|  |
| --- |
| Automated Banking System |

**CSCI 3060U/SOFE 3980U – Winter 2016**

**Course Project Assignment #2**

**Front End Rapid Prototype**

**Design Document**

**Q&A Group Members:**

**Mohannad Abdo 100523158**

**Jeremy Kwok 100341977**

**Mirna Zohiry 100535658**

1. **Introduction**
   1. **Purpose**

This document provides a high and low level overview of the system architecture of the Automated Banking System. It outlines the architecture and functionality of the front end design. This architecture is developed around the requirements of the system. The document will then display the architecture of the system by providing detailed description of all classes and methods. Class and method relations will be represented in a UML diagram.

* 1. **Scope**

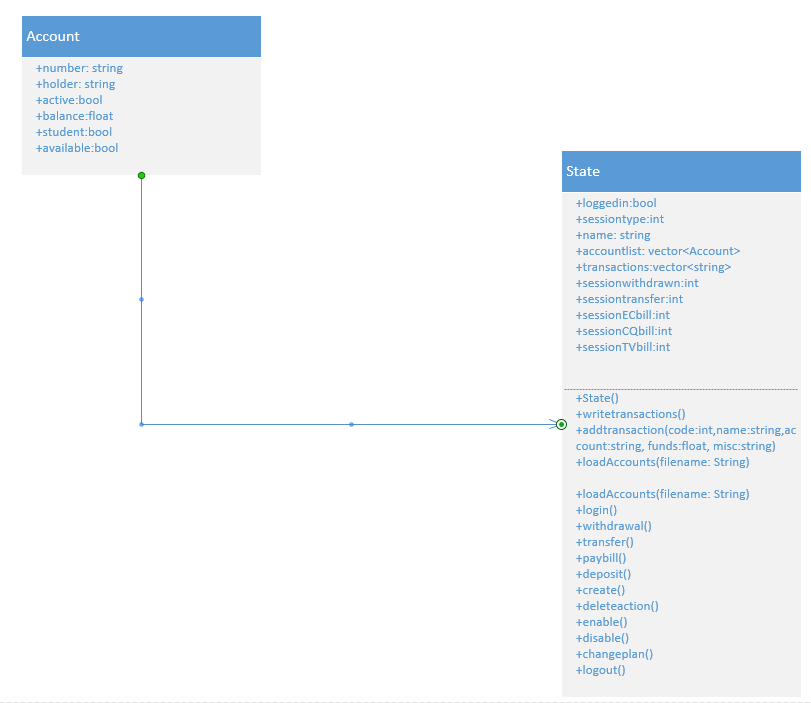
Automated Banking System is a prospective bank system which allows bank administration and bank members to create, maintain and complete transactions on bank accounts. The system consists of two main parts; the front end, an automated teller machine terminal for simple banking transactions, and the back end, an overnight batch processor to maintain and update a master banking account file. The Front End Rapid Prototype described in this document acts as a base design and a first version of the final product. It encompasses the overall structure of the system to be tested at an intermediate level but does not include final source code design.

* 1. **Overview**

The next section of the document has described high-level architectural design of the Automated Banking System in a UML diagram representation. The methods, specified classes, and their interactions. The third and final section of the Software Design Document is on low-level architectural design. This section includes a detailed explanation of what each class and method does, presented in a table along with input and output files of the current automated banking system.

1. **Architectural Design**
   1. **UML Class Diagram**

The high-level architecture of the system is shown in the UML diagram below. There are two classes used which are the Account class and the State class. The State class contains methods that allow for bank account transactions to occur. Also it checks the status of the account and whether it is logged in or not. The Account class contains information regarding each account such as the number, holder name, and balance.



1. **Detailed Design**
   1. **Detailed Design Explanation**

The following table describes the classes and methods used in the Automated Banking System architectural design.

|  |  |
| --- | --- |
| **Class:** State | |
| **Description:** The State Class contains all the data for the current state of the current session. It includes the name of the current logged in user, the session type whether it is standard user who is logged in or an admin. This class also keeps track of all accounts in the system as long as newly created ones. Moreover, it is responsible of all bank transactions occurred by either a standard user or an admin. In addition to logging in and out of the system. | |
| **Method** | **Description** |
| State() | State is a constructor of the State Class. It initializes some of the state variables, such as loggedin, sessionwithdraw, sessiontransfer, sessionECbill, sessionCQbill, and sessionTVbill. |
| addtransaction() | The addtransaction method adds a transaction to the transactions vector. |
| writetransactions() | The writetransactions method writes all the transactions in the transaction vector to the output file. |
| loadaccounts() | The loadaccounts method loads accounts from the bank accounts file to be stored in the system for transactions’ purposes. |
| login() | The login method allows a user to log in to his/her account by entering account name if a standard user. |
| withdrawal() | The withdrawal method allows a user to withdraw money from his/her account. It asks for amount to be withdrawn, checks for system constrains, and withdraw amount if applicable. |
| transfer() | The transfer method allows a user to transfer money from one account to another. It asks for amount to be transferred, checks for system constrains, and transfer money if applicable. |
| paybill() | The paybill method allows a user to pay a bill from his/her account. It asks for the company in which the bill will be paid to as well as amount to be paid, checks for system constrains, and pay the bill if applicable. |
| deposit() | The deposit method allows a user to deposit money to his/her account. It asks for amount to be deposited, checks for system constrains, and deposit amount if applicable. |
| create() | The create method creates new bank accounts. It can only be used by the admin in which he can create accounts for newly joined users. |
| deleteaction() | The deleteaction method allows the deletion of an existing account from the system. This method is privileged and can only be used by the admin. |
| enable() | The enable method allows in enabling one account. The account that is to be enabled, must be disabled in order for enable method to work properly. This method is privileged and can only be used by the admin. |
| disable() | The disable method allows in disabling an enabled account. The account that is to be disabled must be enabled in order for disable method to work properly. This method is privileged and can only be used by the admin. |
| changeplan() | The changeplan method is used to change an account plan from a student to a non-student plan. This method is privileged and can only be used by the admin. |
| logout() | The logout method allows a user to log out of the session and it writes all transactions occurred to the output file. This method is to be used at the end of each session. |
| **Class:** Account | |
| **Description:** The Account Class contains all the data for each single account. It includes the account number, account holder’s name, account state; whether the account is active or not, the current balance of the account, the account plan; student or non-student plan, and information about whether the account is available for transactions or not. | |
| Account() | Account is a constructor of the Account Class. It initializes some of the account variables, such as number, holder, active, balance, student, and available. |

* 1. **Input and Output Files**

Our program loads the CurrentBankAccounts.txt into the system and streams the output to output.txt. The writetransactions() method writes all the transactions in the transaction vector to the output file.