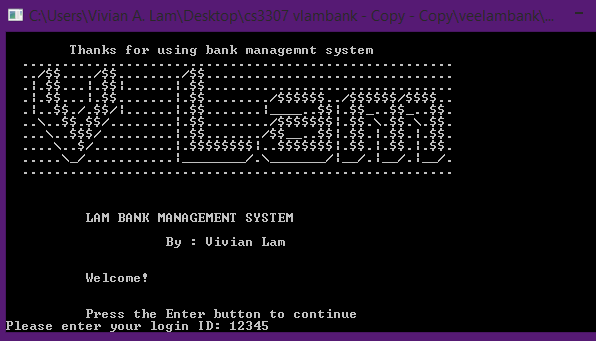
System deletes the account



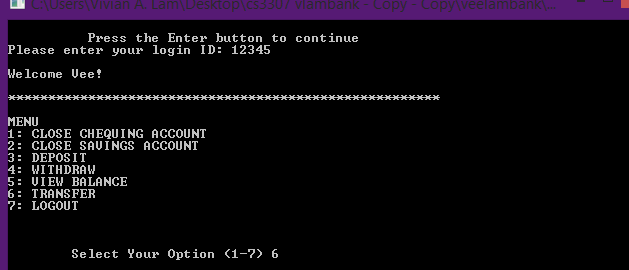
System shows that the account no longer exists

**5. Customer transfers funds**

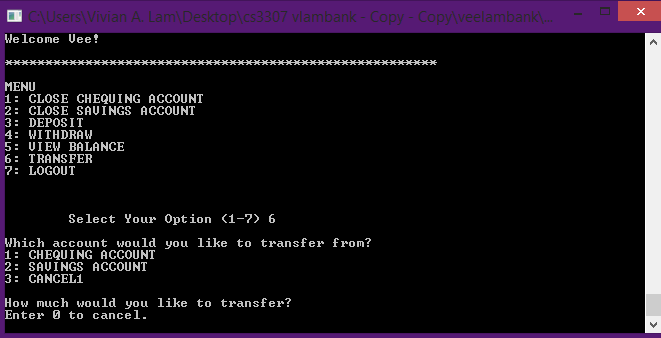
Customer logs in



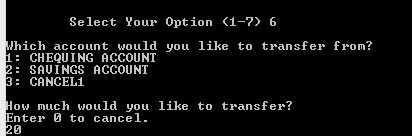
Customer selects transfer funds



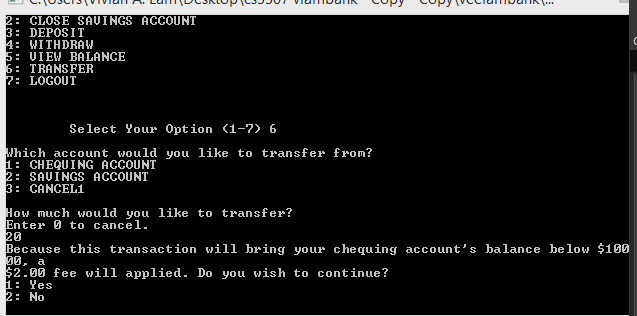
Customer selects which account to transfer from



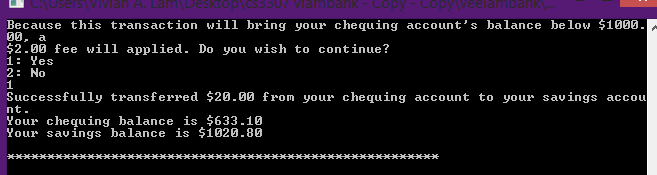
Customer selects amount



System transfer funds and applies warning if appropriate



System displays the new account data



**Deliverable 5: Implementation of the enhanced requirements**

Implemented login by having the program read text files (ones for customers and ones for manager/maintenance staff). If the ID numbered entered does not exist in any of the text files, the system will detect it as an invalid login. If the number does exist in the system, then it will be read and the corresponding customer and their details will be loaded. Customers is a vector array and each attribute of them is written to the file, accessed by a pointer. Sstream and iostream make this possible

**Deliverable 6: What I learned**

* Header files help keep the program neat. I should use them more.
* Pointers can reference anything, from a data structure to a variable etc.
* Function declaration helps the compiler and program know what types to accept, especially if the actual function is defined somewhere else later.
* The double colon “::” before a function ensures that resolution occurs from the global namespace instead of local namespace.
* I already knew about function prototypes from C, so I got more practice using them.
* C++ syntax