**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, Skone Panyatrir, Daniel Lara, John Parker Wilson, Davinder Singh |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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**

**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**

* Method **isAccount**will verify if the user is a customer of the bank
* Method connectToDatabase to establish connection with the server
* Method downloadFromDatabase will get necessary account information from the server and create a local copy
* Methods deposit and withdrawal will adjust the amount of money within a customer’s account
* Method openShutter will open the shutter for money deposits and withdrawals
* Method checkBalance will show customer the balance in the indicated account
* Method printReceipt will print a receipt of the customers transactions
* Method endTransaction will call uploadToServer to return the updated account object, and will then delete its local copy and sign out user

**2.2 Server Side**

* Method **clientInputHandler**will listen for triggers from clients in order to access data of customers or employees associated with client originator.
* For employee, this method will initialize the filename **“employee\_id.txt”** and call on **loadFile**to open the file.
  + It will then call on **chk\_emp\_id** to authenticate username and password
  + The method will listen for employees request to access customer data
    - **get\_cust\_info** will be call on to relay back customer’s data to employees
  + The method will listen for employee request to save data and call on **saveFile**.
  + The method will also listen for authentication request from employees for approval of transaction over $10000 and call on **chk\_auth\_level** to confirm proper level of authorization.
* For ATM, this method will initialize the filename **“customer.txt”** and call on **loadFile**to open the file.
  + It will call on **chk\_atm\_auth** to authenticate user account to pin
  + **get\_cust\_info** will be call on to provide ATM with customer’s data
  + **saveFile** will be called upon the completion of ATM transaction

**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

**3.2 Description of ATM**

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

* accountConfirm – a bool that will be true if the server was able to verify the account
* ATMCondition – an enum that will list the ATM’s condition like broken or working
* Cash – It will be of type currency, a custom class, and will keep track of the money in the machine
* CurrentUser – An account that stores the account data gotten from the server for local use

**3.3 Description of Server**

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

* numEmployee - number of concurrent employee logged in
* numCust - number of concurrent customers accessing the system
* Employee[ ] - array to store concurrent employee
* Customer[ ] - array to store concurrent customers information

**3.4 Description 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 does not have unlimited money
* 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**