**Banking System**

*Design Document*

# **Introduction**

This document outlines the requirements for the Banking System. It will explain the purpose and features of the system, what the system will do, and the interface of the system. This document is intended for the client of the system.

## **Goals and Objectives**

This document describes important aspects of the implementation of the **#bankingsystem**, **#ATM**, **#teller**, **#customer**, and **#account**.

## **Statement of Scope**

Decisions in this document are made based on the following priorities (most important first): Maintainability, Usability, Portability, Efficiency

## **Software context**

1.3.1 The abstract class User will integrate with the classes Teller and Customer. It will maintain account permissions separated by ID and name.

1.3.2 In order to maintain security, the class Login will be used to verify the User. Login will process user credentials via getID() and getPassword().

1.3.3 The class Account will maintain balance as well as the customer(s) that are on the account.

1.3.4 The class Action will log a history of actions made in the account. Action will inherit from the enum class actionType.

## **Major Constraints**

Issue 1: How can we determine which accounts belong to which customer?

Option 1.1: Store the account and customer information in a map and search by unique id.

Issue 2: How can we determine what types of actions can be performed/requested from the customer?

Option 2.1: Depending on where the customer requests its action type determines whether it can do that action. If a customer requests to create an account at a ATM they would be denied, but if a customer requests to create an account at a bank, a bank teller has the power to create the account for the customer.

# **Data Design**

## **Client Side**

* There will be a GUI that displays multiple options for the customer to pick. It will show a bank option and an atm option. They will both ask for login information to identify the customer. It will also display some of the action types such as withdraw, deposit, transfer, create account, etc.

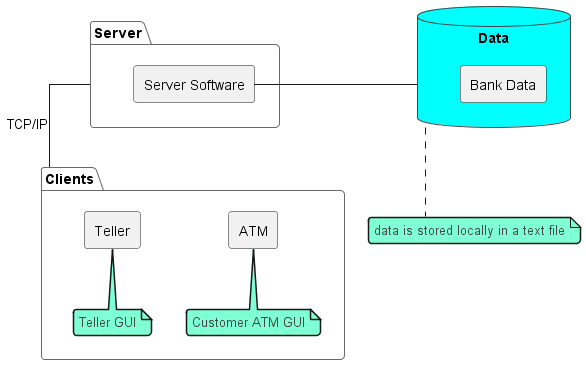
## **Server Side**

* The primary class banking system will handle most of the server-side data handling. It has the data methods readData() and saveData() to read and store information about the customer.
* We can think about the banking system as our server and the ATM is the client that will inherit some of the methods that our banking system does.

# **Architectural and Component-level Design**

## **Program Structure**

### **Architecture Diagram**



## **Description of client**

3.2.1 Client processing narrative

(functions and processes described)

The bank teller and customers will be required to login with their given ID and password. The class Login implements the methods getID() and getPassword().

A new customer will make a request to the teller to create an account. Through the BankingSystem class, the method createAccount() will be utilized to make the account.

The ATM will use the sendRequest() method in order to process what action the customer wants to perform. Since this will be a customer interacting with the ATM, actions are limited to: WITHDRAW, DEPOSIT, and TRANSFER.

3.2.2 Client interface description

(input and output interfaces described)

Clients will be presented with a login page where they will need to input their assigned ID as well as a password. There are two seperate logins, one for bank employees to access the banking system and one for bank customers to access an ATM. The customer can add funds into their account, they can also withdraw money and make transfers, provided they have adequate funds to do so. Any action made in the account will not only be shown on the ATM system, but it will also be reflected on the banking system. The bank employee/ teller can perform the same functions as a customer does with some added administrative commands such as creating or deleting accounts and adding or removing people from an account.

3.2.3 Client processing details

(algorithmic description)

Accounts and customers data will be read from and saved within the banking system via a text file. Actions made in the account will be saved in the Action class. Actions made by the customer in the ATM system will be passed on to the banking system.

## **Description of Server**

The server software will have no interface the user will interact with. The server’s will run on a machine and use local files on that machine to save and load data about the banking system. The server software’s role is to be the intermediary between the data and the client. It will take requests from clients, verify them, update the data, and send information back to the clients.

## **Software Interface Description**

3.4.1 Human Interfaces

We will be making two interfaces that humans will interact with. We will be making an ATM interface and a teller interface. Using these ATM interfaces, customers will be able to login using their ID and password. When logged in the user will be able to check their bank account(s) balance, withdraw, deposit, and transfer money. The teller interface will allow tellers to login using their ID and password and then do the actions the customer can do on their behalf. The teller interface will also include all the ATM options as well as options for opening and closing accounts.

# **User Interface Design**









# **Restrictions, Limitations, and constraints**

## 5.1 Since the Java Application will require a server application and client application but not use databases or other frameworks, due to time and requirement constraints data will be unencrypted for simplicity's sake.

5.2 We are required to use the Eclipse IDE so for our test cases we will be implementing JUNIT testing.

5.3 Using no other frameworks and outside libraries means we will be using the GUI widget toolkit swing.

5.4 Since databases and other online means of storage are not allowed, we will store all data in text files.

5.5 We are required to create a GUI application without console inputs/outputs.

# **Testing Issues**

6.1 Whether the banking system/atm will stop transactions if there is more than one customer logged into the same account doing transactions at the same time

6.2 Is the banking system/atm correctly withdrawing and depositing the correct amount of money every time

6.3 Check to make sure that the process of clients sending requests, the server receiving them, and the client receiving the response is working correctly.

6.4 Check that adding a customer to a joint account allows the new customer to deposit/withdraw/etc. from that account.

6.5 Check to make sure the bank teller has the correct action types and that customer isn’t able to access bank teller commands.

6.6 Check that each customer’s ID is unique to that customer and prevent duplicate IDs from being created

# **UML**

