

**Group 6 Banking App**

*Design Document*

*Project Guide****:***

Christopher Smith

**Submitted by:**

Tuong Thien Tran - ar3940

Angela Wei - aq5229

Gurjas - ie2846

Ebadullah - xp4648

Anthony - dt5848

# **Purpose**

This document outlines the requirements for the Group 6 Banking Application.

## **Scope**

This document will catalog the user, system, and hardware requirements for the Group 6 Banking App. It will not, however, document how these requirements will be implemented.

## **Definitions, Acronyms, Abbreviations**

ATM: Automatic Teller Machine: Allows the user to perform simple bank actions without the help of a bank employee.

Online Customers: Online Customers is an actor, who uses their user credentials to login to access their account, which gives them access to their accounts.

Teller: Teller is an actor, who uses their credentials to login to access their banking account, which grants them the ability to help users (ex: freeze user accounts)

Account: Class has different types (checkings, savings, credit card). After logging into the user, the user gains the ability to complete basic bank behaviors. (ex: deposit/withdraw)

## **References**

Use Case Specification Document – Step 2 in assignment description

UML Use Case Diagrams Document – Step 3 in assignment description

Class Diagrams – Step 5 in assignment description

Sequence Diagrams – Step 6 in assignment description

## **Overview**

The Group 6 Banking Application is designed to be a system for a large bank. Authorized bank employees will be able to use this software to assist customers with their banking needs. The interface provides most standard banking services. There will also be an automated teller machine (ATM) application for users. The application is written using Java with a GUI that operates over TCP/IP. Security is a big point of emphasis for this project.

# **Overall Description**

## **Product Perspective**

What is the product? In order for the bank to provide the best experience for its users, there needs to be an easy way for the customer to exercise the features available to them at their own convenience. Providing a banking app will help the customers make changes to their account without having to interact with a human teller, and will allow the bank to automate processes that don’t require human labor.

What does the product do? The system allows a bank customer to make changes to their account like transferring money, sending money, making payments, etc. This system will allow a bank teller employee to log onto the system and do everything that a standard customer user can do as well as some more functionality (eg: grant a credit limit increase, terminate an account, cash a check, etc.)

Problem domain: Banking, money, finance, accounts, savings, checkings, credit/debit cards, interest rates, credit limits, money transfer, currency exchange, late/overdraft fees, etc.

Black box view: User wants to make changes to their account → account changes are made

## **Product Architecture**

The system will be organized into about 10 major modules: the ATM module, the Account module, the Checking module, the Credit Card module, the Save module, the User module, the Teller module, the Authenticator module, Client module, and the Server module.

Note: System architecture follows standard OO design practices.

## **Product Functionality/Features**

The high-level features of the system are as follows: Withdraw and deposit money, transfer and send money from accounts, pay bills, view transaction history. Note: These features are expressed with greater detail in Section 3 of this document.

## **Constraints**

Since users need to be able to view their transaction history for the their accounts, a server client connection needs to be implemented

Since our users need a secure account, our login needs to be able to authenticate a user

Since users will use a browser or installed application to access the system, no browser-specific or mobile specific code is to be used in the system

## **Assumptions and Dependencies**

It is assumed that the maximum number of users for an individual account will be 1 on a single device

It is assumed that if users expect to have the application behaving as intended, that a secure internet connection will be established

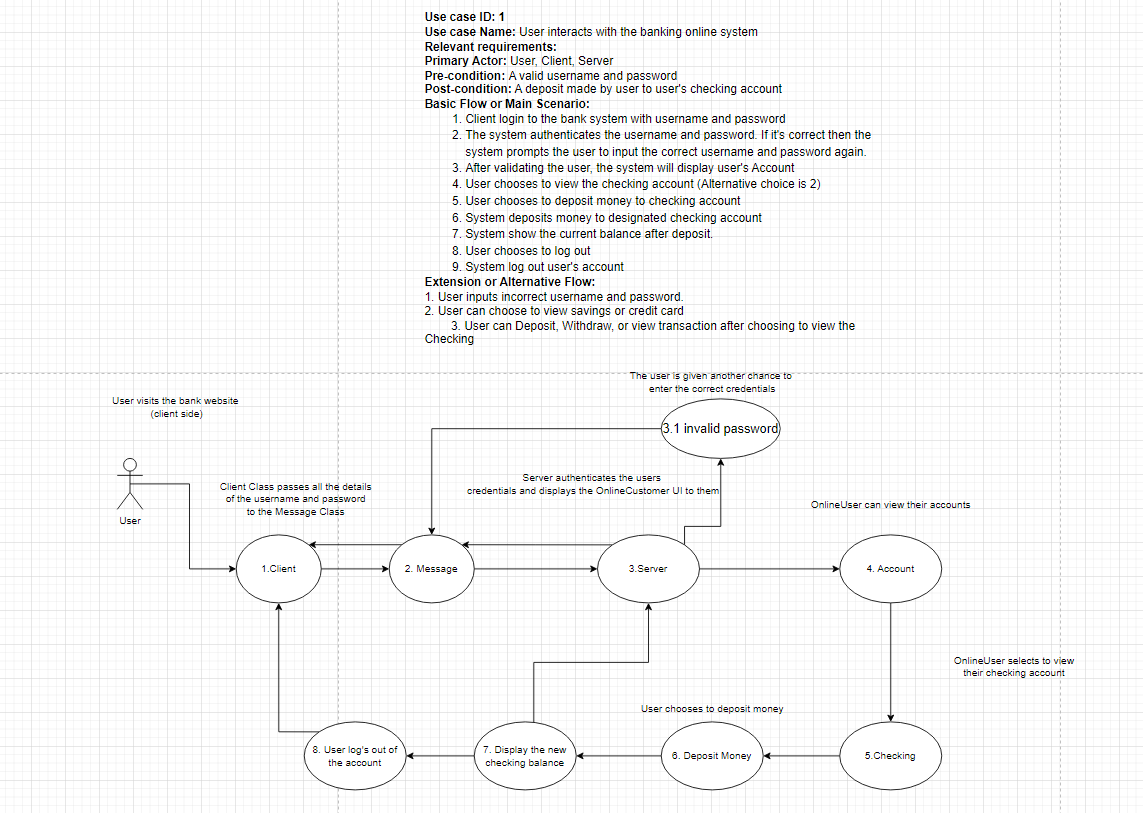
It is assumed that users will be able to contact a human teller for some actions regarding user account changes

The ability of the application to work as intended depends on the user’s ability to update the application accordingly

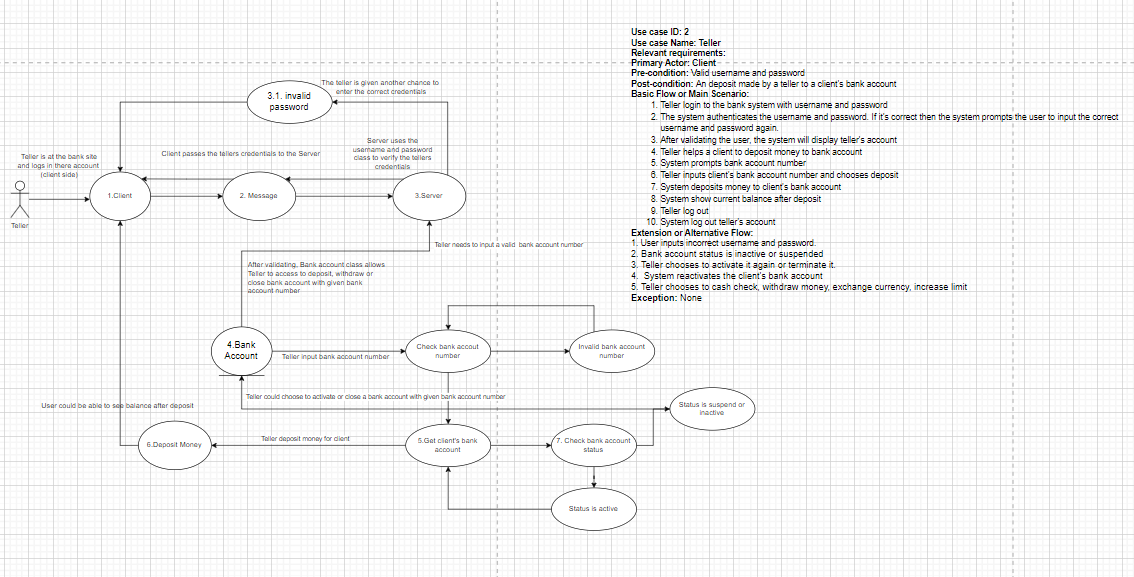
# **System architecture**

## **Use cases**

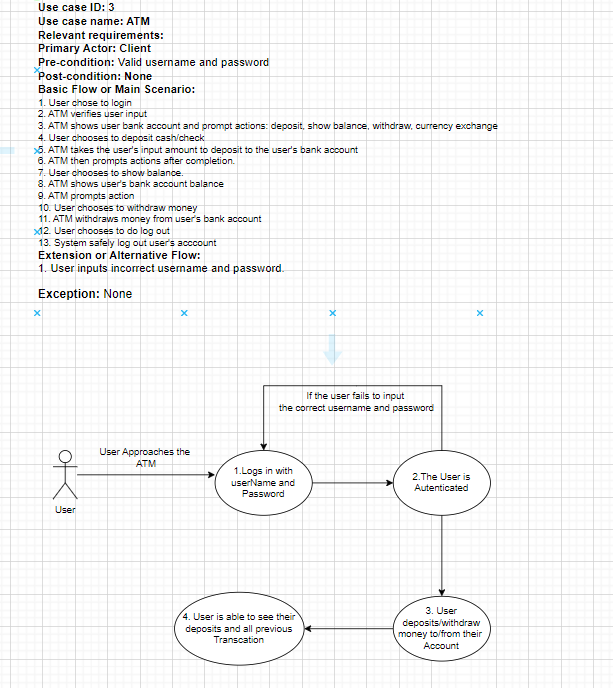
3.1.1 Use case #1



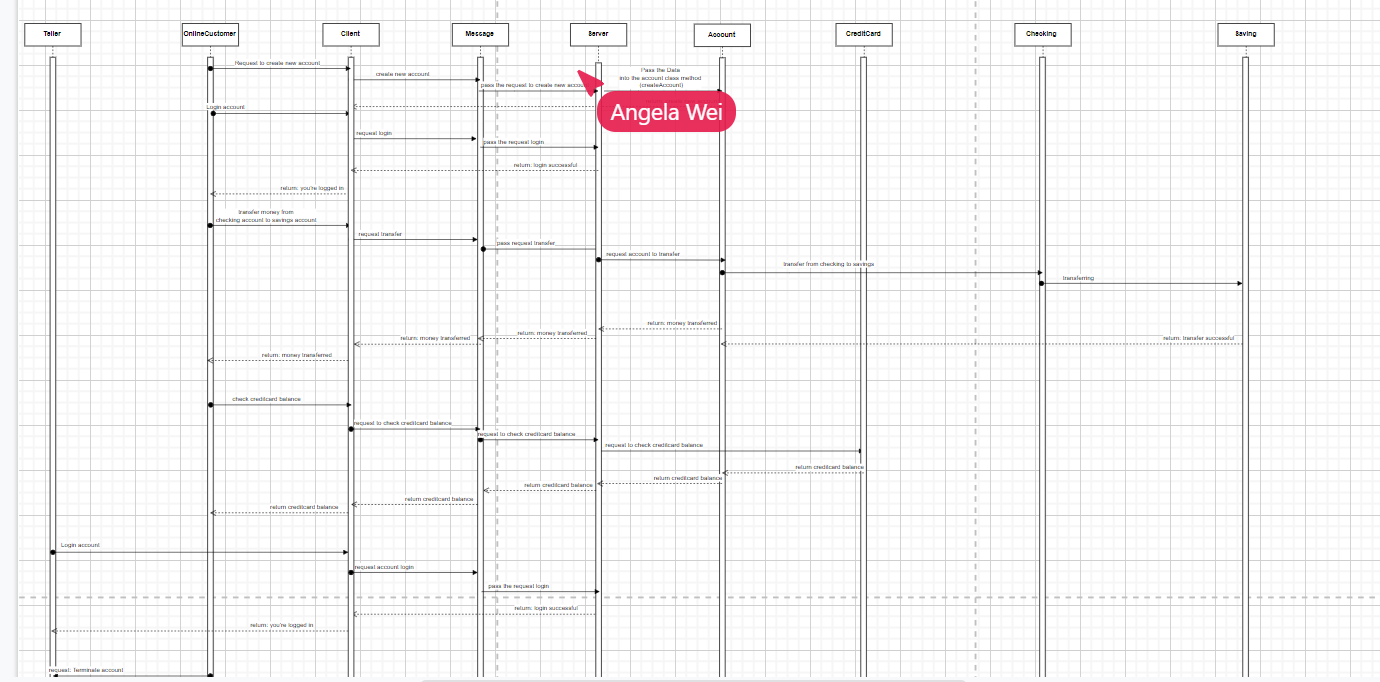
3.1.2 Use case #2



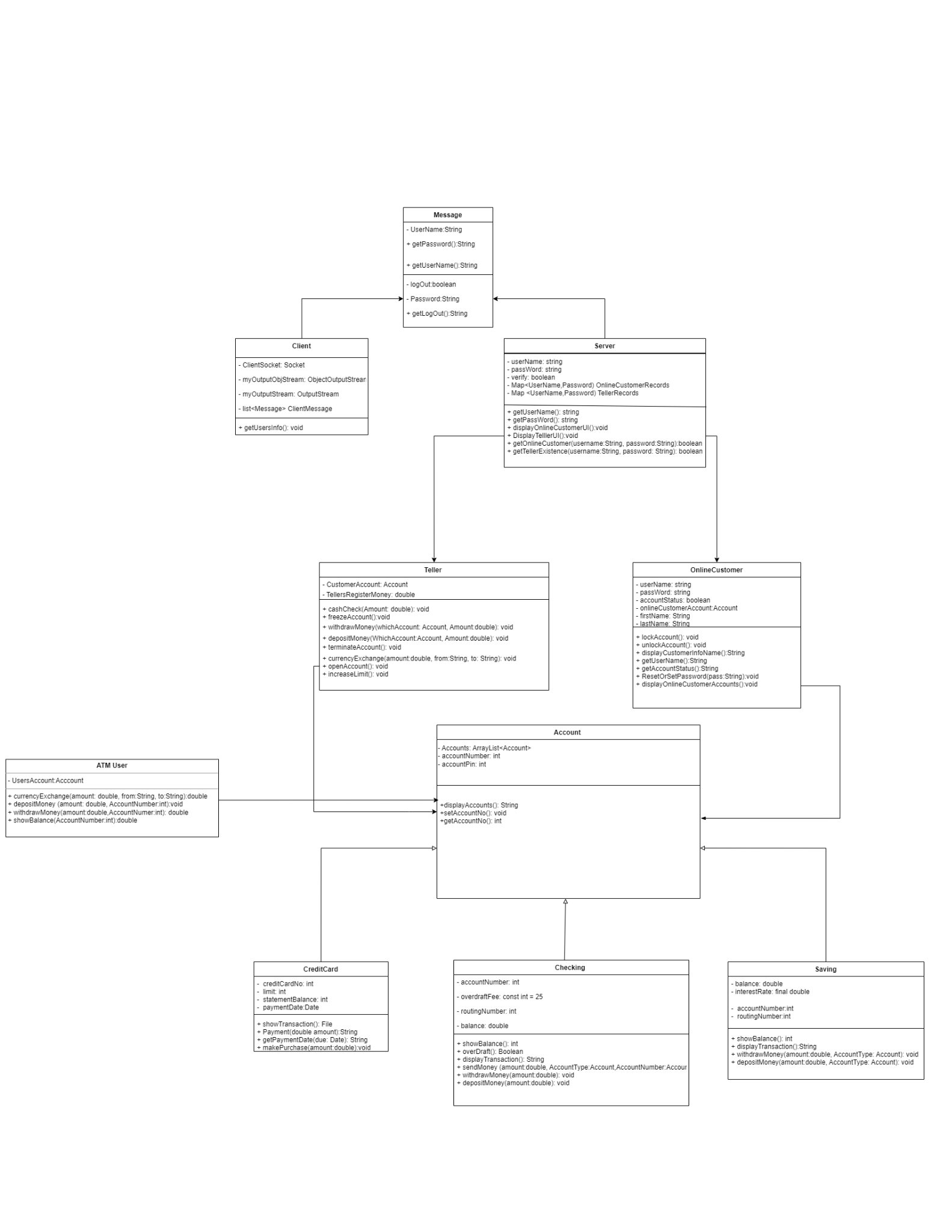
3.1.3 Use case #3



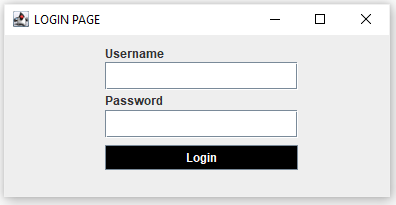
## **Sequence diagrams**



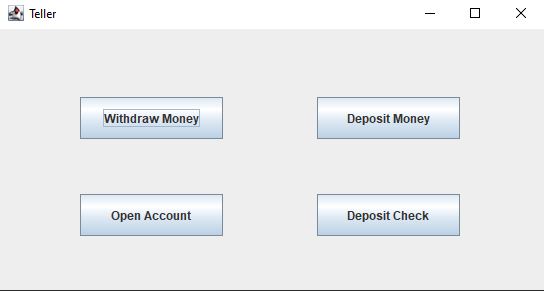
## **UML**



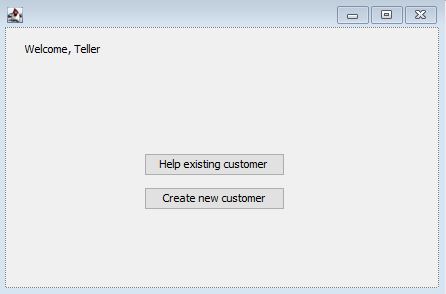
# **User Interface Design**



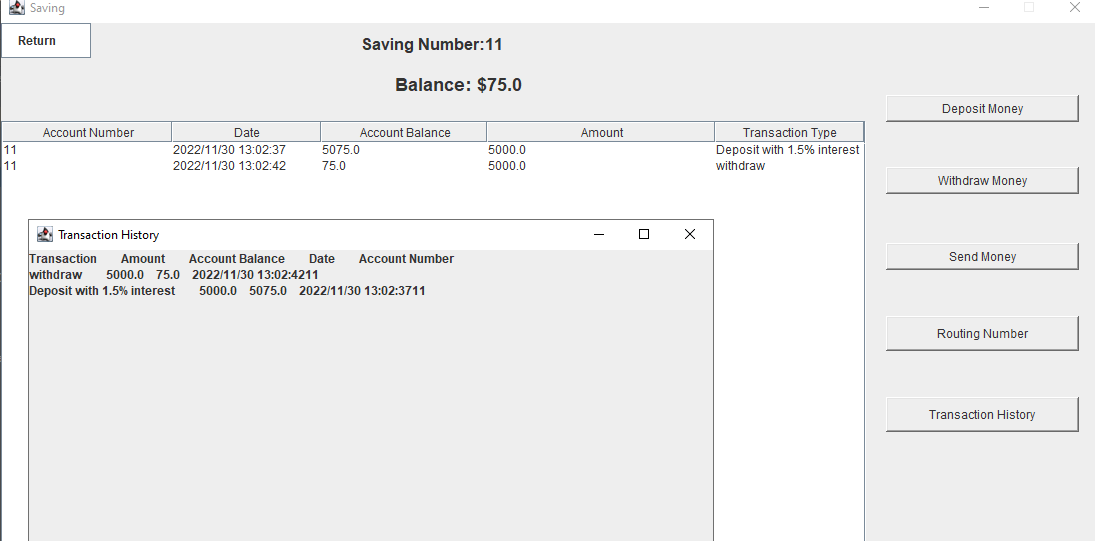
User Login



Teller options



Teller welcome screen



Saving Gui for customer with transaction history

# **Dependencies**

5.1.1

The original functionality of Teller GUI was to validate telleruser’s credentials and then validate customer user's credentials. Teller GUI would appear after the validation process was completed. During the implementation of the Teller GUI class, we were unable to get the class to complete the validation and have the GUI pop up.

5.1.2

The checking and savings accounts cannot be accessed or used unless the user verifies their username and password.

5.1.3

In order to make the GUI classes, we had to utilize the swing toolkit as well as import the Window Builder tool.

5.1.4

The teller has to open an account for the customer before the customer can access the system.