**Bank System Software**

Software Design Specification

**Group 6**

**Daniel, Davinder, John, Juan, Skone**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 4/5/2022 | 1.0 | Initial Version | Juan Gomez, SkonePanyatrir, Daniel Lara, John Parker Wilson, Davinder Singh |
| 4/23/2022 | 1.1 | Updated to have UML section and updated ATM | Juan Gomez |
| 4/24/2022 | 1.2 | Included ATM UML | Juan Gomez |
| 5/3/2022 | 1.3 | Update to 2.2, 3.3, added 6.5 | Skone Panyatriratn |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**1.** **Introduction 4**

1.1. Goals and objective 4

1.2. Statement of Scope 4

1.3. Software Context 4

1.4. Major consraints 10

**2.** **Data Design 11**

2.1. ATM Side 11

2.2. Server Side 11

2.3. Client Side 11

2.4. Internal/Employee Side 11

**3.** **Architectural And Component Level Design 12**

3.1. Program Structure 12

3.2. Description of ATM 12

3.3. Description of Server 13

3.4. Description of Client 12

3.5. Description of Internal/Employee 13

**4.** **User-Interface Design 14**

**5.** **Restrictions, limitations, and constraints 14**

**6. UML diagrams 14**

6.1. ATM 14

6.2. Server 14

6.3. Customer 14

6.4. Employee 14

**1. Introduction**

**1.1 Goals and Objectives**

This document describes the design, behavior, and interactions of the classes we have designed for the banking system

**1.2 Statement of Scope**

Decisions were made based on the following priorities- Usability, efficiency, and ease of use

**1.3 Software Context**

The Account information will be stored within the server and used via get\_cust\_info and chk\_emp\_id.

**1.4 Major Constraints**

Issue 1- How should the server send data to the other systems

Option 1- Send it as a packaged account object

Option 2- Send information as a long string

**2. Data Design**

**2.1 ATM Side**

* There will be three constructors for different levels of input from the driver
* Methods openShutter and closeShutter will open and close the money shutter on the machine
* Method calculate bills will see how many of each bill are required to withdraw money
* Method setUpIO will create the input and output objects needed for server communication
* Method start will run the ATM
* Method login will send the login message to the server and get the Account object. There will be two versions, one for regular use and one for debug/test use
* Methods deposit and withdrawal will allow the user to remove or add money to their bank account. There will be two versions of each, one for regular use and one for debug/test use
* Method logout will sign user out of the server
* Method stringToAccount will convert strings into Account objects. There will be two versions, one for regular use and one for debug/test use
* Method printReceipt will print out the receipt of the last transaction. There will be two versions, one for regular use and one for debug/test use
* Method timeOut will check to see if the user’s session has expired
* Method enableDebug will enable debug mode for the ATM
* Methods getDebug, getSocket, getUser, getBankAccount,getStatus,and getLoggedIn will return the appropriate objects
* Method setStatus will allow the ATM’s status to be set
* Method clearUser will clear local user data

**2.2 Server Side**

* **Server** wait for connect from clients
* Upon establishing a connection Server runs **ClientHandler** to send and receive **Message** object
* **ClientHandler** assign bank number to **bankName (Bank#)** and load **“Bank#-Accounts.txt,” “Bank#-Customers.txt,” “Bank#-Employees.txt”** into collection: **AccountsData**, **CustomersData**, and **EmployeesData**
* **ClientHandler** calls **desktopHandler** method when sender is “desktop”
  + **desktopHandler** handles type: login, inquiry, authorization, update, and logout
  + login check username and password again **EmployeesData** and returning status “success”
  + inquiry provides account information
    - to get account information inquiry checks if card number, account number, and pin belongs to one another by calling **isAccountBelongToCard** method under **CustomersData**
    - inquiry update **Account** status to “in use”
    - inquiry return **Message** to clients with **Customer** info and **Account** info
  + authorization checks if individual logging in has managerial position authorization
  + update save changes to **Customer** last name, first name, and address to file
  + update also save changes to **Account** status and balance to file
  + logout end employee current session by closing connection to server, returning status “success”
* **ClientHandler** calls **atmHander** method when sender is “atm”
  + **atmHandler** handles type: login, and logout
  + login checks if card number match pin entered
    - upon success **Server** change **Account** status of the card to “in use”
    - upon success call on **getAccount** method from **AccountsData**
    - return **Message** to client with **Account** info
  + logout change **Account** status from “in use” to “not used”
  + logout call **addOrModifyAccount** method to update balance
  + logout call **save** method to apply changes to file
  + logout close connection to **Server** and return status: “success”

**2.3 Customer Side**

* Method toString will convert Customer data into a string value

**2.4 Internal/Employee side**

**3. Architectural And Component Level Design**

**3.1 Program Structure**

The banking system will run on having a central database server that connects with ATMs

and an internal system for customers to use. ATM is used by customers by themselves while the internal system is used by employees to help customers

**3.2Description of ATM**

The ATM will consist of all the methods listed in 2.1. The ATM will also include the following variables-

* loggedin – a bool that will be true if the server was able to verify the account
* status – an enum that will list the ATM’s condition like broken or working
* Banknumber – A string that indicates which branch the ATM is at
* user – An account that stores the account data gotten from the server for local use
* socket – Will store the socket for communicating with the server
* OutputStream – An outputStream for server communication
* Objectoutputstream – An Objectoutputstream for server communication
* InputStream – An InputStream for server communication
* ObjectInputStream – An ObjectInputStreamfor server communication
* startTime – A long for storing the start time
* debug – A Boolean to store whether or not server is in debug mode

**3.3Description of Server**

In addition to the server methods listed in 2.2, the following variables and methods are also included:

* clientSocket : Socket – the socket receive from incoming client
* employee : EmployeesData – the collection used during processing of Message received
* customer : CustomersData – the collection used during processing of Message received
* account : AccountsData – the collection used during processing of Message received
* OutputStream – An outputStream for to-client communication
* ObjectOutputStream – An ObjectOutputStream for to-client communication
* InputStream – An InputStream for from-client communication
* ObjectInputStream – An ObjectInputStream for from-client communication

**3.4Description of Customer**

The Customer will consist of all the methods listed in 2.1. The ATM will also include the following variables-

* userName - the Customer’s name
* bank\_id - Customer’s unique identifier
* pin - Customers pin number used for accessing account information
* acct\_num - The account number associated with the customer
* acct\_type - Determines what type the account is, ie Checking, saving, etc.
* balance - the Customer’s balance

**3.5Description of Internal/Employee**

**4. User interface design**

**4.1 ATM**

The ATM will output basic JFrame objects for customers to interact with

**4.2 Server**

The server uses console/command line interface

**4.3 Customer**

The Customer uses a console/command line interface and can display its information as a string value using the toString method.

**4.4 Internal/Employee**

**5. Restrictions, Limitations, and Constraints**

**5.1ATM**

* The ATM cannot withdraw more than $10,000 at a time because more than that requires employee approval

**5.2Server**

* One bank server for customers and employees

**5.3 Customer**

* Each customer must a have a unique bank\_id

**5.4 Internal/Employee**

**6. UML diagrams**

**6.1ATM**

**Text

Description automatically generated**

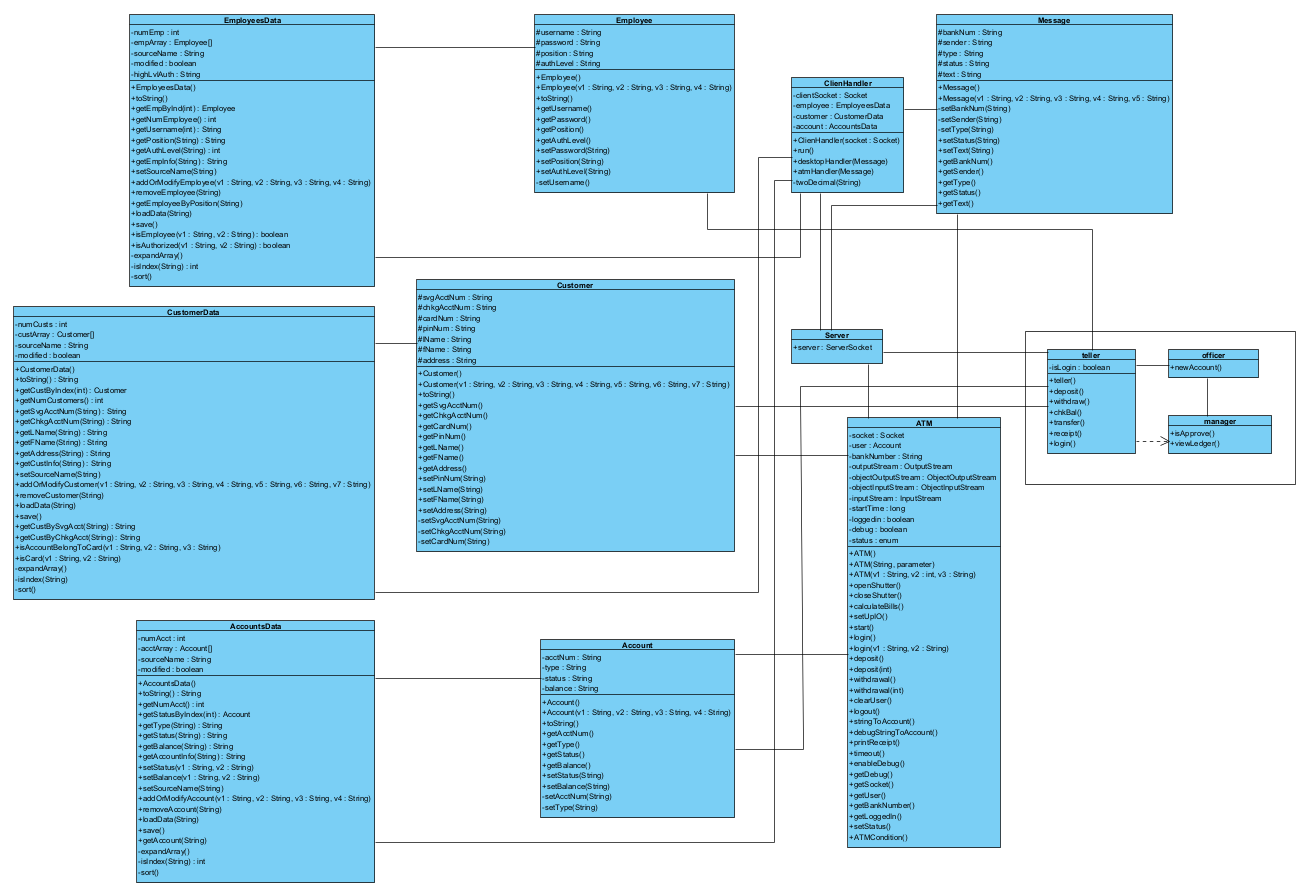
**6.2 Server**

* Please see 6.5 Full UML Diagram

**6.3 Customer**

**6.4 Internal/Employee**

**6.5 Full UML Diagram**

****