**Banking Application**

*Group 3*

Design Specifications

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 3/18/2025 | 1.0 | Initial Version. Created the Document, edited Table of Contents and edited the formatting. | Alexandra Molchanenko |
| 03/18/2025 | 1.1 | Edited Table of Contents and headers | Matthew Zhang |
| 03.18.2025 | 1.2 | Formatted and added UML Diagrams | Matthew Zhang |
| 3/25/2025 | 1.3 | Edited and wrote the Purpose section of the document. | Alexandra Molchanenko |
| 4/7/2025 | 1.9 | Added Updated Class Diagrams | Matthew Zhang |
| 4/7/2025 | 2.0 | Added potential class candidates; Finished Login class functionalities | Matthew Zhang |
| 4/8/2025 | 2.5 | Finished class candidates | Matthew Zhang |
| 4/8/2025 | 2.9 | Updated Class Diagrams | Matthew Zhang |
| 4/9/2025 | 3.0 | Added Sequence Diagrams section | Sophia Bansil |
| 4/10/2025 | 3.5 | Added Use Cases section | Sophia Bansil |
| 4/10/2025 | 4.0 | Added Use Case Diagram | Sophia Bansil |
| 4/10/2025 | 5.0 | Added GUI Prototype Pages; Added more Server Use cases | Alexandra Molchanenko |
| 4/10/2025 | 6.0 | Added Use Case Diagrams | Phakinh Sinlee |
| 4/10/2025 | 7.0 | Added Gantt Chart | Alexandra Molchanenko |
| 4/10/2025 | 8.0 | Replaced Use Case Diagrams | Matthew Zhang |
| 5/6/2025 | 9.0 | Edited Class Candidates and updated class diagrams | Matthew Zhang |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Purpose 6](#_Toc197465044)

[1.1. Scope 6](#_Toc197465045)

[1.2. Definitions, Acronyms, Abbreviations 6](#_Toc197465046)

[1.3. Overview 6](#_Toc197465047)

[2. Class Candidates 7](#_Toc197465048)

[2.1. Login 7](#_Toc197465049)

[2.2. Teller 7](#_Toc197465050)

[2.3. Client 7](#_Toc197465051)

[2.4. ATM 7](#_Toc197465052)

[2.5. Message 8](#_Toc197465053)

[2.6. Server 8](#_Toc197465054)

[2.7. ClientProfile 8](#_Toc197465055)

[2.8. Account 8](#_Toc197465056)

[2.9. Transaction 9](#_Toc197465057)

[2.10. CheckingAccount 9](#_Toc197465058)

[2.11. SavingAccount 9](#_Toc197465059)

[2.12. CreditLine 9](#_Toc197465060)

[3. Use Cases 10](#_Toc197465061)

[3.1. Client Use Cases 10](#_Toc197465062)

[3.2. Teller Use Cases 10](#_Toc197465063)

[3.3. Server Use Cases 11](#_Toc197465064)

[4. Use Case Diagram 12](#_Toc197465065)

[5. Class Diagram 13](#_Toc197465066)

[5.1. Central Server 13](#_Toc197465067)

[5.2. Message Classes 14](#_Toc197465068)

[5.3. Database and Data Objects 15](#_Toc197465069)

[5.4. Teller 16](#_Toc197465070)

[5.5. Client 17](#_Toc197465071)

[5.6. Login 18](#_Toc197465072)

[6. Sequence Diagram 19](#_Toc197465073)

[7. Prototype 28](#_Toc197465074)

[8. Gantt Chart 35](#_Toc197465075)

# Purpose

## Scope

This document explains the design for the Banking Application made by Group 3. It includes diagrams and descriptions that show how the system is set up and how it should work, such as class diagrams, use case diagrams, and sequence diagrams. The goal is to show how the system will support both ATM services for clients and tools for bank employees (tellers). This document focuses on the design, not the actual coding or implementation.

## Definitions, Acronyms, Abbreviations

ATM – Automated Teller Machine

Client – A person using the bank’s services (ATM or account features).

Teller – Bank employee who can manage customer accounts.

UML – Unified Modeling Language.

Overdraft – When someone tries to take out more money than they have.

Autopay – Automatic payments that happen on a schedule.

## Overview

The Banking Application is meant to handle everyday banking tasks for a large bank. It works across different bank branches and helps clients use ATMs and tellers manage customer accounts. The system stores information on a shared server, keeps track of all actions, and helps prevent fraud. It includes features like deposits, withdrawals, account creation, and automatic payments.

# Class Candidates

## Login

* + 1. Allow users to login as a Teller or Client
    2. Allow users to input username and password for authentication
    3. Login method checks inputted username and password with stored data (accessed over the network)
    4. Allows Teller or Client to access corresponding applications on successful login authentication
    5. Handles failed logins and tracks failed login attempts

## Teller

* + 1. Holds ProfileMessage and AccountMessage objects allowing teller to see displayed information without exposing internal bank logic
    2. Creates a new profile for a client
    3. Can add a new checking account, saving account, or line of credit to a client’s profile
    4. Can edit the client’s personal information
    5. Remove a checking, savings, or line of credit from client’s account
    6. Can view a client’s account and edit a bank account’s information
    7. Can make transactions on a client’s bank account by using the client’s profile data
    8. Client data is saved after alterations are made to client bank information
    9. Client is sent to and from the server using Message class instances

## Client

* + 1. Can view all accounts that are under ownership of the client (including accounts shared to them by other users)
    2. Can select a bank account

## ATM

* + 1. Clients are prompted an ATM GUI after selecting a bank account from displayed bank accounts on their Client Profile GUI
    2. Clients can make withdrawals or deposits on their bank account
    3. ATM will save transactions made to the account by sending the data via Message over the network to the Server after the transaction is made
    4. Saving will occur when client exits the ATM GUI whether through manually exiting or after a period of inactivity
    5. ATM will close the connection after a period of inactivity and return to login screen

## Message

* + 1. An abstract class which defines the attributes every unique message subclass will abide by
    2. Contains an enumeration TYPE which lists all the various message types which determine their purpose
    3. Holds an attribute of type which determines the response of the server or client when received
    4. Holds sessionInfo which contains the client’s session ID (for identification), username, and activity

## Server

* + 1. Maintains a list of ClientProfile instances which holds the client’s bank account information and bank accounts
    2. Maintains a list of valid Teller login information
    3. Will have a clientHandler class imbedded within the class to handle multiple clients
    4. Will authenticate incoming requests to connect with server; If authenticated, will return a message indicating success
    5. Will send requested bank data to authenticated clients using Message classes
    6. Will remove accounts from database if requested from authenticated client
    7. Will store the data on a plaintext file
    8. Will load data from a plaintext file by parsing each line of the plaintext file

## ClientProfile

* + 1. Holds String attributes for client’s username and password
    2. Holds a String attribute containing client’s phone number
    3. Holds a String attribute containing client’s address
    4. Holds a String attribute containing client’s legal name (first and last)
    5. Holds a boolean attribute that checks for whether the account is in use
    6. Holds an array of bank accounts that the client may own

## Account

* + 1. Each account instance holds a unique ID used for tracking and searching
    2. Each account has a boolean “check” which will be turned on if the account is in use; other users cannot access the account if in use
    3. Each account has a float balance which will track the amount of money stored in the account
    4. Each account will have an array of transactions which records all withdrawals and deposits made to the account

## Transaction

* + 1. Holds a Date instance which tracks the date the transaction was made or created
    2. Records amount of money being transferred
    3. A note is made to state whether the transaction was a deposit, withdrawal, or another type of transaction

## CheckingAccount

* + 1. Subclass of the Account Class

## SavingAccount

* + 1. Subclass of the Account Class
    2. Integer attribute that holds limit of withdrawals that can be made
    3. Method will track number of withdrawals and prevent further withdrawals if limit is reached
    4. A monthly timer that will reset the number of withdrawals made back to zero when a transaction is made

## CreditLine

* + 1. Subclass of the Account Class
    2. Tellers can only create a line of credit for customers who already have a checking account with at least $1000 in balance
    3. Tellers will set the credit limit when creating a new credit line for the client
    4. Transactions will only go through if the client has not exceed credit limit

# Use Cases

## Client Use Cases

* + 1. Client Log In
    2. ATM Log Out
    3. View ATM Home Page
       1. View all accounts
       2. View own profile info
       3. Choose account to perform actions with
          1. Deposit - Checking
          2. Deposit - Savings
          3. Deposit - Credit
          4. Withdraw - Checking
          5. Withdraw - Savings
          6. Withdraw - Credit
          7. View Account History
          8. Exit current screen

## Teller Use Cases

* + 1. Teller Log In
    2. Teller Log Out
    3. Create New Client Profile
    4. Lookup Client Profile
    5. View Client Profile Overview Page
       1. View all Client accounts
       2. View Client profile info
       3. Edit Client Profile info
       4. Create New Account for Client
          1. Create Checking Account
          2. Create Joint Checking Account
          3. Create Savings Account
          4. Create Credit Account
          5. Exit current screen
       5. Choose account to perform actions with
          1. Deposit - Checking
          2. Deposit - Savings
          3. Deposit - Credit
          4. Withdraw - Checking
          5. Withdraw - Savings
          6. Withdraw - Credit
          7. View Account History
          8. Close Account
          9. Exit current screen
       6. Client Profile Log Out

## Server Use Cases

* + 1. Load data from database into Server
    2. Save Data to Persistent Format
    3. Handle incoming messages
    4. Check Message TYPE
    5. Add request to Threadpool Queue
       1. Assign request to Short-lived thread
    6. Authenticate Login credentials
    7. Validate Session ID
    8. Assign Session ID
       1. add ID and username to sessionInfo map
    9. Delete Session ID
    10. Log Account Access Event
    11. Log Login Attempt
    12. Add to userProfiles
    13. Search userProfiles
        1. Modify Account Balance
        2. Set account activity
        3. Add Account to Client Profile in userProfile
        4. Check for Checking Account with $1000 balance (for creating credit account)
        5. Delete Account from Client Profile in userProfile
        6. Create Joint Account for 2 users
    14. Search TellerProfiles

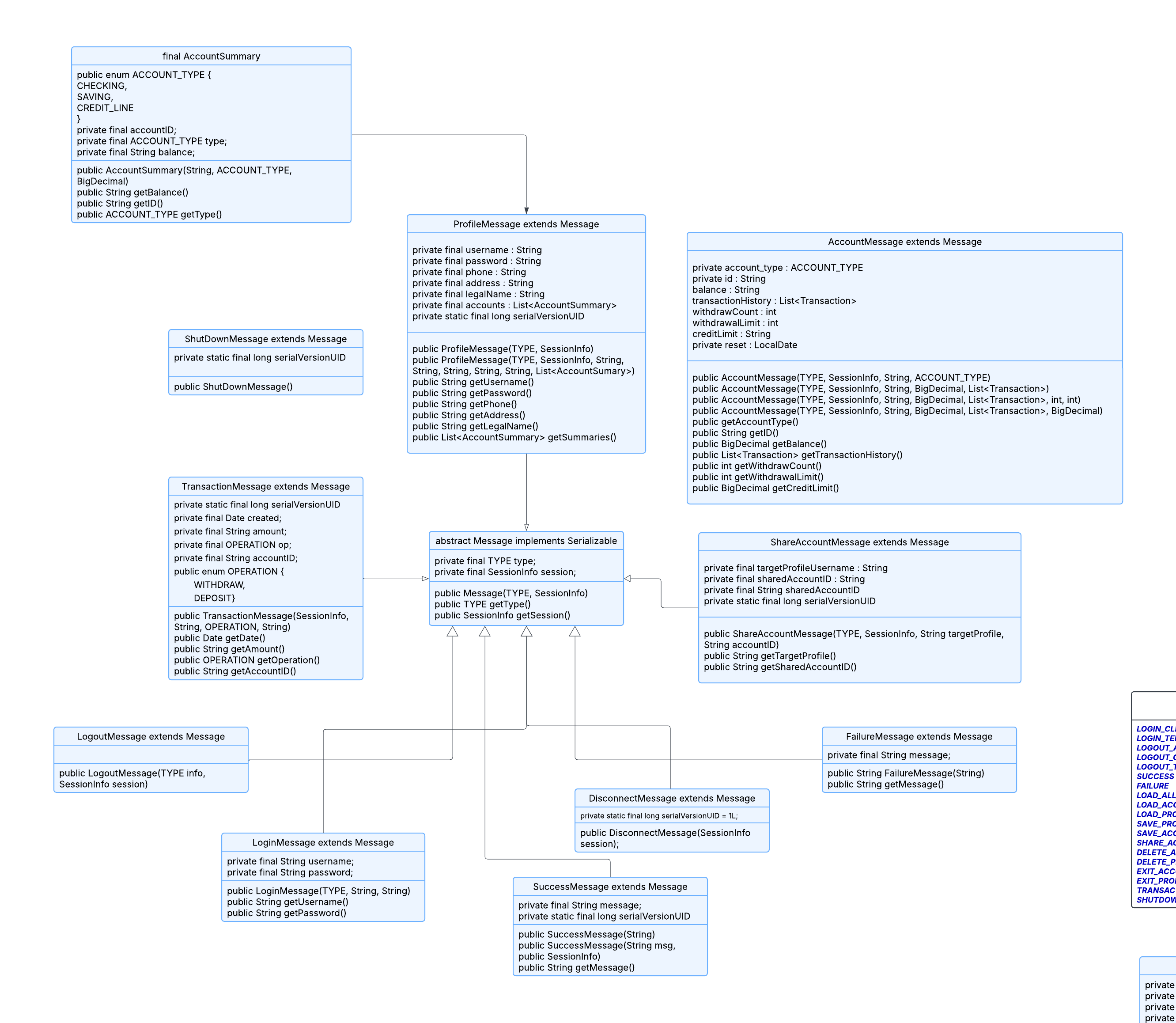
# Use Case Diagram

****

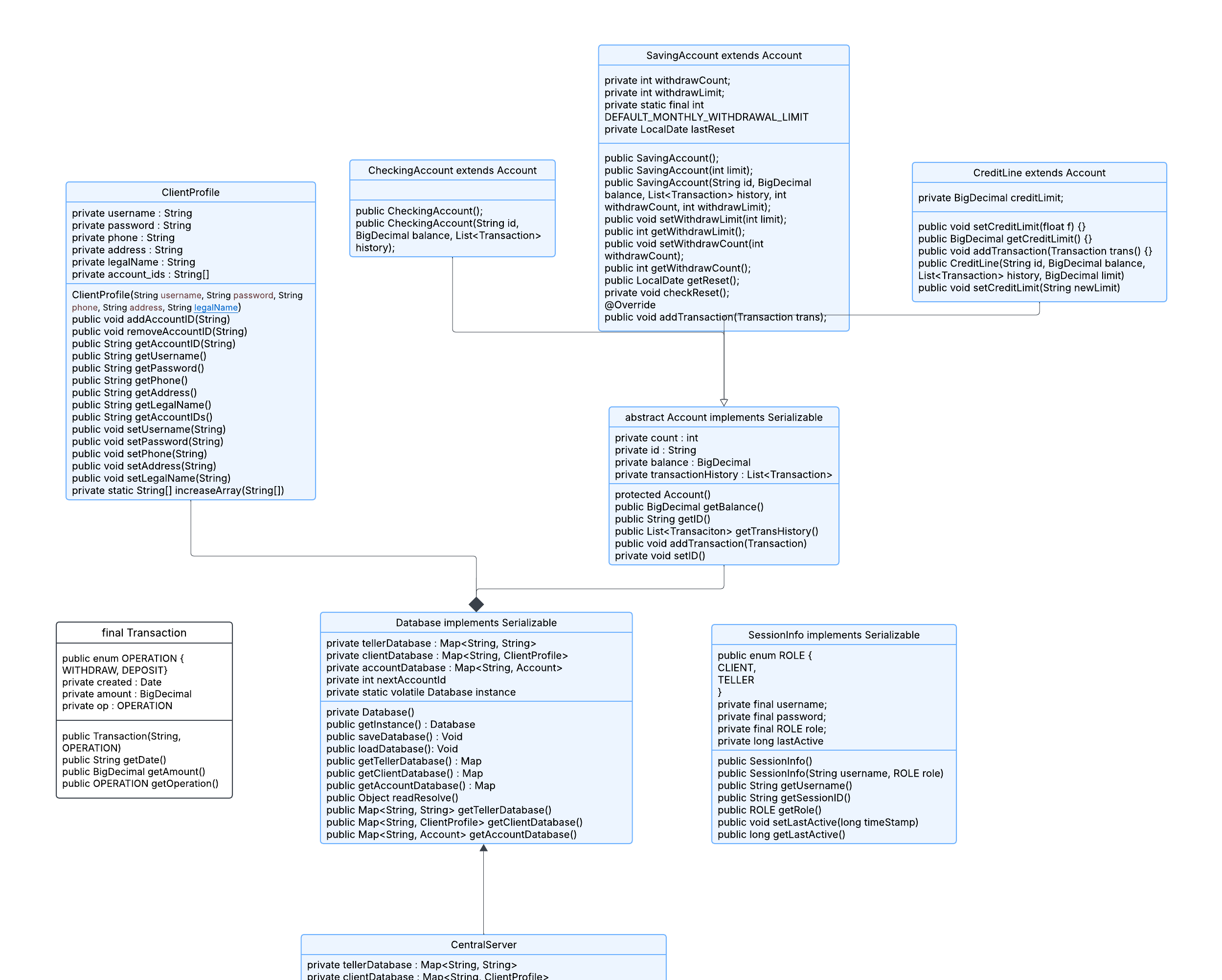
# Class Diagram

## Central Server

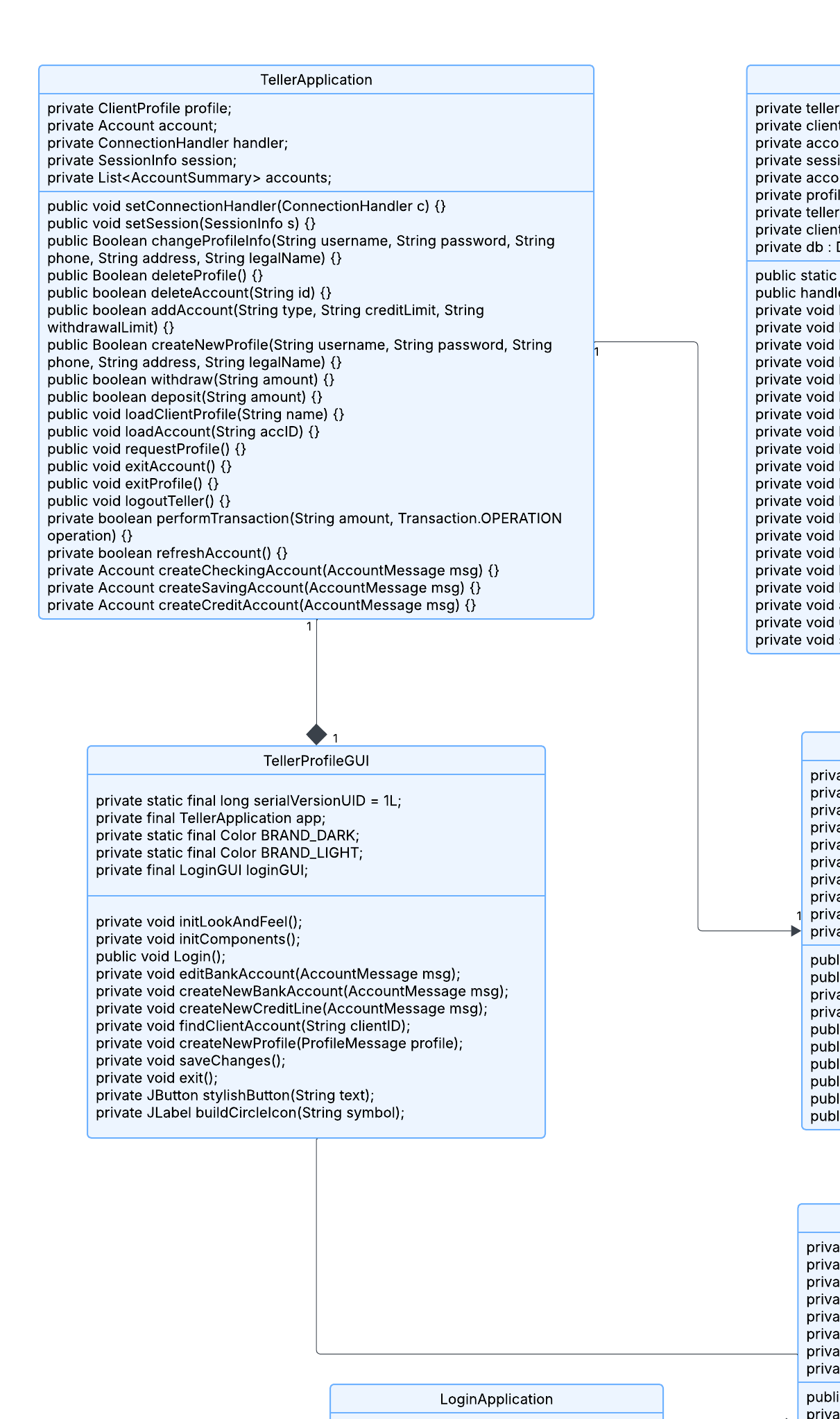
## Message Classes



## Database and Data Objects



## Teller

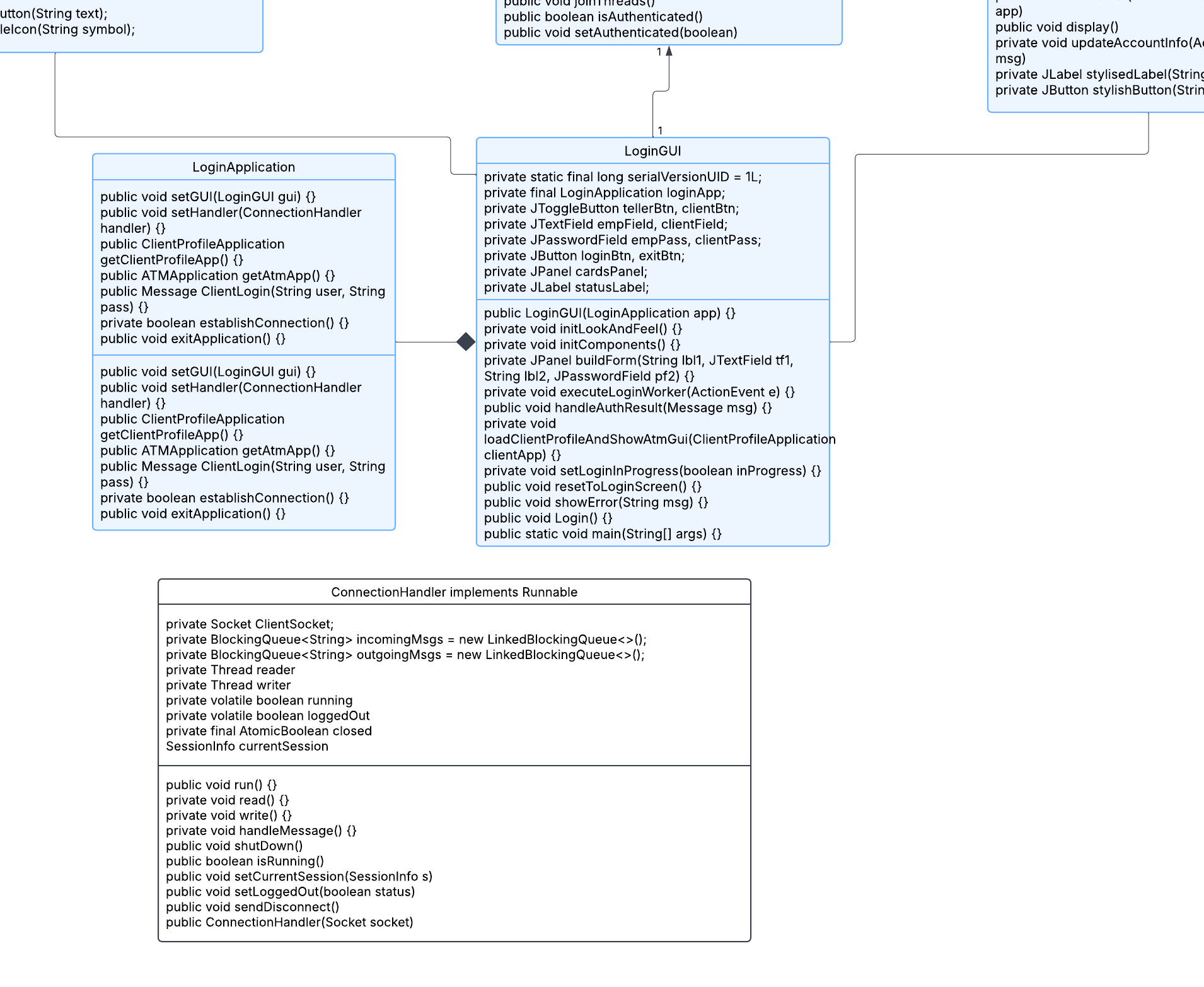


## Client

A diagram of a server

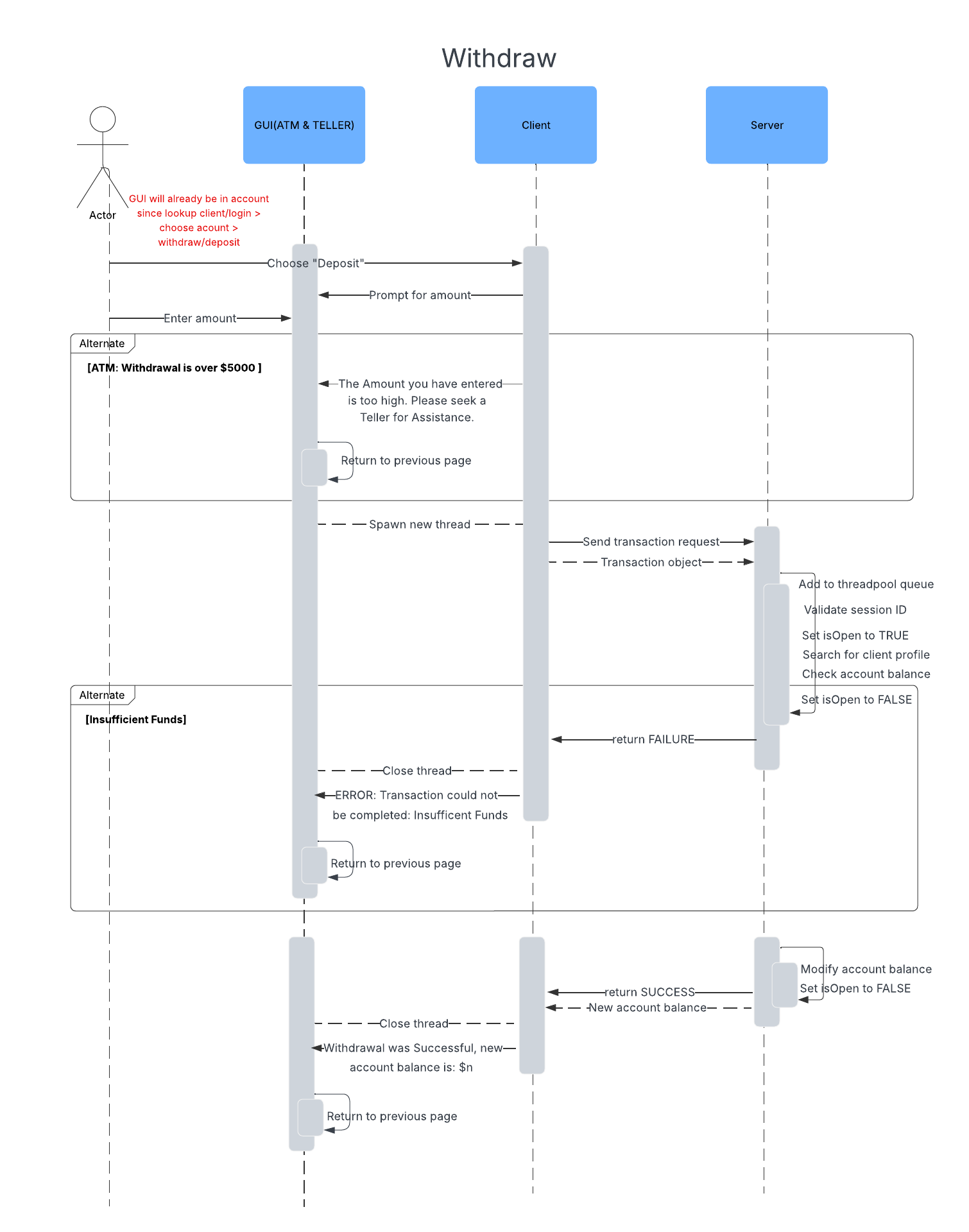
AI-generated content may be incorrect.

## Login

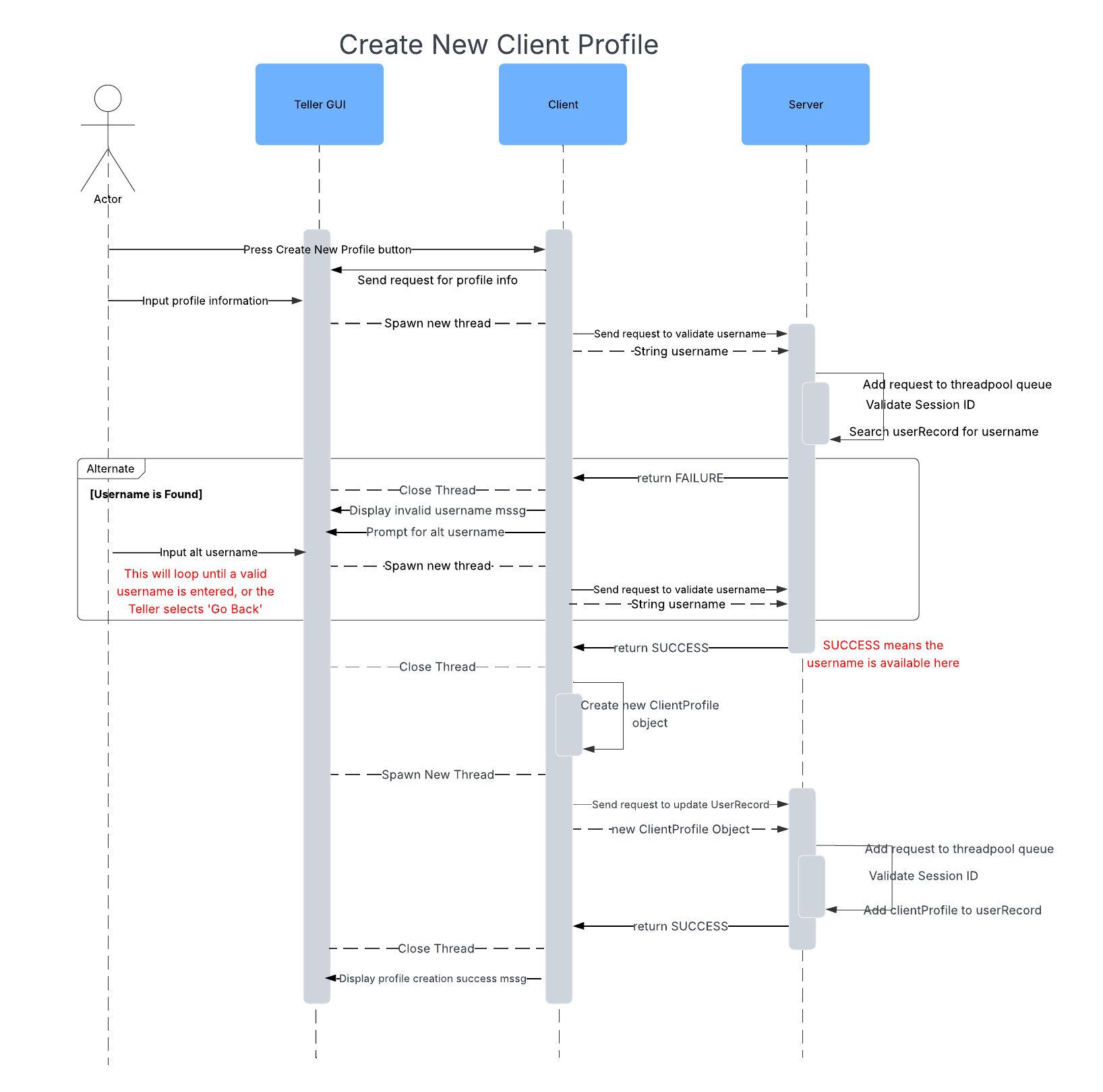


# Sequence Diagram

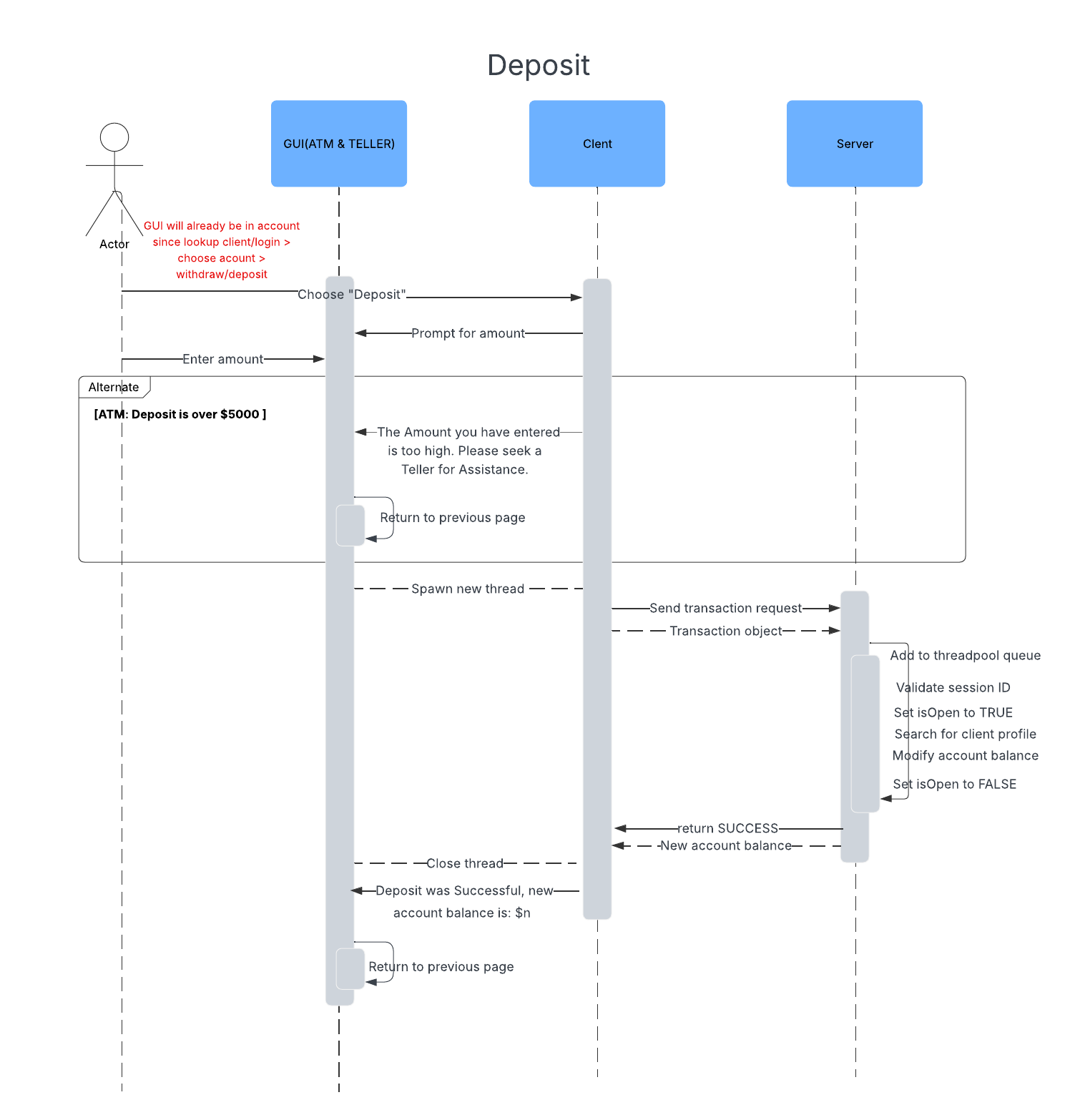
**6.1 Withdraw**

****

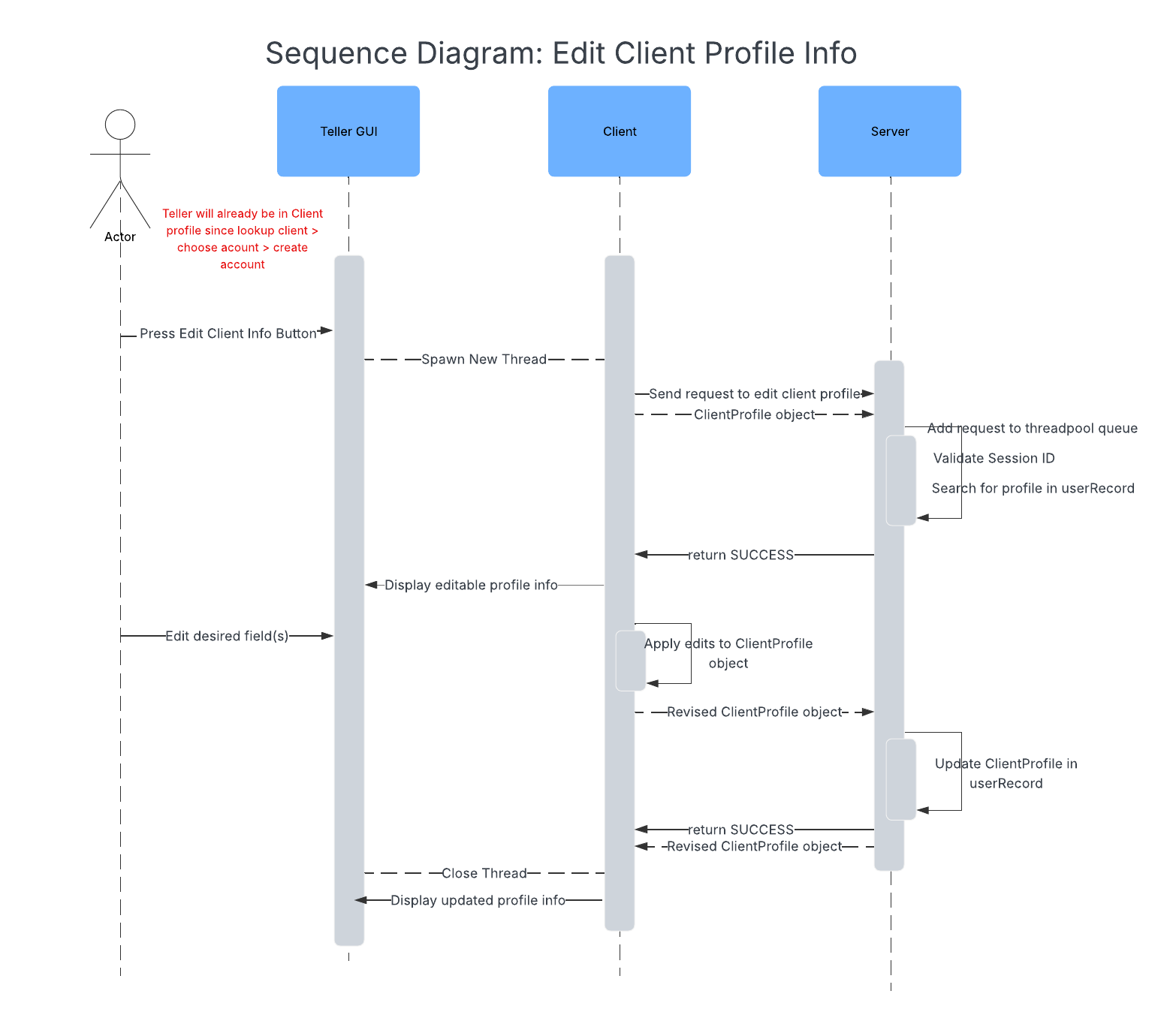
**6.2 Create New Client Profile**

****

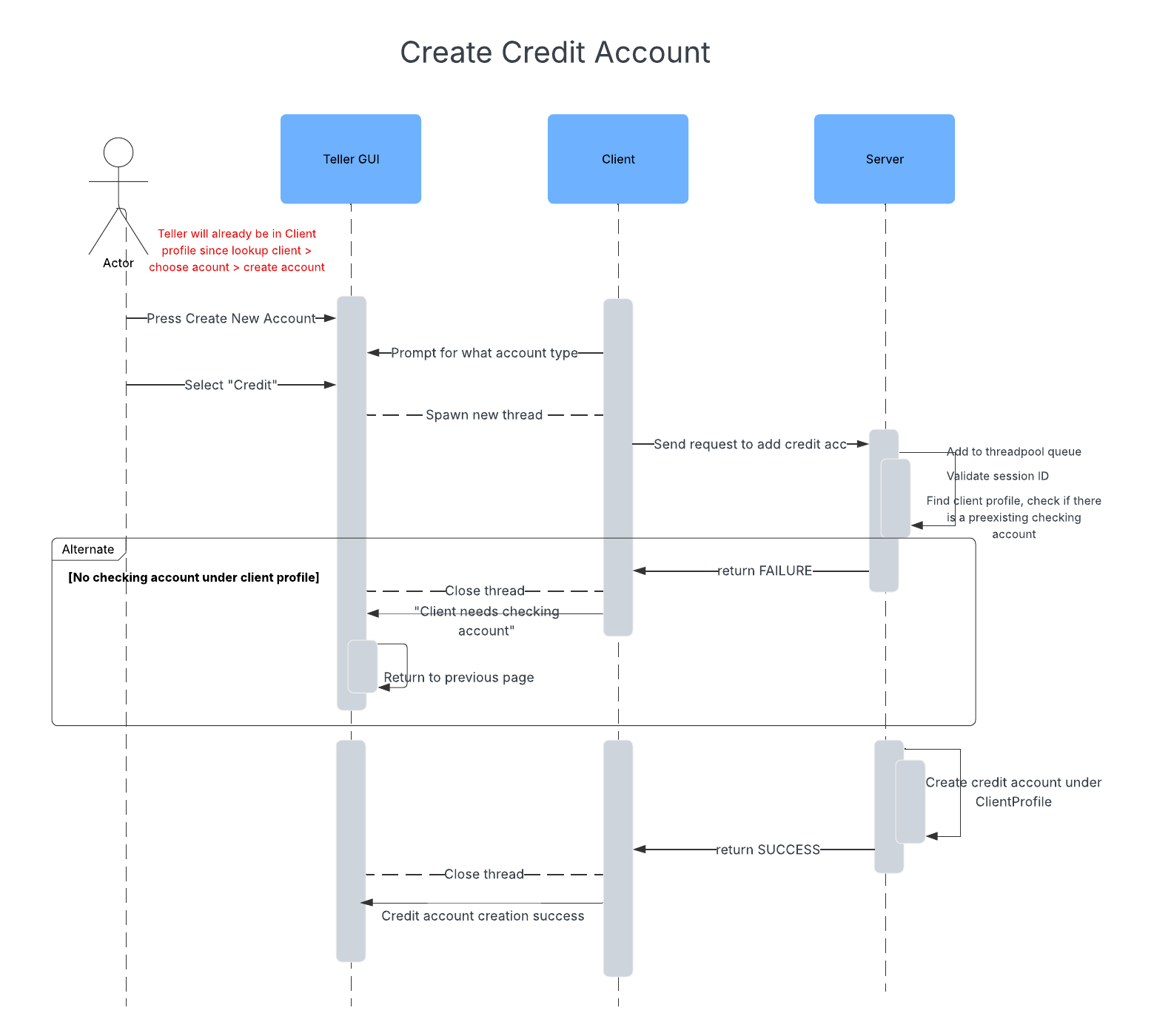
**6.3 Deposit**

****

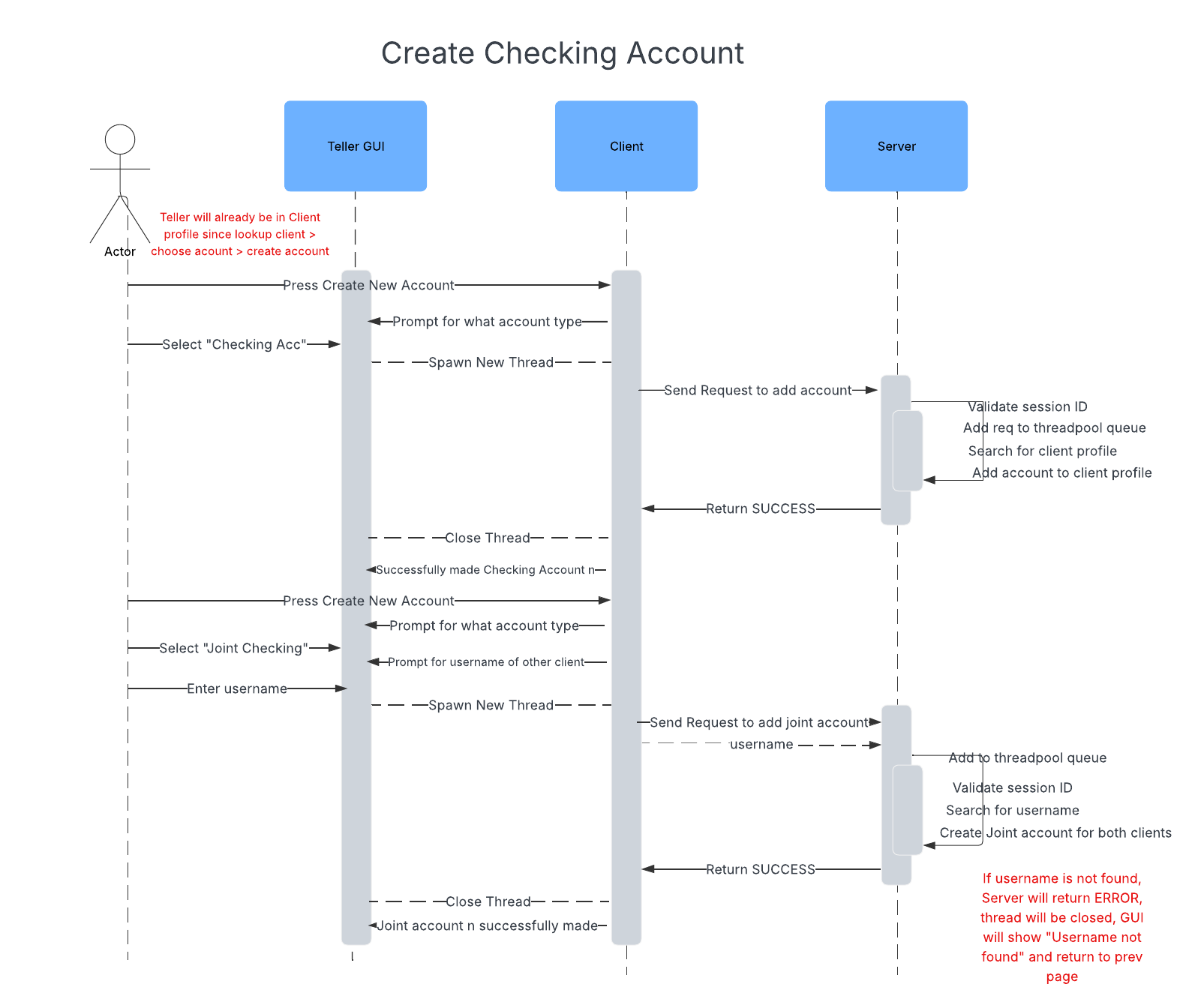
**6.4 Edit Client Profile Info**

****

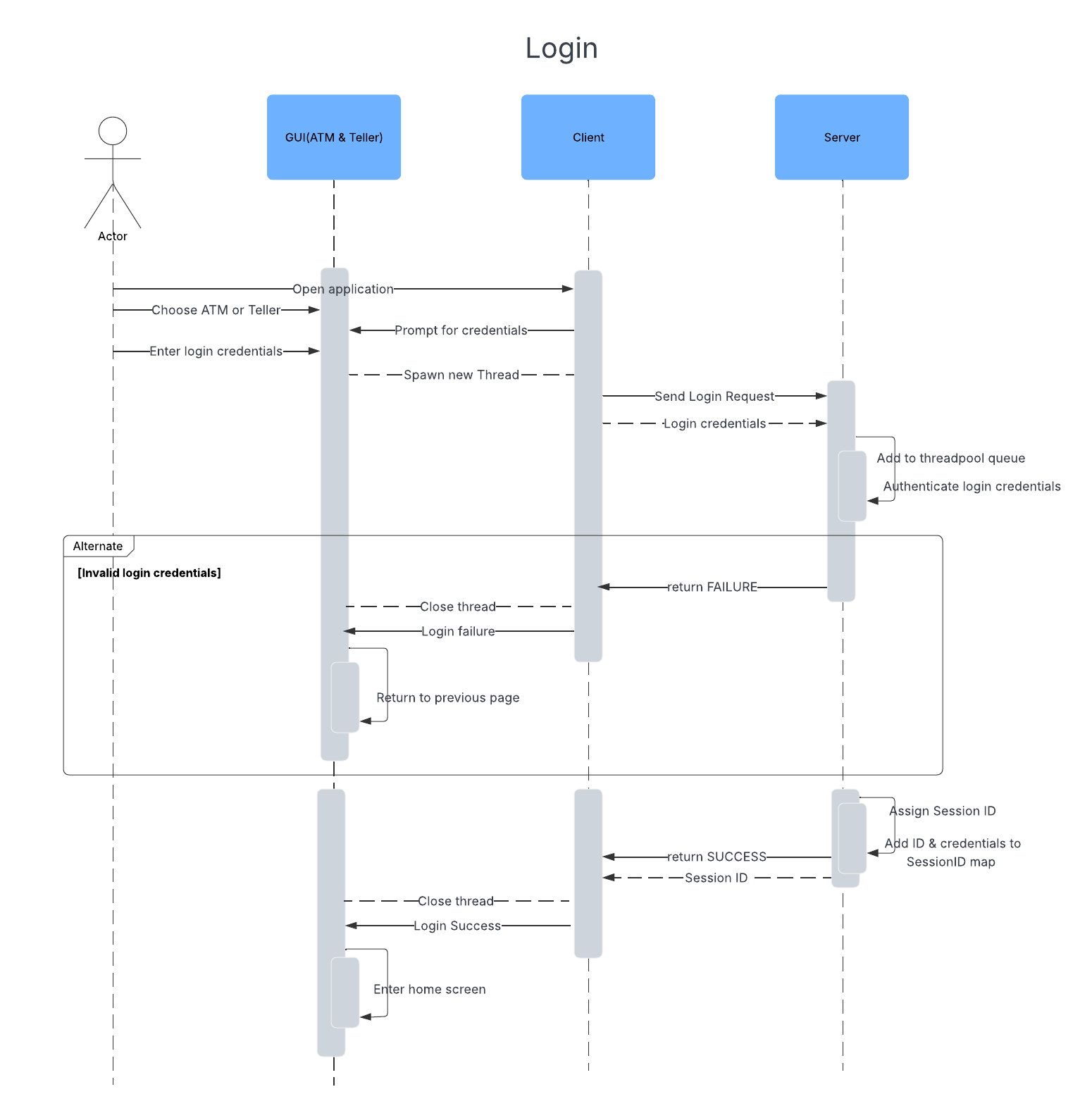
**6.5 Create Credit Account**

****

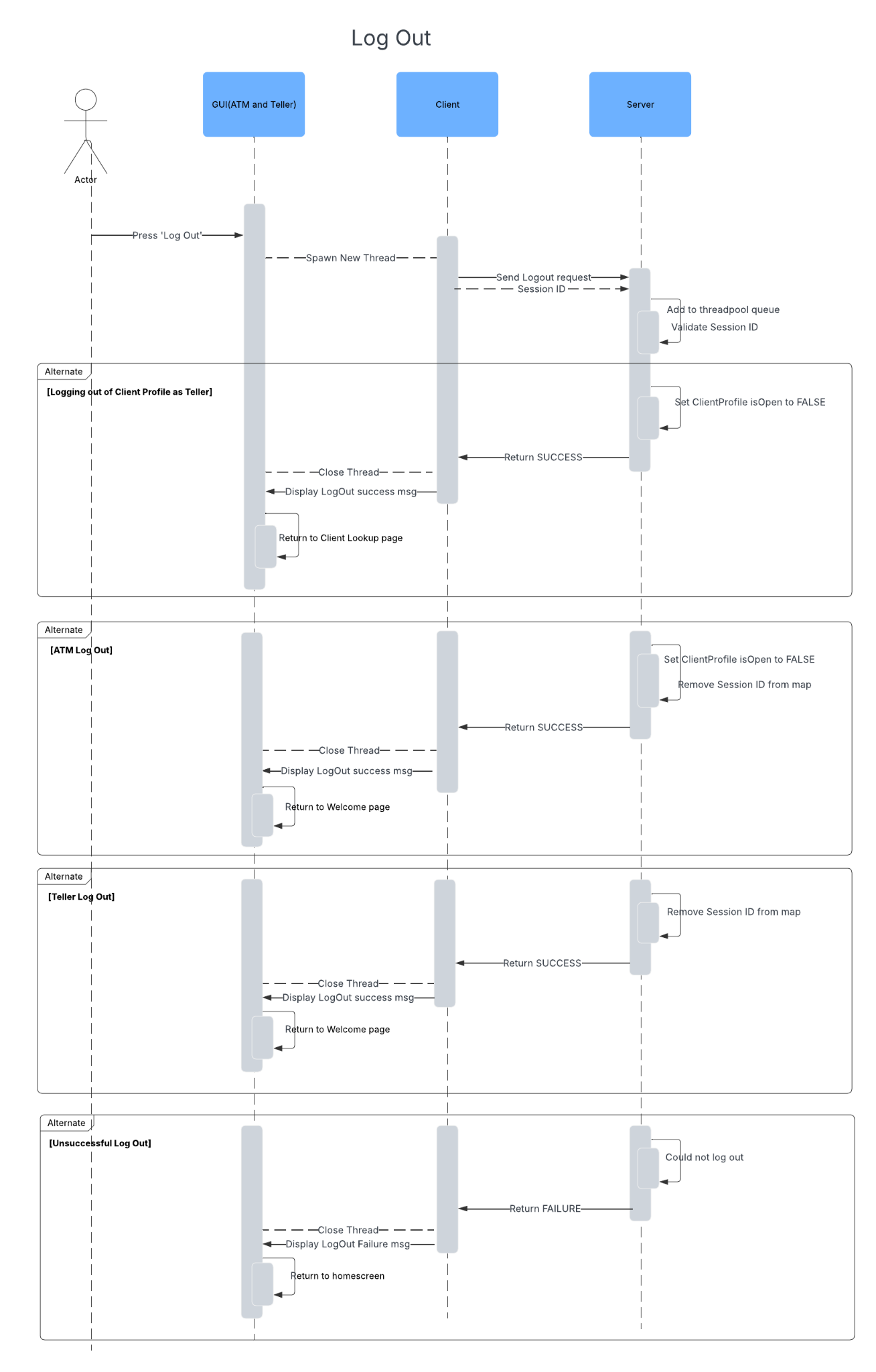
**6.6 Create Checking Account**

****

**6.7 Log In**

****

**6.8 Log Out**

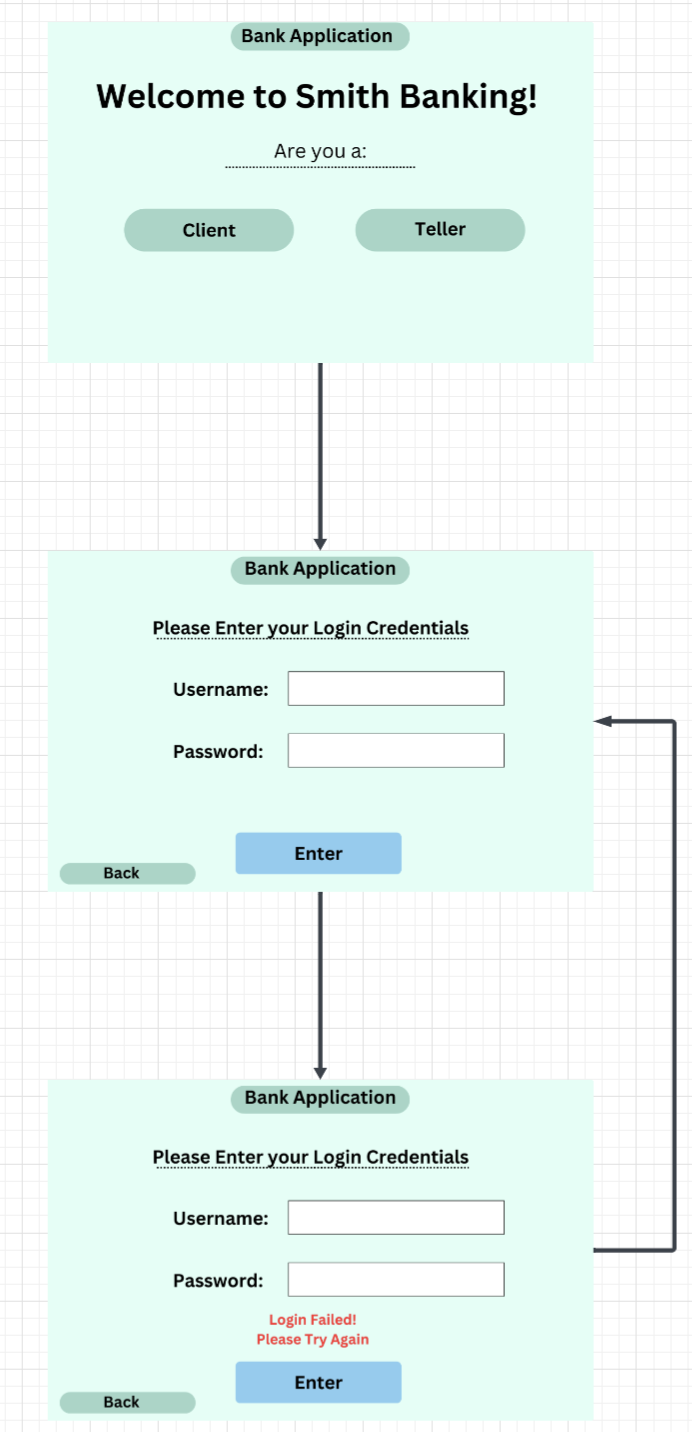
****

**6.9 Delete Account**

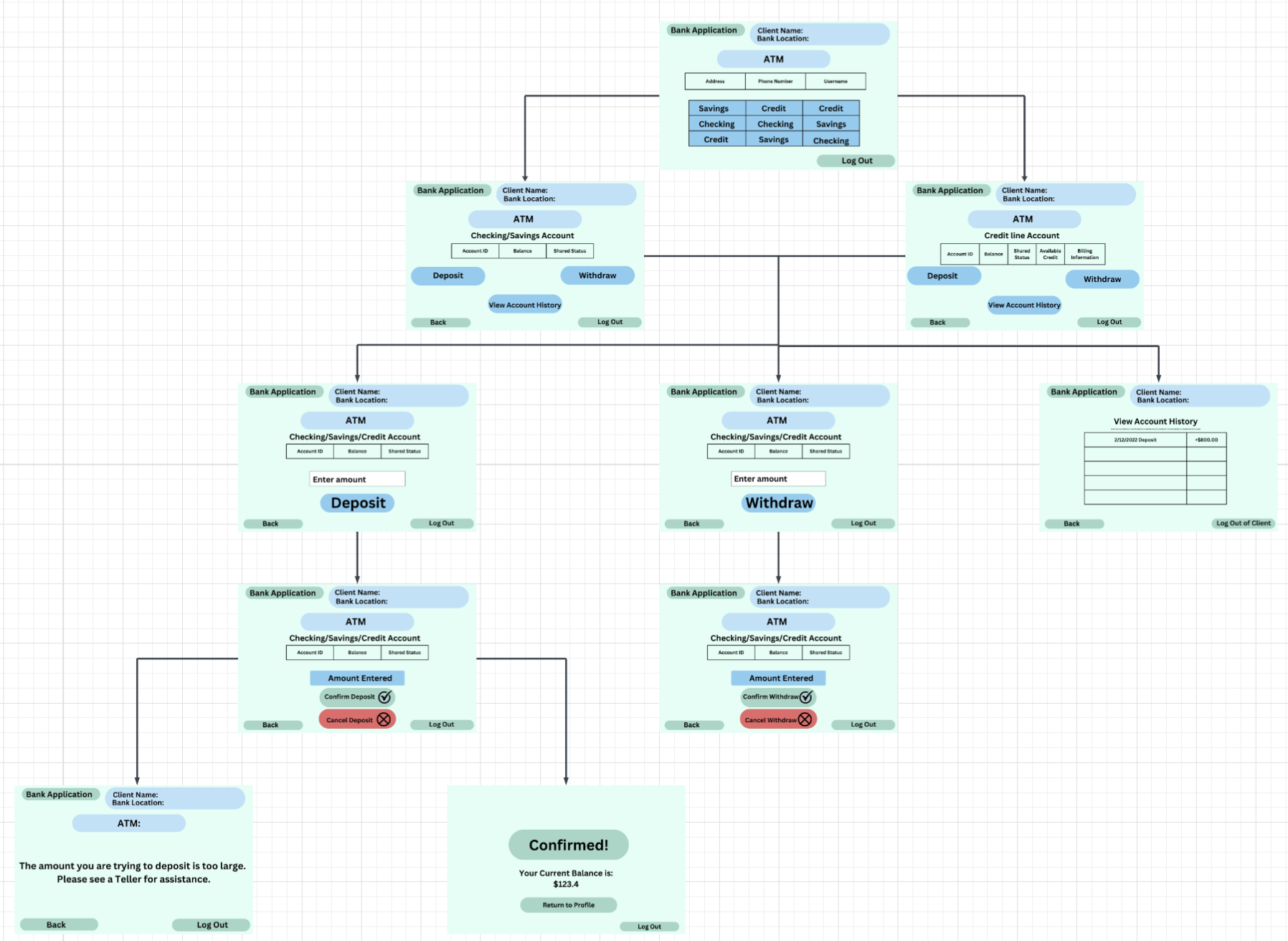
****

# Prototype

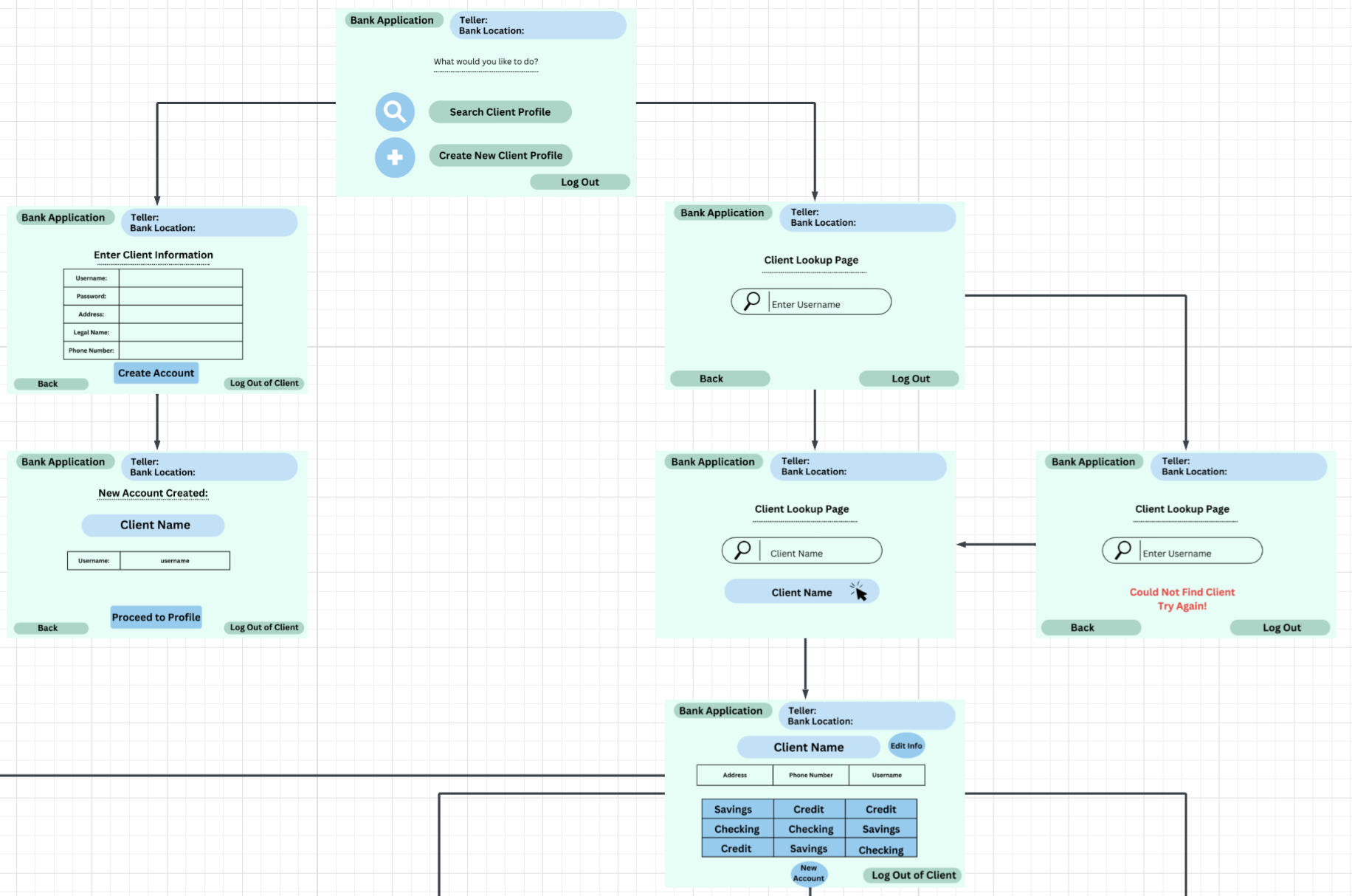
**7.1 Home page and Login**

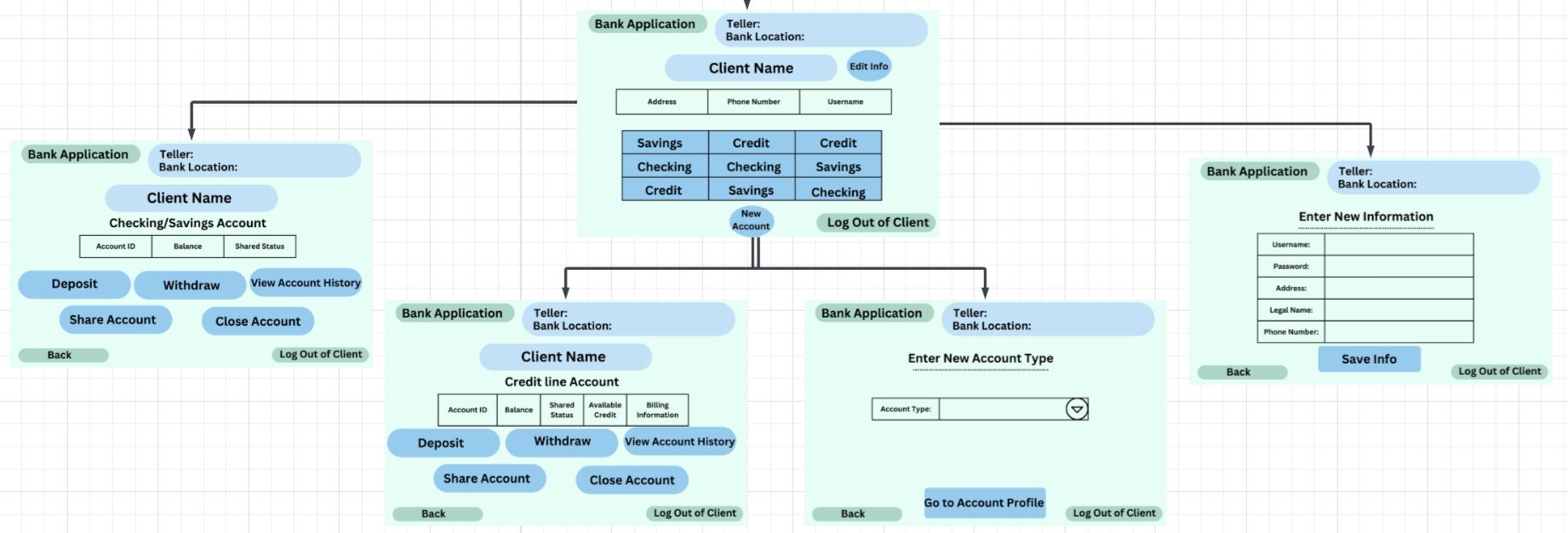
****

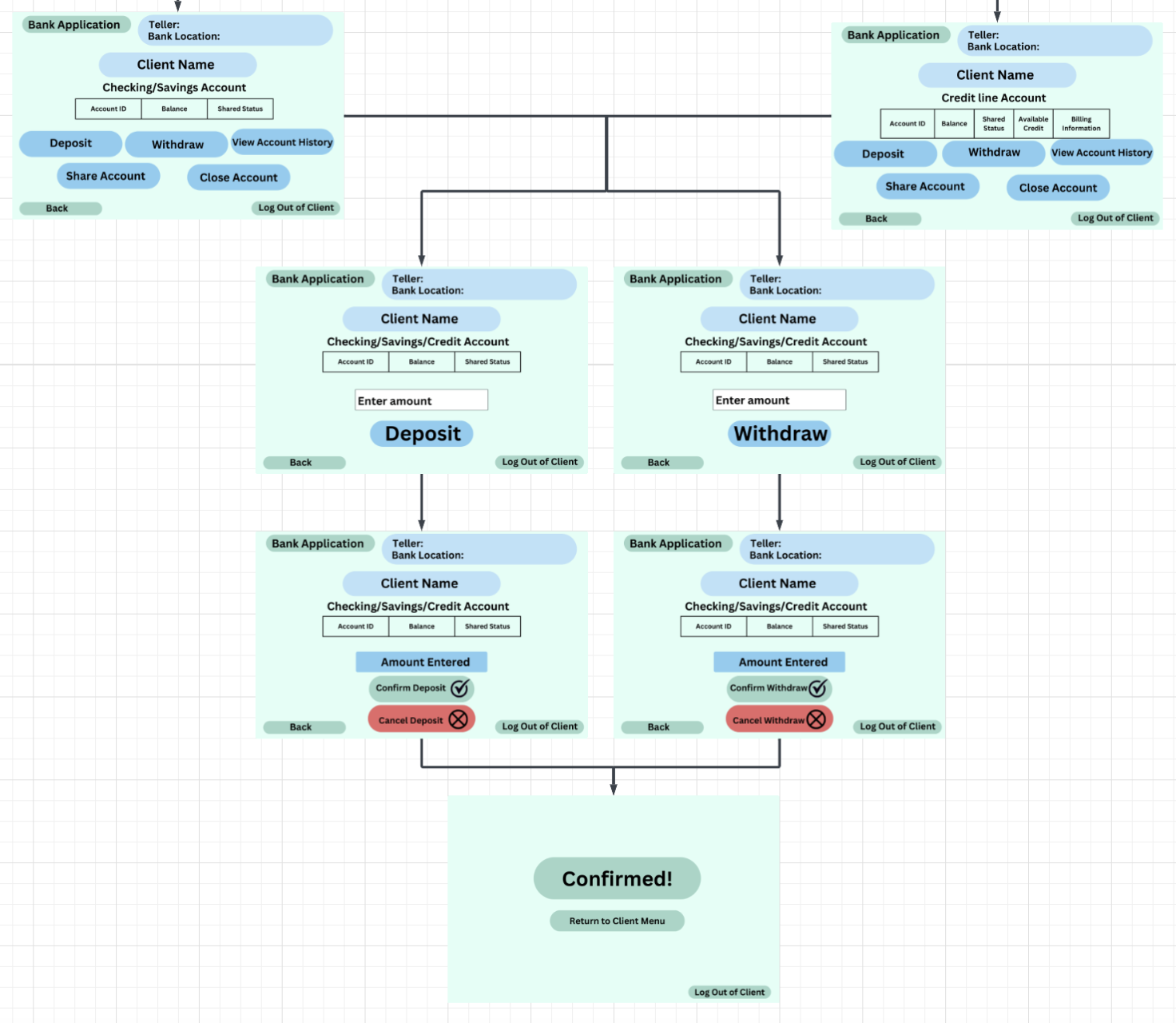
**7.2 ATM Pages**

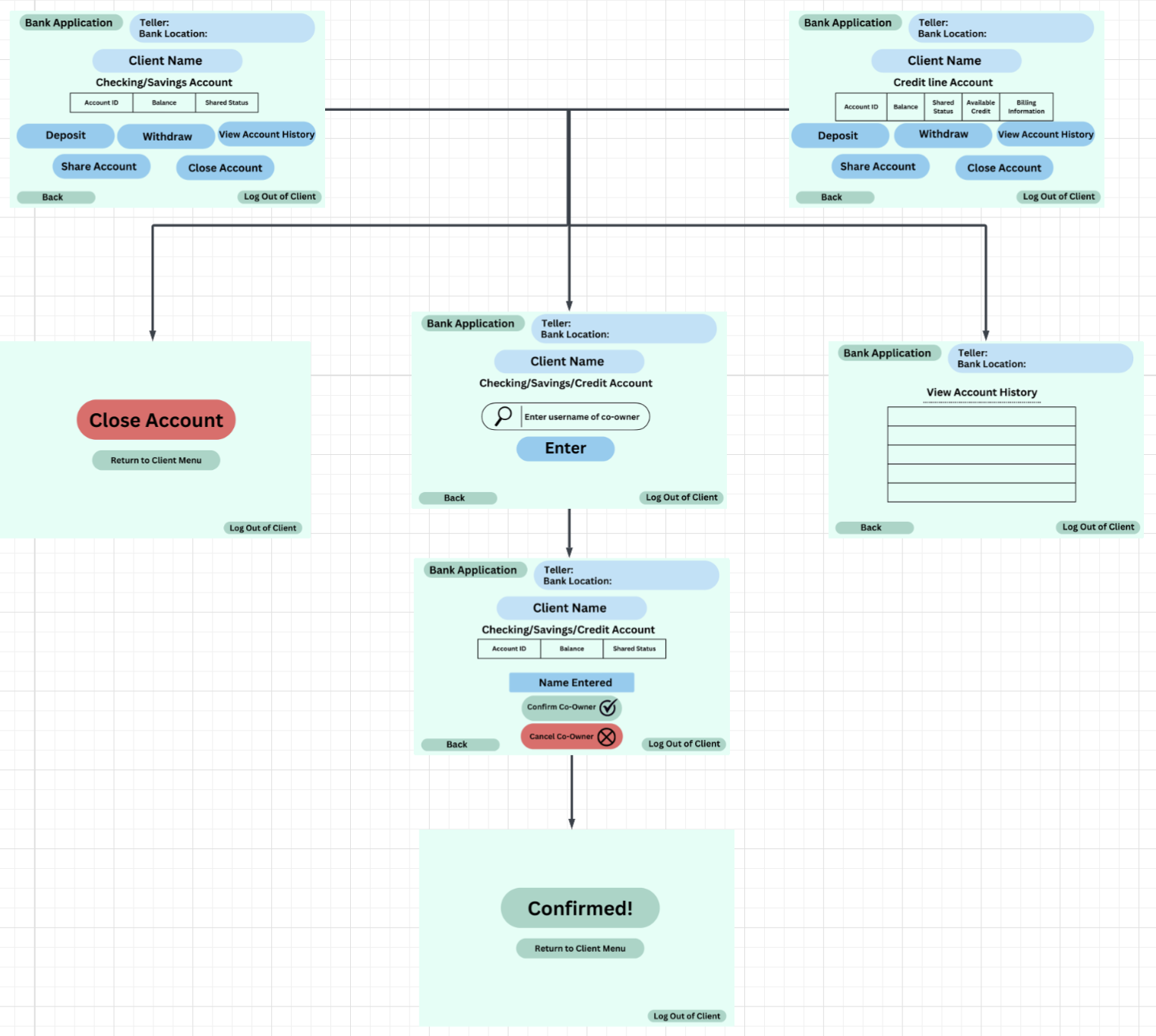
****

**7.3 Teller Pages**

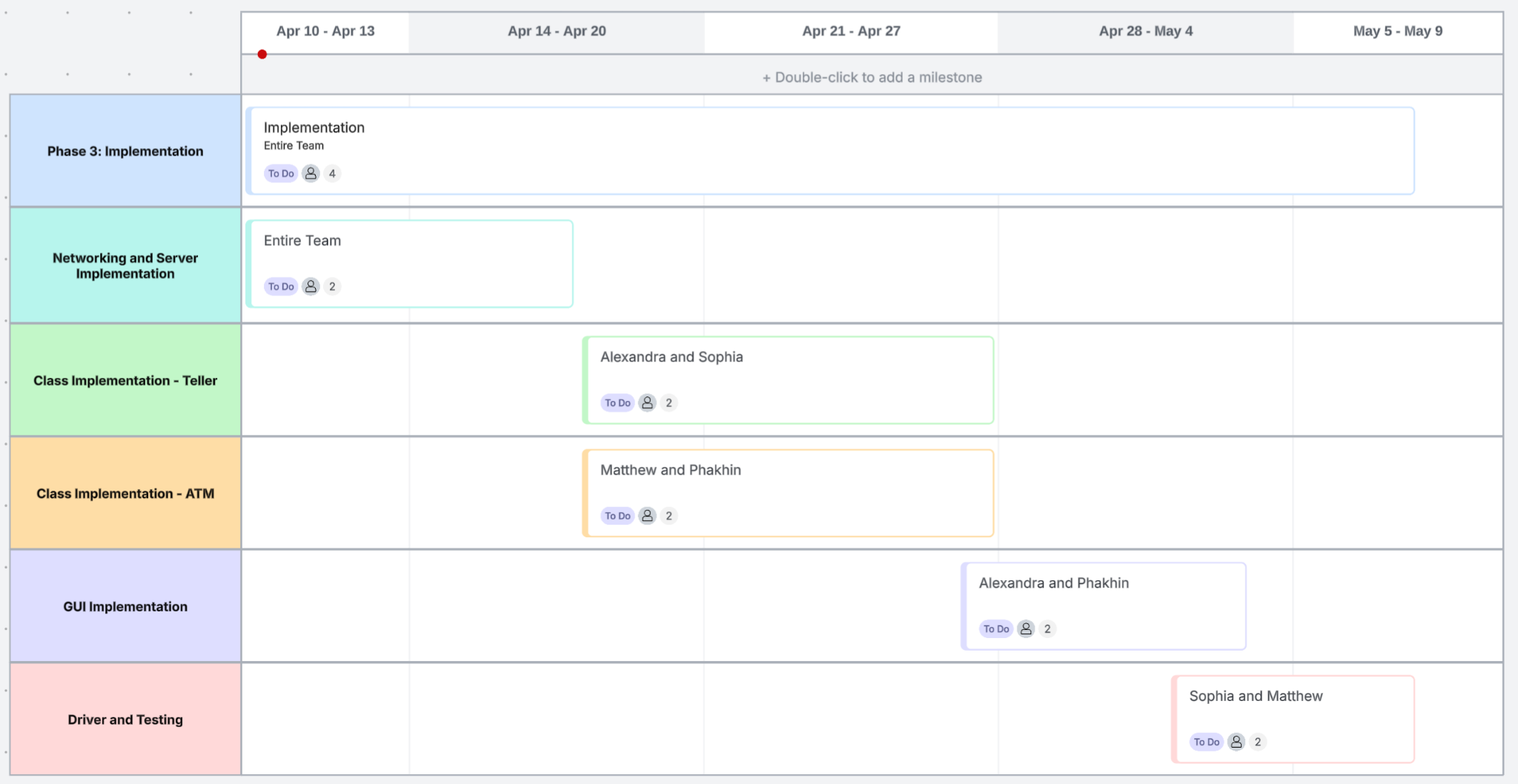
****

****

****

****

# Gantt Chart

****