# Group 3

*Banking System*

*Software Requirements Specification*

## Revision History

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 02/22/2024 | 1.0 | Initial Version | Sungmo Koo  Daniel Rodriguez  Breanne Loo |
| 02/28/2024 | 1.1 | Added diagrams and small edits | Sungmo Koo  Daniel Rodriguez  Breanne Loo |
| 02/29/2024 | 1.2 | Created two new classes after seeing what other groups have created. Updated Use Case Specifications. Finalized it. | Sungmo Koo  Daniel Rodriguez  Breanne Loo |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ii**

## Table of Contents

1. **PURPOSE** [**4**](#_heading=h.ihv636)
   1. SCOPE [4](#_heading=h.30j0zll)
   2. Definitions, Acronyms, Abbreviations [4](#_heading=h.3znysh7)
   3. References [4](#_heading=h.tyjcwt)
   4. Overview [4](#_heading=h.1t3h5sf)
2. **OVERALL DESCRIPTION** [**5**](#_heading=h.32hioqz)
   1. Product Perspective [5](#_heading=h.2s8eyo1)
   2. Product Architecture [5](#_heading=h.3rdcrjn)
   3. Product Functionality/Features [5](#_heading=h.lnxbz9)
   4. Constraints [5](#_heading=h.1ksv4uv)
   5. Assumptions and Dependencies [5](#_heading=h.2jxsxqh)
3. **SPECIFIC REQUIREMENTS** [**6**](#_heading=h.1hmsyys)
   1. Functional Requirements [6](#_heading=h.3j2qqm3)
   2. External Interface Requirements 7
   3. Internal Interface Requirements 7
4. **NON-FUNCTIONAL REQUIREMENTS** **8**
   1. Security and Privacy Requirements 8
   2. Environmental Requirements 8
   3. Performance Requirements 8
5. **USE CASE SPECIFICATIONS**……………………………………………………………………….**9**
6. **UML USE CASE DIAGRAMS**..………………………………………………………………………**13**
7. **UML CLASS DIAGRAM**.……………………………………………………………………………..**15**
8. **UML SEQUENCE DIAGRAM**…………….………………………………………………………….**16**

**iii**



This document outlines the requirements for the Banking System (B.S.).

### Scope

### 

### Our banking systems will provide a basic account creation system, contact system, atm system, login system, and fraud protection.

### Definitions, Acronyms, Abbreviations

B.S.: Banking System

### References

Use Case Specification Document – See page 8

* User signs into their account
* Creating Account
* Deleting Account
* ATM or Teller window, Bank User chooses Teller
* ATM or Teller window, bank user chooses ATM
* Bank User uses ATM to deposit money
* Bank User uses ATM to withdraw money
* Bank User uses ATM to transfer money

UML Use Case Diagrams Document – See page 13

* Owner Changes User Status

Class Diagram – See page 15

Sequence Diagrams – See page 16

* Bank User makes depot at ATM
* User logs into Bank System
* Creating Account by Teller

[GitHub](https://github.com/danny-zebby/Banking_System)

### Overview

### Our Banking system is a Java application with a GUI that operates over TCP/IP. It allows users to talk to a teller to create, change, and delete checking and savings accounts. Along with this, there is an around the clock ATM service for basic deposits, withdraws, and transfers.



### Product Perspective

When implemented correctly the B.S.will provide a means to handle transitions through a server. In addition it will also be operational even without working staff thanks to the ATM service.

### Product Architecture

The B.S. will be organized into five major modules: the User module, the Bank User module, Teller module, Owner Module, and the ATM module.

### Product Functionality/Features

The high-level features of the B.S. are as follows (see section 3 of this document for more detailed requirements that address these features): An ATM service that handles Transaction, withdrawals, and deposits. Will have unique account creation. Will have different categories of Users. Users can have multiple accounts, and they can share accounts with other users.

### Constraints

2.4.1 The B.S. will not run without Java installation. (see section 4 of this document for more environmental requirements)

2.4.2. The number of unique users is the same size as the max value an integer could be in java, 2,147,483,647.

2.4.3.

### Assumptions and Dependencies

2.5.1 If an account exists we can assume there will be at least one user who owns the account.



### Functional Requirements

* + 1. **Common Requirements:**
       1. There should be a main module called Bank that holds other modules together.
       2. There will be a login page, and way to store this login information.

### User Module Requirements:

3.1.2.1 A user logins in as a bank user or teller with their name, birthday, unique username, and password.

3.1.2.2. A user can be classified as a bank user, a teller, or an owner.

3.1.2.3. The User Module works with the server module for logging in a user in and keeping track of whether or not a user is currently logged in.

### Bank User Module Requirements:[[1]](#footnote-0)

3.1.3.1. Bank User will have methods to get their id, username, and password

3.1.3.2 Bank users are presented with two options, ATM service and Teller assistance if available. The number of tellers available should be presented to the user as well.

3.1.3.3 Bank users can own as many accounts as they like.

3.1.3.4 A bank user with access to an account can via the teller add other users to the account. Any user can close how many people can enter an account.

3.1.3.5. Bank Users have cash that they can add to their accounts.

### Teller user Module Requirements:[[2]](#footnote-1)

3.1.4.1 The Tellers can create and delete saving and checking accounts.

3.1.4.2 The accounts the teller creates have a pin that is randomly generated

3.1.4.3 Tellers can change the pin of a user’s account at their request.

### Owner user Module Requirements:

3.1.5.1. The owner user should have access to turn users into tellers and tellers to users.[[3]](#footnote-2)

3.1.5.2. The owner has the same accessibility as the bank user and teller.

3.1.5.3. The owner’s login is unique in the source code and owners are only created through the source code.

### ATM Module Requirements:

3.1.6.1. ATM functions should be deposit, withdraw, and transfer.

3.1.6.2. ATM receives the account number and pin number from user

3.1.6.3. ATM transfer funds only work if a Bank User wants to send funds to a different account. A Bank User cannot request funds from another account.

### Account Module Requirements:

3.1.7.1. The accounts are linked to at least one or more Bank User

3.1.7.2. Accounts have a status type of checking or saving.

3.1.7.3. Accounts have a total that must get saved in a txt file via the server.

3.1.7.4. To make any changes to an account, the correct pin must be matched to the account.

### Server Module Requirements:

3.1.8.1. The server module works with the user module for logging in users, and keeping track of whether or not a user is logged in the B.S..

3.1.8.2. The server is responsible for connecting clients to the server and allowing them to access our B.S.

3.1.8.3. The server can connect multiple clients via multithreading.[[4]](#footnote-3)

3.1.8.4. The server will need keep track of accounts and information in a txt file

### External Interface Requirements

* + 1. This is a Java application with a GUI that operates over TCP/IP

### Internal Interface Requirements

* + 1. This B.S. requires a server application and client application
    2. There is no web or HTML component. No databases, libraries, frameworks, or other technologies



### Security and Privacy Requirements

* + 1. Login into one account from two different attempts should be prevented.
    2. IP addresses of users will not be saved in our B.S.

### Environmental Requirements

* + 1. B.S. must be developed in JDK.
    2. B.S. must at least be ran on a JRE
    3. B.S. must be run on a computer.

### Performance Requirements

* + 1. B.S. must communicate with or between users within an appropriate[[5]](#footnote-4) amount of time.
    2. Server could possibly have some down time.[[6]](#footnote-5)
    3. B.S. will not apply a finite number to data structures, Data structures will be dynamically sized.



Case ID: 000001

Case Name: User signs into their account

Relevant Requirements: 3.1.2.1, 3.1.2.2

Primary Actor: User

Pre-Conditions: A teller has already created an account for the user.

Post-Conditions: The user will be able to access their bank user account using their credentials.

Basic Flow/Main Scenario:

1. Users enter their name, birthday, unique username and password onto the login.
2. The system validates the information given from the login.
3. System responds by allowing the user to login to their bank user account.

Alternative flow:

* If the user is also a teller, they are logged in as a teller instead of a bank user.
* If the user is an owner, they are logged in as an owner instead of a bank user.
* If the user puts in any of the following information incorrectly, the system does not log the user in and asks them to try again.

Exceptions:

* If a user attempts to sign into an account that does not exist, an error message is displayed.

Related Use Cases:

* ATM or Teller window, Bank User chooses Teller
* ATM or Teller window, Bank User chooses ATM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Use Case ID: 000934

Use Case Name: Creating Account

Relevant Requirements: 3.1.4.1 / 3.1.4.2

Primary Actor: Teller

Pre-conditions: The teller must be logged into the system using a security code.

Post-conditions:

1. The new account (either checking or saving) is successfully created in the system.
2. The user is provided with a randomly generated PIN.

Basic Flow or Main Scenario:

1. Teller logs into the system using a security code.
2. Teller selects the option to create a new account.
3. System prompts the teller to specify the type of account (checking or saving).
4. Teller chooses the account type.
5. System generates a random PIN for the account.
6. Teller inputs customer details (name, account number, pin, amount of balance).
7. Teller verifies the details and confirms account creation.
8. System stores account information, associated PIN, and the selected account type.

Extensions or Alternate Flows:

* If any required customer information is missing or invalid, the system prompts the teller to correct the details.
* If there is a system error during the account creation process, the teller is alerted with an appropriate error message.

Exceptions:

* If the system fails to generate a random PIN, the teller is prompted to retry.

Related Use Cases:

* Account modification, Account deletion.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Use Case ID: 000935

Use Case Name: Deleting Account

Relevant Requirements: 3.1.4.1 / 3.1.4.2

Primary Actor: Teller

Pre-conditions: The teller must be logged into the system using a security code.

Post-conditions:

1. The account has been successfully deleted from the system.

2. A confirmation message is displayed to the teller.

Basic Flow or Main Scenario:

1. Teller initiates the process by logging into the system using the assigned security code.

2. Teller navigates to the customer account management section.

3. Teller selects the option to delete an account.

4. The system prompts the teller to input the account details for verification.

5. Teller provides the necessary account details.

6. The system validates the information and prompts the teller to confirm the deletion.

7. Teller confirms the deletion.

8. The system removes the account from the database.

9. A confirmation message is displayed, indicating successful account deletion.

Extensions or Alternate Flows:

* If the system cannot validate the account details provided by the teller, it prompts the teller to re-enter the information.
* If the teller decides to cancel the deletion process at any step, the system returns to the previous state, and no account deletion occurs.

Exceptions:

* In case of technical issues or errors during the deletion process, an error message is displayed, and the teller is prompted to contact technical support.
* If the account to be deleted has pending transactions or unresolved issues, the system alerts the teller and provides instructions on resolving these matters before proceeding with the deletion.

Related Use Cases:

* Account creation
* ATM or Teller window, Bank User chooses Teller

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Use Case ID: 001881

Use Case Name: ATM or Teller window, Bank User chooses Teller

Relevant Requirements: 3.1.3.2

Primary Actor: Bank User

Pre-conditions: The bank User must complete their login and or account creation.

Post-conditions: The bank user will start a process with the teller and then quit or will start no process with the teller and quit the teller services

Basic Flow or Main Scenario:

1. The Bank User is presented with two buttons, ATM service and Teller Service
   1. The Teller button will have a number in parenthesis.
2. The Bank User selects Teller service
3. The Bank User is prompted with a list of teller services: account creation, account deletion, addition of user to account, close user addition, pin change or to exit teller services.
4. The bank user will then select an option and then that process will begin, once that process is over the bank user will be brought back to step three of this flow.
5. Once the bank user selects quit this process is over.

Extensions or Alternate Flows:

* If there is no teller available at the moment when a user logins into the system and the user selects a teller then the user will be given a message apologizing for no tellers available and then sent back to the select option of teller or ATM.

Exceptions:

* If the User logins into a owner account then their selection screen will include the option to toggle a user’s id to teller or not
* If the User logins into a teller account then their selection screen will include the option to help serve a bank user

Related Use Cases:

* Account creation

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Use Case ID: 180801

Use Case Name: ATM or Teller window, bank user chooses ATM

Relevant Requirements: 3.1.3.2

Primary Actor: Bank User

Pre-conditions: The bank User must complete their login and or account creation.

Post-conditions:The bank user will start a process with the ATM and then quit or will start no process with the ATM and just quit the ATM services

Basic Flow or Main Scenario:

1. The Bank User is presented with two buttons, ATM service and Teller Service
2. The Bank User selects ATM
3. The ATM request for the user’s account number and pin number. The bank user give the following information
4. The pin matches the account number so the ATM prompts the bank user with the following service: deposit, withdraw, transfer, or quit
5. The bank user will then select an option and then that process will begin, once that process is over the bank user will be brought back to step four of this flow.
6. Once the bank user selects quit this process is over.

Extensions or Alternate Flows:

* If the bank user gives an incorrect pin, account number, or does not even own an account then they will not get passed step 3.

Exceptions:

* If the User logins into a owner account then their selection screen will include the option to toggle a user’s id to teller or not
* If the User logins into a teller account then their selection screen will include the option to help serve a bank user

Related Use Cases:

* Account creation
* ATM or Teller window, Bank User chooses Teller

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Case ID: 181900

Case Name: Bank User uses ATM to deposit money

Relevant Requirements: 3.1.6.1, 3.1.6.2

Primary Actor: ATM

Pre-Conditions: A User must have already signed into their account.

Post-Conditions: The money the user has deposited is added to the total amount in their account.

Basic Flow/Main Scenario:

1. The bank user selects to deposit money to an account.
2. The system responds by asking for the amount to deposit.
3. The bank user chooses a specific amount to deposit.
4. The system adds the amount to the total amount in their account.

Alternative flow:

* If the user attempts to deposit zero dollars, the system will not deposit money and tells the user to choose a nonzero amount.

Exceptions:

* If the user attempts to deposit money into an account that doesn’t exist, the system will display an error message.

Related Use Cases:

* ATM or Teller window, bank user chooses ATM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Case ID: 181901

Case Name: Bank User uses ATM to withdraw money

Relevant Requirements: 3.1.6.1, 3.1.6.2

Primary Actor: ATM

Pre-Conditions: Bank User must have already signed into their account.

Post-Conditions: The ATM service will have withdrawn a specific amount of money the bank user has requested out of their account.

Basic Flow/Main Scenario:

1. The bank user selects to withdraw money from their account.
2. The system responds by asking the user the amount of money to withdraw.
3. The bank user chooses a specific amount to withdraw.
4. The system takes that amount of money out of the user’s account to give to the user.

Alternative flow:

* If the bank user attempts to withdraw from an account that holds zero balance, the system does not perform the action and tells the user there is no money in the account.
* If the bank user attempts to withdraw more money than what the account holds, the system tells the user that they are withdrawing more than what the account has currently.

Exceptions:

* If the user attempts to withdraw from an account that doesn’t exist, an error message is displayed.

Related Use Cases:

* ATM or Teller window, bank user chooses ATM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Case ID: 181902

Case Name: Bank User uses ATM to transfer money

Relevant Requirements: 3.1.6.1, 3.1.6.2, 3.1.6.3

Primary Actor: ATM

Pre-Conditions: The User must have already signed into their account.

Post-Conditions: The ATM service will have transferred a specific amount of money from one account to another.

Basic Flow/Main Scenario:

1. The bank user selects to transfer money to another account.
2. The system responds by having the user select the account they want to transfer money to as well as the amount of money to transfer.
3. The bank user chooses an account along with the amount of money to transfer.
4. The system transfers that amount from the origin account to the selected account.

Alternative flow:

* If the amount of money being transferred is zero, the system tells the user to choose a specific amount of money
* If the amount of money being transferred is more than what the account transferring has, the system tells the user that the account does not have the amount specified to transfer.

Exceptions:

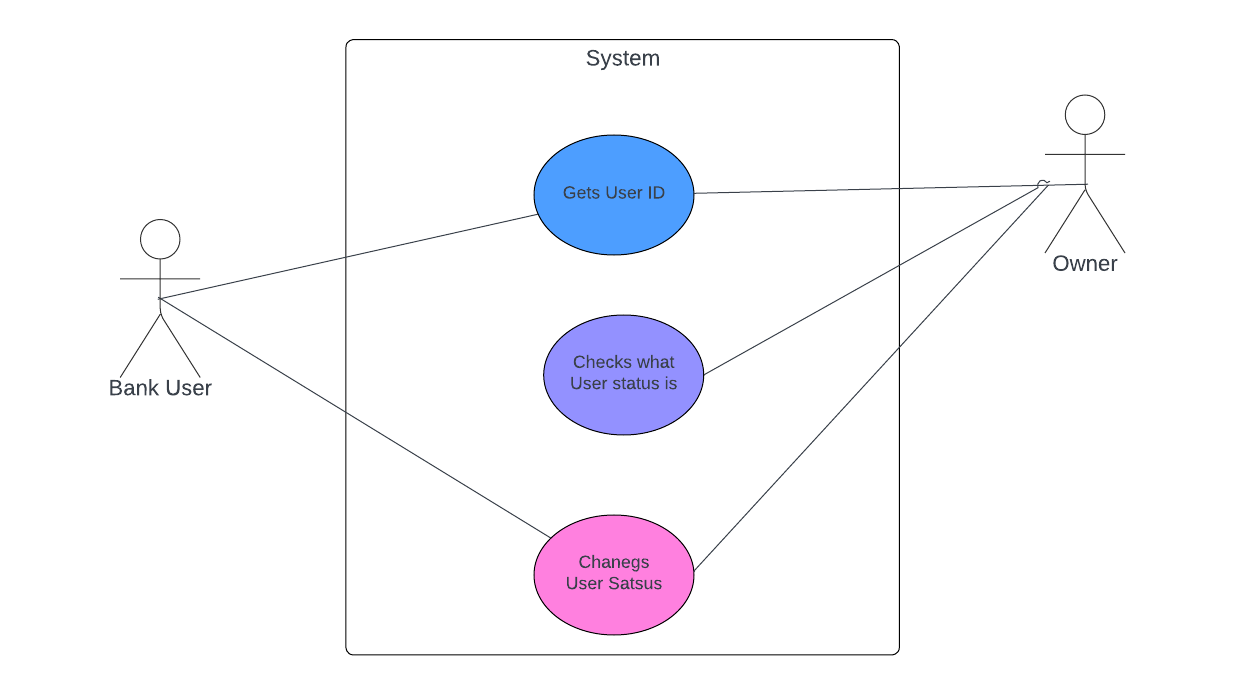
* If the account money is being transferred to or from does not exist, an error message is displayed.

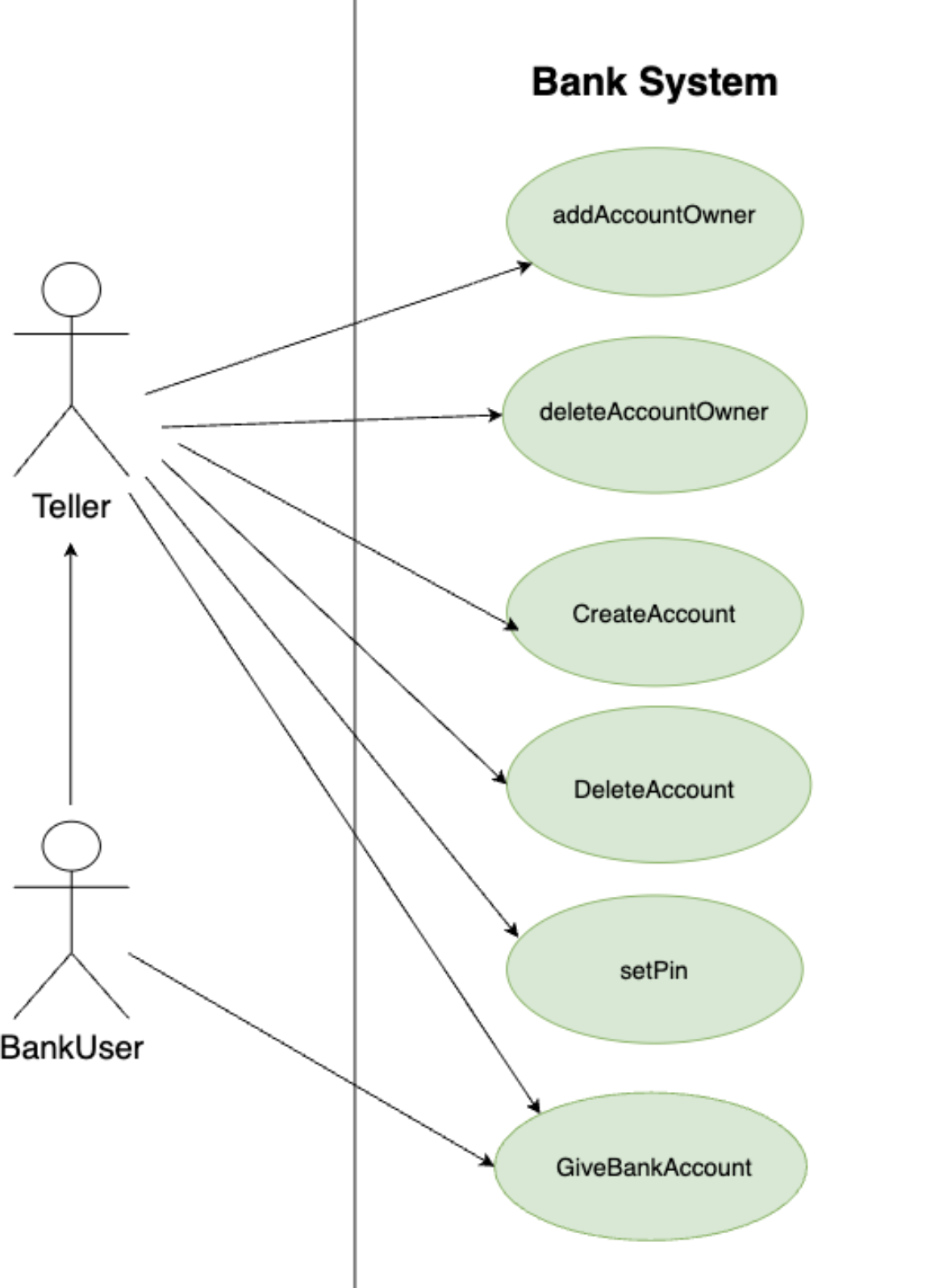
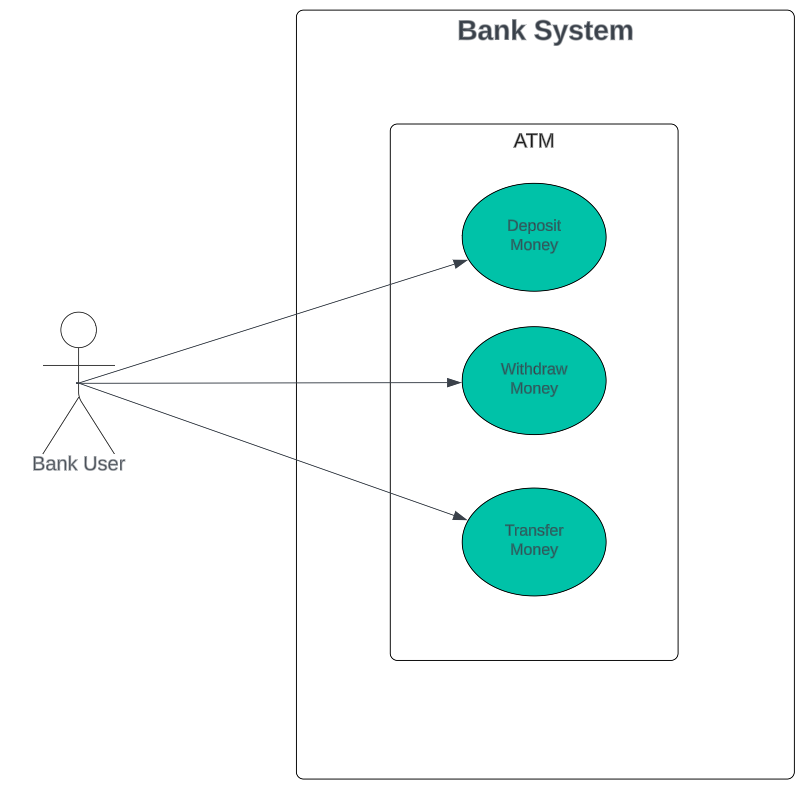
Related Use Cases:

* ATM or Teller window, bank user chooses ATM

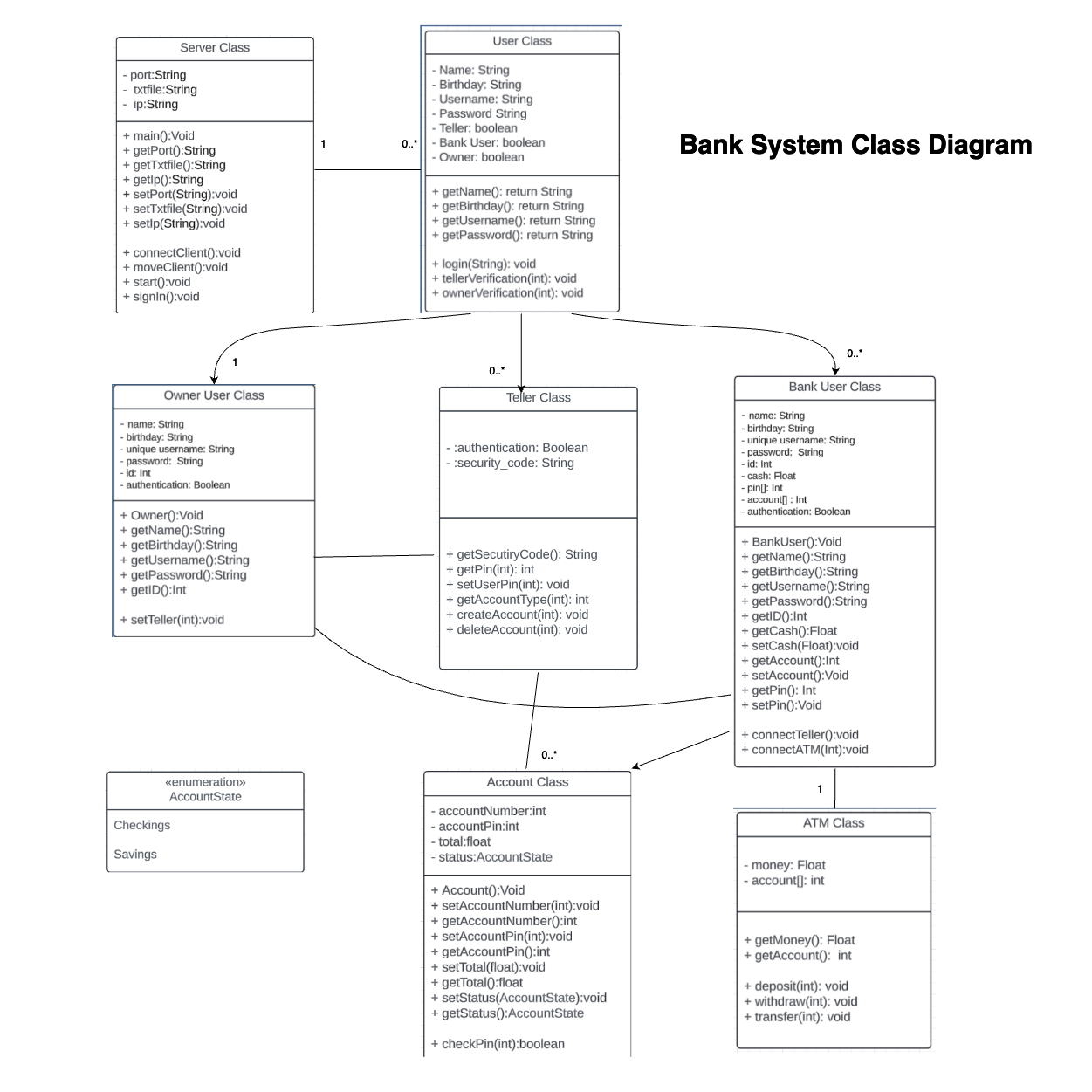


Owner Changes User Status



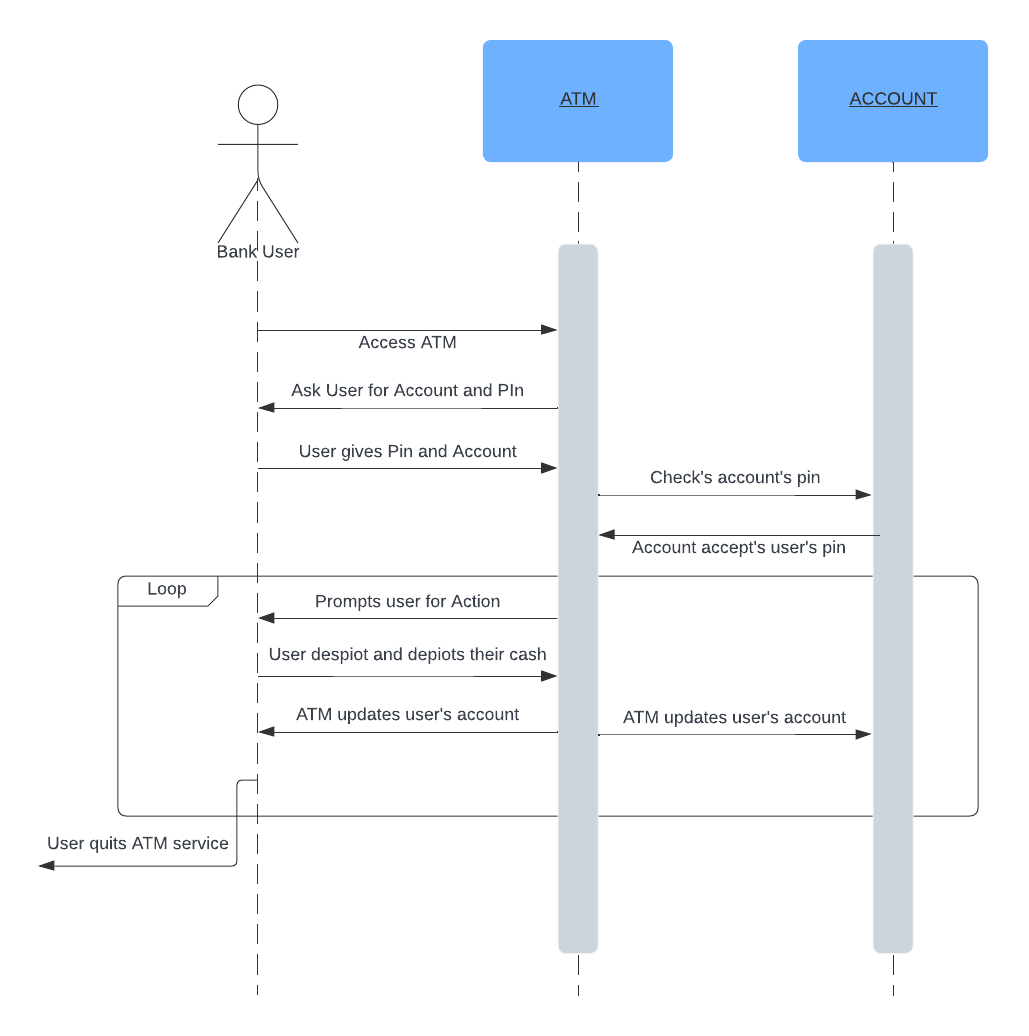




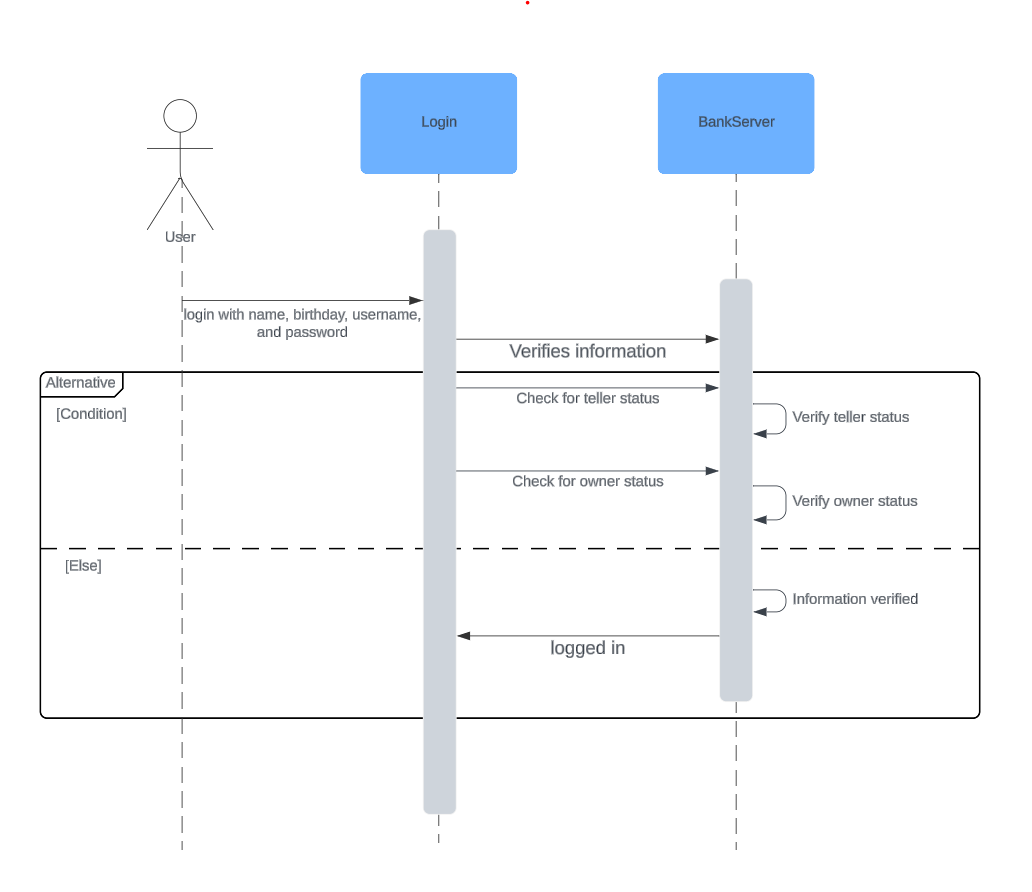


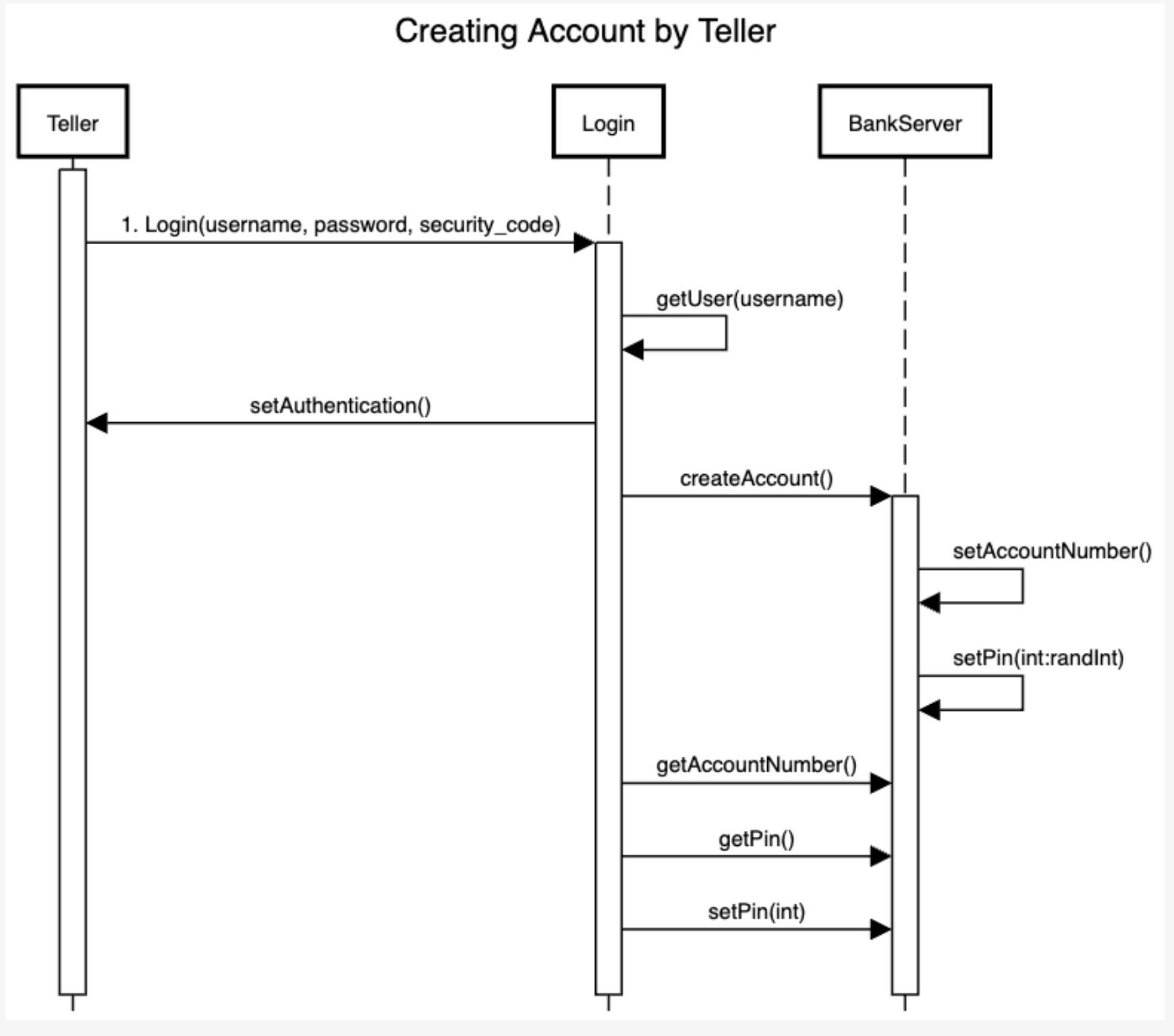


## Bank User makes deposit at ATM



User logs into Bank System





1. The Bank user is a user that is not a teller and is not an owner. They can only access ATMs and ask teller’s for help. The bank user is the simplest a user can be. [↑](#footnote-ref-0)
2. Want to make Tellers have the same functionality as ATM, so anything you can do at an ATM you can also do with a teller, like a real bank. Unsure about adding this to the requirements. [↑](#footnote-ref-1)
3. To prevent fraud of teller’s hiring other tellers, teller’s firing other tellers, tellers adding themselves to any bank account they want. We wanted a set of logins that are unique to owners and instead give the owner the responsibility of hiring and firing tellers. [↑](#footnote-ref-2)
4. Unsure of how exactly multithreading works as of yet. Not sure if this is the route we will go with for in the long run but this is the idea we have right now. [↑](#footnote-ref-3)
5. How much time can not be promised right now, hoping for at least a second [↑](#footnote-ref-4)
6. How much time can not be promised right now [↑](#footnote-ref-5)