

Banking - Group 6

Software Requirements

Specification

Revision History

Date	Revision	Description	Author
2/13/2024	1.0	Initial Version	Justin Dodemaide
2/20/2024	1.1	Added some Functional and Module Requirements	Mateo Tolentino
2/23/2024	1.2	Changed modules, added assumptions + functional requirements + internal/external interface requirements, made class diagram, made sequence diagram	Justin Dodemaide
2/25/2024	1.3	Use cases	Abdoul Camara
2/28/2024	1.3.5	Edited Use Cases	Abdoul Camara
2/28/2024		Use case diagram in progress.	Abdoul and Nico
2/28/2024	1.4	Added Overview, Product Perspective, and Constraints	Mateo Tolentino
2/20/24-2/28/24	1.2, 2.3 & 4	Definition, Acronyms, abbreviation Product Functionality/Features Non- functional Requirements	Kebron Afework
3/14/2024		Sequence diagrams for use cases	Kebron, Justin

	Date	Description	Author
	3/14/2024	Use Case Diagram	Kebron Afework
	3/17/2024	Overarching Class Diagram	Abdoul Camara
	3/19/2024	User Account Module Class Diagram	Mateo Tolentino
	3/19/2024	Overarching Class Diagram	Nico Savaides
	3/21/2024	State Machine Class, Overarching Class Diagram, Sequence Diagram	Justin Dodemaide
	3/21/2024	Use Cases, Use Case Diagram, Sequence Diagram, Worked on Slides	Abdoul Camara
	3/22/2024	Sequence Diagram, Overarching Class Diagram, Worked on Slides	Mateo Tolentino
	3/26/2024	Gantt Chart, Worked on Slides	Nico Savaides
	3/26/2024	Worked on Slides, Gantt Chart	Kebron Afework

	04/04/24- 04/18/24	Work on the main control interface/buttons/transitions to interface, State Machine, some parts of the GUI, and Account service.	Justin Dodemaide
	04/04/24 - 04/18/24	Work on the client/server section of the code and debugging the code.	Abdoul Camara
	04/04/24- 04/18/24	Work on Gui(login interface), Account service class, deliverables, schedule, redid User+ Account and client+server class diagram.	Kebron Afework
	04/04/24- 04/18/24	User class, Account class, some Gui, and Junit testing	Mateo Tolentino
	04/04/24- 04/18/24	Client class, Authentication service class, and Junit testing.	Nico Savaides.

Table of Contents

1. PURPOSE.....	5
1.1. SCOPE.....	5
1.2 DEFINITIONS, ACRONYMS, ABBREVIATIONS.....	5
1.3. REFERENCES.....	5
1.4 OVERVIEW.....	45
2. OVERALL DESCRIPTION.....	5
2.1. PRODUCT PERSPECTIVE.....	5
2.2. PRODUCT ARCHITECTURE.....	5
2.3. PRODUCT FUNCTIONALITY/FEATURES.....	5
2.4. CONSTRAINTS.....	5
2.5. ASSUMPTIONS AND DEPENDENCIES.....	5
3. SPECIFIC REQUIREMENTS.....	7
3.1. FUNCTIONAL REQUIREMENTS.....	7
3.2. EXTERNAL INTERFACE REQUIREMENTS.....	7
3.3. INTERNAL INTERFACE REQUIREMENTS.....	7
4. NON-FUNCTIONAL REQUIREMENTS.....	9
4.1. SECURITY AND PRIVACY REQUIREMENTS.....	9
4.2. ENVIRONMENTAL REQUIREMENTS.....	9
4.3. Performance Requirements.....	9
5. USE	
CASES.....	10
6. USE CASES DIAGRAM.....	16
7. CLASS DIAGRAM.....	17
8. SPECIFIC CLASS	
DIAGRAMS.....	17
8.1 State Machine class	
Diagram.....	18
8.2 User +Account class Diagram	
.....	19
8.3 Client + Server class	
Diagram.....	20
10. SPECIFC SEQUENCE	
DIAGRAMS.....	21
10.1 Withdrawal	21
10.2 Deposit	22

23	10.3 Create an Account	
24	10.4 Transfer	
	10.5 User + Account	24

1. Purpose

This document outlines the requirements for the Banking System.

1.1. Scope

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

1.2. Definitions, Acronyms, Abbreviations

ATM -> Automated Teller Machine

GUI -> Graphical User Interface

RBAC -> Role-Based Access Control

1.3. References

Use Case Specification Document – Section 5

UML Use Case Diagrams Document – Section 6

Class Diagrams – Section 7

Sequence Diagrams – Section 8

1.4. Overview

The Banking Application is designed to provide a secure and convenient method to deposit, withdraw, and transfer money. Furthermore, allowing services such as creating, updating, and deleting customer accounts in a series of sequence steps.

2. Overall Description

2.1. Product Perspective

The Banking Application provides a secure and convenient method for a Client to deposit, withdraw, and transfer their money as well as cash and deposit checks. The Client can choose to be helped by an ATM or a Teller.

2.2. Product Architecture

The system will be organized into 5 major modules: the State Machine module, the Server module, the Server Client module, the User module, and the Account module.

2.3. Product Functionality/Features

- 2.3.1 Transaction Process Feature
- 2.3.2 Account Management Feature
- 2.3.3 User Management Feature
- 2.3.4 Server and Client Communication Features
- 2.3.5 External and Internal Features

2.4. Constraints

The User Interface must be intuitive and user-friendly so that Clients can use it without issue.

User and Account data must be stored in text files as comma-separated strings (field1,field2.field3,...) to allow the Server and Client modules to correctly interface with the data

Customers can only be helped by two roles: Teller and ATM

2.5. Assumptions and Dependencies

- 2.5.1 Only 1 user can be signed on at a time per machine
- 2.5.2 Tellers have access to different options than customers (making new accounts or removing accounts)
- 2.5.3 In normal conditions, customer-teller interactions would be face-to-face, and the teller's computer would be inaccessible to the customer.
- 2.5.4. Adding or removing teller IDs is outside the responsibility of this program

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Usernames and PINs are alphanumeric strings between 6 and 20 characters.

3.1.1.2 The client chooses between an ATM or teller transaction process, with the teller process having access to more options than the ATM process. The interfaces for both these processes are the same, but with the inaccessible options being invisible.

3.1.2. State Machine Requirements:

3.1.2.1 The driver is responsible for transitioning between prompt interfaces.

3.1.2.1.1 The first interface, Process Prompt, allows the user to choose “ATM” or “Teller.” The process is represented by a boolean “tellerProcess” in the State Machine class.

3.1.2.1.2 The second interface, Login Prompt, requires the user to enter their username and password.

3.1.2.1.3 The third interface, Main Controls, allows the user/teller to create an account, withdraw, deposit, check balance, transfer funds, add/remove accounts to their existing account, and delete their account as appropriate actions with a Teller (make and remove account options are disabled/invisible if ATM process). Each option has its logic that runs when the option is selected.

3.1.2.2 The driver provides a single point of access for the User reference and the Client reference.

3.1.2. Client Module Requirements:

3.1.2.1 Networking client communicates with the server by sending serialized string messages over the networking using TCP/IP protocol

3.1.2.2 The networking client is responsible for authenticating the user credentials. Attempting to log in involves communicating with the server to see if the file associated with the user exists (see 3.3).

3.1.2.3 Networking client can make new users or accounts by communicating with the server to make new files associated with credentials

3.1.2.4 Networking client updates user and account information by sending data to the server to be stored in respective files

3.1.2.5 Networking client removes user and account information by deleting their associated file

3.1.3. Server Module Requirements:

3.1.3.1 Server program is running even when 0 server clients are available

3.1.3.2 Server creates new threads when a client connects to poll for and process client messages

3.1.3.3 Server parses messages to store user, and account information in text files

3.1.4. User Module Requirements:

- 3.1.4.1 Users are instantiated by the ServerClient module upon User information being received from the Server
- 3.1.4.2 Users need a unique username and PIN, required to sign in
- 3.1.4.3 Users can have multiple accounts associated with them
- 3.1.4.4 There can only be one user signed in at a time per machine
- 3.1.4.5 User is responsible for account operations: logic for deposits, withdrawals, transfers, etc are handled within the User class
- 3.1.4.6 User communicates to Client for saving whenever an account operation is done
- 3.1.4.7 Users save just the IDs of their accounts, the actual data of accounts is stored separately

3.1.5. Account Module Requirements:

- 3.1.2.1 Accounts have unique IDs, generated when they are first created, that are used to retrieve their data files when being loaded
- 3.1.2.2 Accounts are either checking or savings
- 3.1.2.3 Accounts are not allowed to have below \$0 in their account
- 3.1.2.4 Accounts do not communicate with ServerClient for saving or loading, User is responsible for saving Account data
- 3.1.2.5 Accounts can be owned by multiple users
- 3.1.2.6 Accounts are created and stored in a text file for each user.

3.2. External Interface Requirements

- 3.2.1 Information on each of a User's accounts is only displayed after a successful login
- 3.2.2 The Main Controls interface, in addition to the buttons corresponding to user actions (deposit, withdraw), has a column displaying the User's account information
- 3.2.3 Each account and its information is stored in a panel within this column, with the panel horizontally displaying the Account fields (id, name, total, ...)

3.3. Internal Interface Requirements

- 3.3.1 User and Account data are stored in text files on the server machine
- 3.3.2 User file names follow the format: "USER" + username
- 3.3.3 Account file names follow the format: "ACCOUNT" + account ID
- 3.3.4 Data within the files is stored in the format: field/value/field1/value1, in which "field" is a property of either the User or Account class (username, pin, id, total, ...) and "value" is the corresponding value
- 3.3.5 It is the responsibility of the Client to parse the text into data that can be used for object instantiation
- 3.3.6 Teller IDs are strings stored in a single text file

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1 The system Should implement a role-based access control (RBAC)which restricts unwanted third parties from accessing any information(teller has their login to access their interface).

4.1.2 The Files containing customer's info should be restricted by an authorized party, in this instance it would be the server.

4.2. Environmental Requirements

4.2.1 The Software must be adaptable to changes over time, and the design must be built to be flexible to new features and updates.

4.2.2 The Software must be compatible with various operating systems like Linux, Mac, Windows, and more.

4.2.3 This system should also be compatible with various kinds of networks whether it be wireless, WAN, or LAN

4.3. Performance Requirements

4.3.1 The system should have a quick response time when it comes to user interactions. For example, the login interface should not take more than 10 seconds to verify the account information and allow the customer or the teller to log in.

4.3.2 The system should support multiple users or transactions to occur without an effect on the system's performance.

4.3.3 The systems should be up and working 95 percent of the time and 5 % of downtime for any needed maintenance

5. Use Cases

- 1. Login to the bank system or ATM**
- 2. Creating an account**
- 3. Updating an account**
- 4. Closing an account**
- 5. Deposits**
- 6. Withdrawals**
- 7. Transfers**

Use Case Specification:

1. Use Case ID: 001

Use Case Name: Login to the Bank system or ATM

Requirements:

Primary Actors: User (can be a Teller or a customer)

Pre-conditions:

The user has an active account with the banking system.

The user has valid login credentials (username and password).

Post-conditions:

Users are granted access based on authentication success.

Basic Flow:

Users provide a valid username and password.

The system verifies credentials against stored user profiles.

Upon successful authentication, the system grants access.

Extensions or Alternate Flows: Invalid credentials, authentication fails, the system denies access and logs attempts.

2. Use Case ID: 002

Use Case Name: Creating an account

Requirements:

Actors

Primary Actor: Teller

Supporting Actors: Banking System

Pre-conditions

The bank employee is authenticated and authorized to create accounts.

The customer has provided the necessary documentation and information for account creation.

Post-conditions

A new customer account is created and active.

The customer is informed of their new account details.

Basic Flow

Bank employee initiates the account creation process through the banking system's interface.

An employee enters customer information and selects the account type.

The banking system validates the information provided and checks for any existing customer records to avoid duplicates.

The system creates the account, assigning a unique account number.

The system generates a welcome package for the customer, including account details and information on how to use banking services.

The Bank employee informs the customer that the account has been successfully created and provides the welcome package.

Extensions or Alternate Flows: N/A

Exceptions: Invalid input data; unauthorized access attempt; system fails to update the database.

Use Case ID: 003

Use Case Name: Updating an account

Requirements:

Actors:

Primary Actor: Teller

Supporting Actors: Banking System

Pre-conditions

The bank employee is authenticated and authorized to update accounts.

The customer's account exists and is active.

Post-conditions

The customer's account information is updated in the banking system.

Basic Flow

Bank employee searches for the customer account using the banking system.

Employee selects the account to be updated and enters the new information or changes (e.g., change of address, updated contact details).

The banking system validates the updated information.

The system updates the account information as per the changes made.

The Bank employee confirms with the customer that their account information has been updated.

Extensions or Alternate Flows: N/A

Exceptions: Invalid input data; unauthorized access attempt; system fails to update the database.

Use Case ID: 004

Use Case Name: Closing an account

Requirements:

Actors

Primary Actor: Teller

Supporting Actors: Banking System

Pre-conditions

The bank employee is authenticated and authorized to close accounts.

The customer's account exists and is active.

All pending transactions or disputes related to the account have been resolved.

Post-conditions

The customer's account is closed and no longer accessible.

Any remaining balance has been transferred as per the customer's instructions.

Basic Flow

The customer requests account closure through a bank employee.

Bank employee searches for the customer's account and reviews it for any pending transactions or balances.

The employee initiates the account closure process if there are no impediments.

The banking system processes the closure, transferring any remaining funds to an account specified by the customer.

The system archives or deletes the account information according to the bank's policy on customer data.

The Bank employee informs the customer that their account has been successfully closed.

Extensions or Alternate Flows: N/A

Exceptions: Invalid input data; unauthorized access attempt; system fails to update the database.

Use Case ID: 005

Use Case Name: Deposits

Requirements:

Actors

Primary Actor: Customer

Supporting Actors: Teller (optional), Banking System

Pre-conditions

The customer is authenticated.

The account into which the deposit will be made exists and is active.

Post-conditions

The account balance is increased by the deposit amount.

A transaction record is created

Basic Flow

The customer initiates a deposit transaction via an ATM or bank employee.

The customer or bank employee enters the amount to be deposited.

The banking system verifies the account details and processes the deposit.

The system updates the account balance.

The system generates a transaction receipt.

The customer receives confirmation of the successful deposit.

Extensions or Alternate Flows:

Insufficient Funds: The system displays an error message, and the transaction is not processed.

System Timeout: For ATM, the session times out if there's no activity for a certain period. For employees, a warning is displayed before automatic logout.

Exceptions:

Transaction failure due to technical issues; the system displays an error message and logs the incident.

Unauthorized transaction attempt; the system blocks the transaction and alerts the system administrator.

Use Case ID: 006**Use Case Name:** Withdrawals**Requirements:****Actors**

Primary Actor: Customer

Supporting Actors: ATM, Banking System

Pre-conditions

The customer is authenticated.

The account from which the withdrawal will be made exists, is active, and has sufficient funds.

Post-conditions

The account balance is decreased by the withdrawal amount.

A transaction record is created.

Basic Flow

The customer initiates a withdrawal transaction at an ATM.

The customer enters the amount to be withdrawn.

The ATM verifies the transaction with the banking system.

The banking system checks the account for sufficient funds.

If funds are sufficient, the system processes the withdrawal, updates the account balance, and instructs the ATM to dispense the cash.

The ATM dispenses the cash.

The system generates a transaction receipt.

The customer receives the cash and transaction receipt.

Extensions or Alternate Flows:

Insufficient Funds: The system displays an error message, and the transaction is not processed.

System Timeout: For ATM, the session times out if there's no activity for a certain period. For employees, a warning is displayed before automatic logout.

Exceptions:

Transaction failure due to technical issues; the system displays an error message and logs the incident.

Unauthorized transaction attempt; the system blocks the transaction and alerts the system administrator.

Use Case ID: 007

Use Case Name: Transfers

Requirements:

Actors

Primary Actor: Customer

Supporting Actors: Banking System

Pre-conditions

Both the sending and receiving accounts exist and are active.

The sending account has sufficient funds for the transfer.

Post-conditions

The sending account balance is decreased by the transfer amount.

The receiving account balance is increased by the transfer amount.

A transaction record is created for both accounts.

Basic Flow

The customer initiates a transfer transaction via online banking or a bank employee.

The customer enters the recipient's account details and the amount to be transferred.

The banking system verifies both accounts and checks the sending account for sufficient funds.

The system processes the transfer, debiting the sending account and crediting the receiving account.

The system updates both account balances.

The system generates a transaction receipt.

The customer receives confirmation of the successful transfer.

Extensions or Alternate Flows:

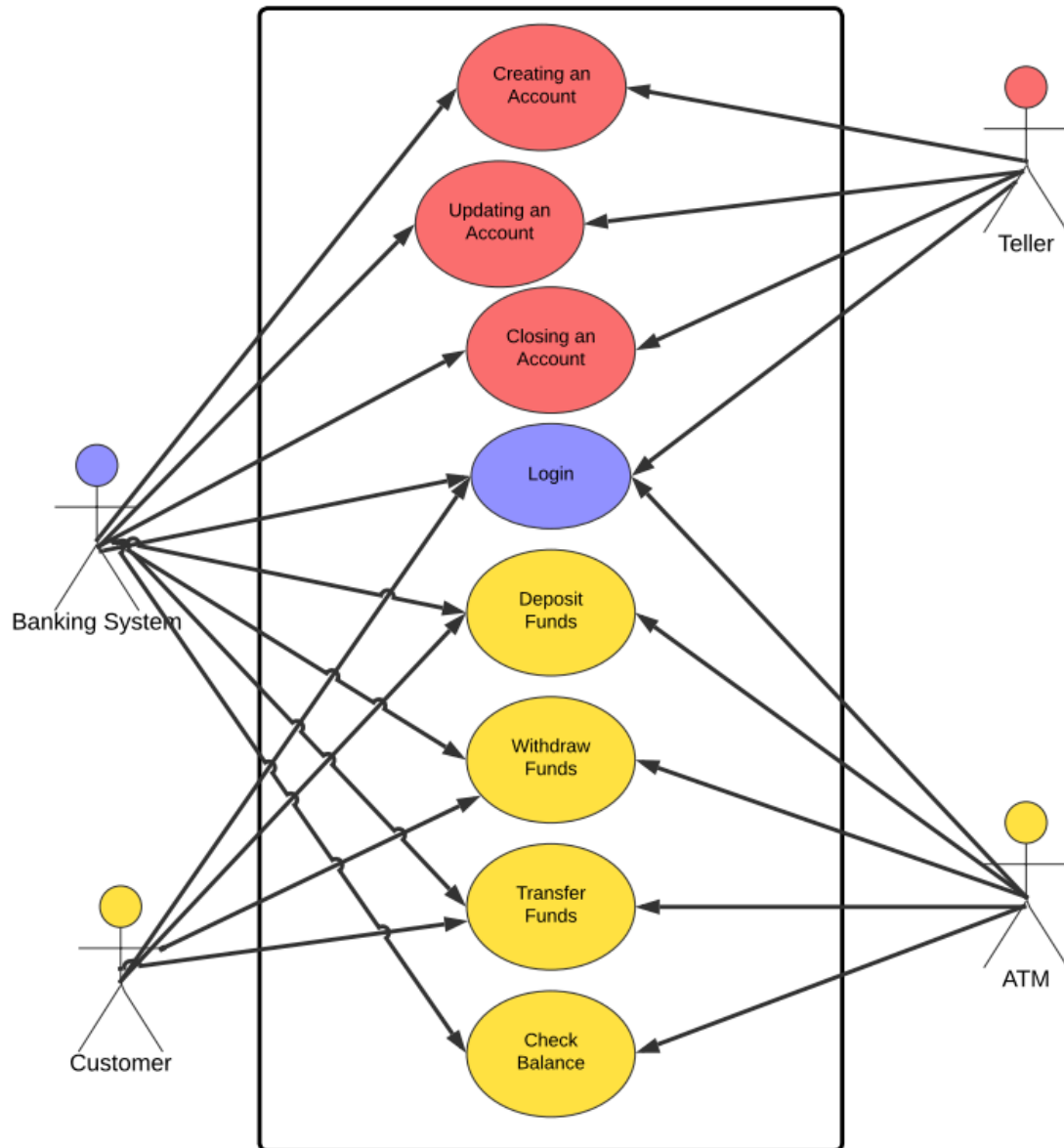
Insufficient Funds: The system displays an error message, and the transaction is not processed.

Exceptions:

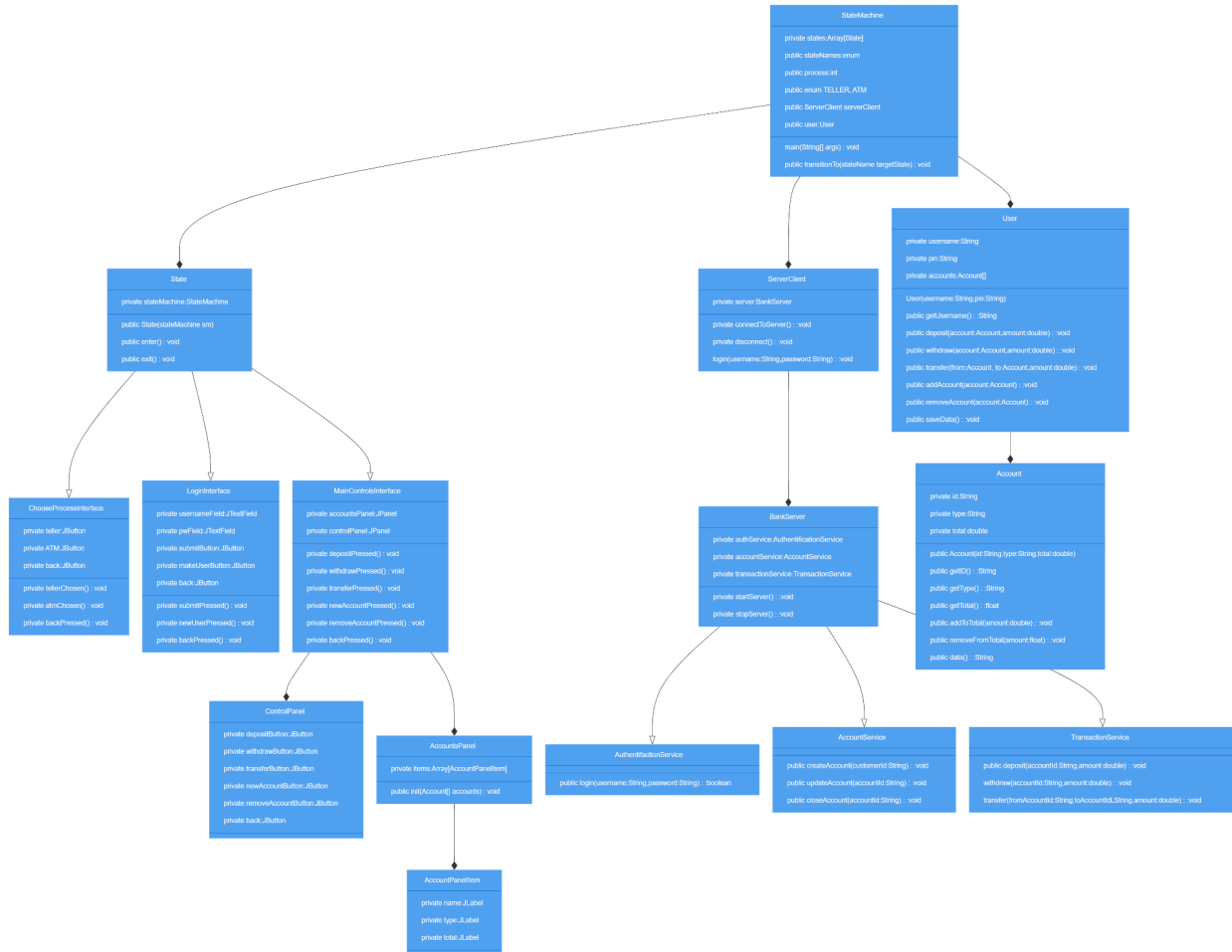
Transaction failure due to technical issues; the system displays an error message and logs the incident.

Unauthorized transaction attempt; the system blocks the transaction and alerts the system administrator.

6. Use Case Diagram

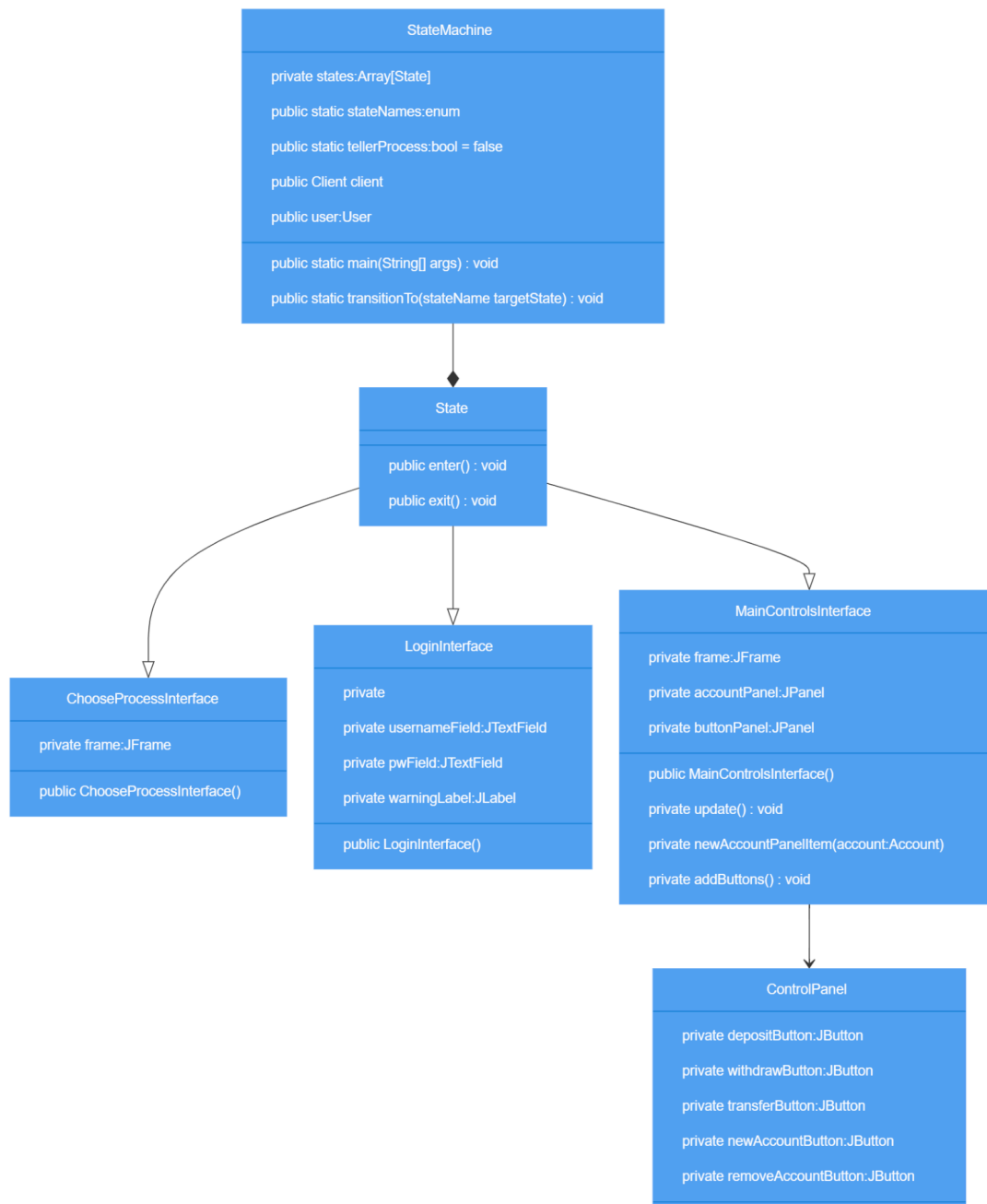


7. Class Diagram

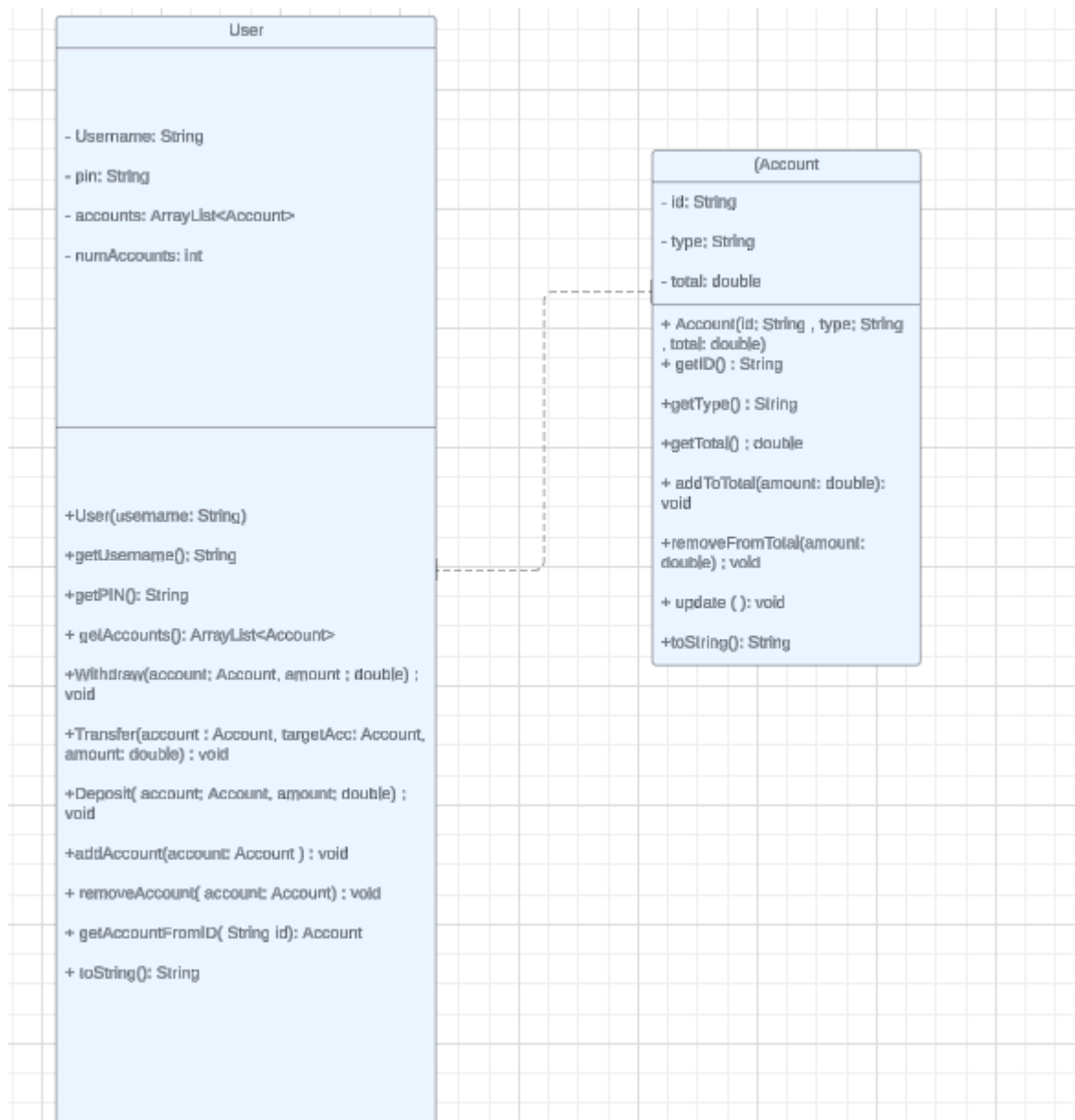


8. Specific Class Diagrams

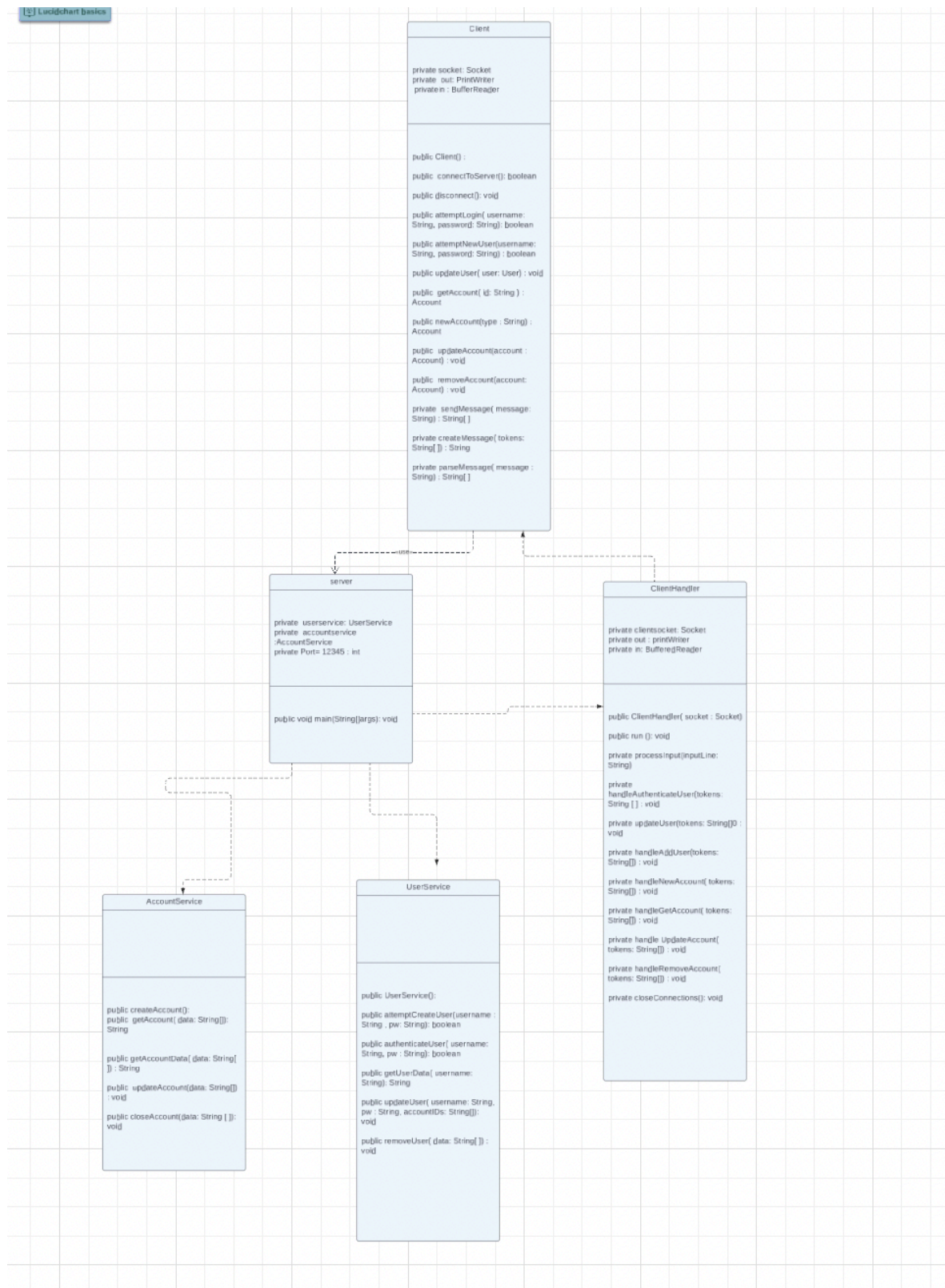
8.1 State Machine Class Diagram



8.2 User + Account Class Diagram

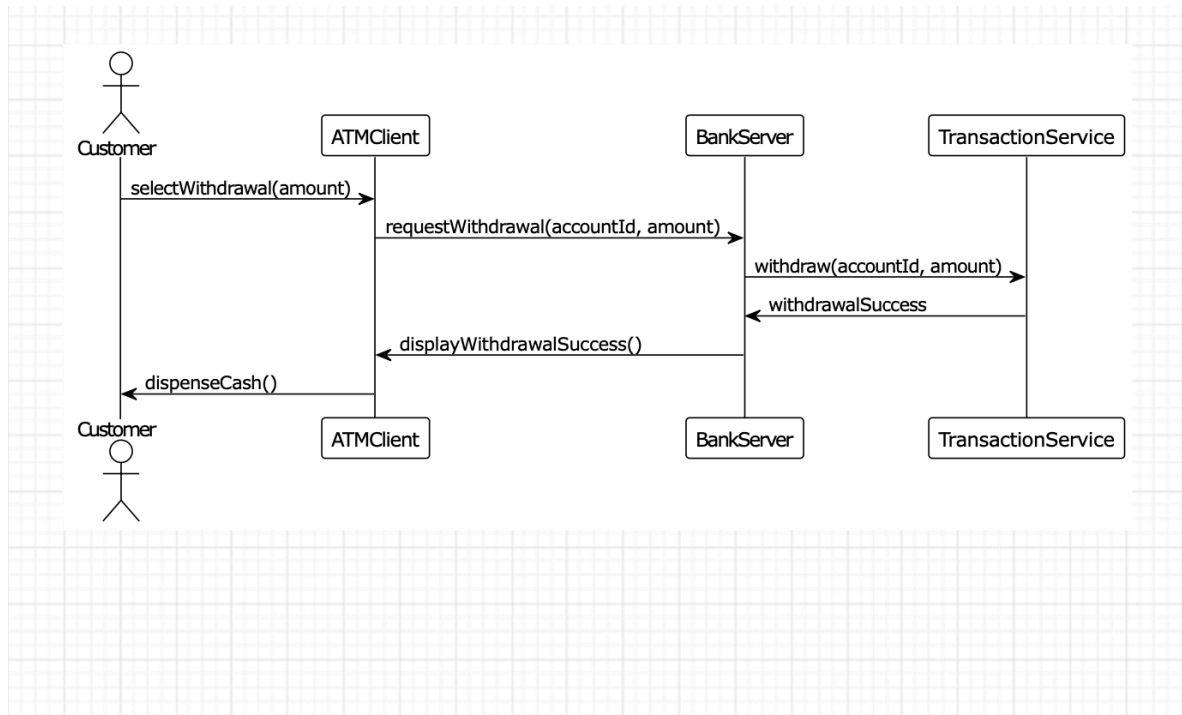


8.3 Client + Server Class Diagram

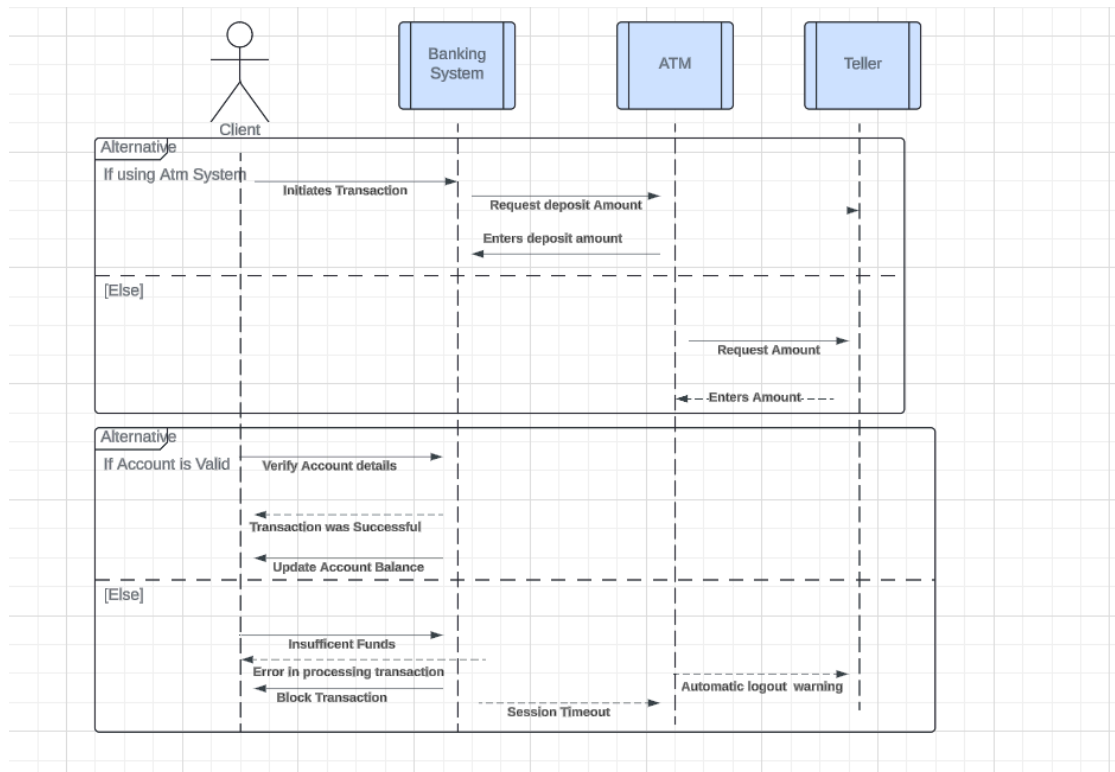


10. Specific Sequence Diagram

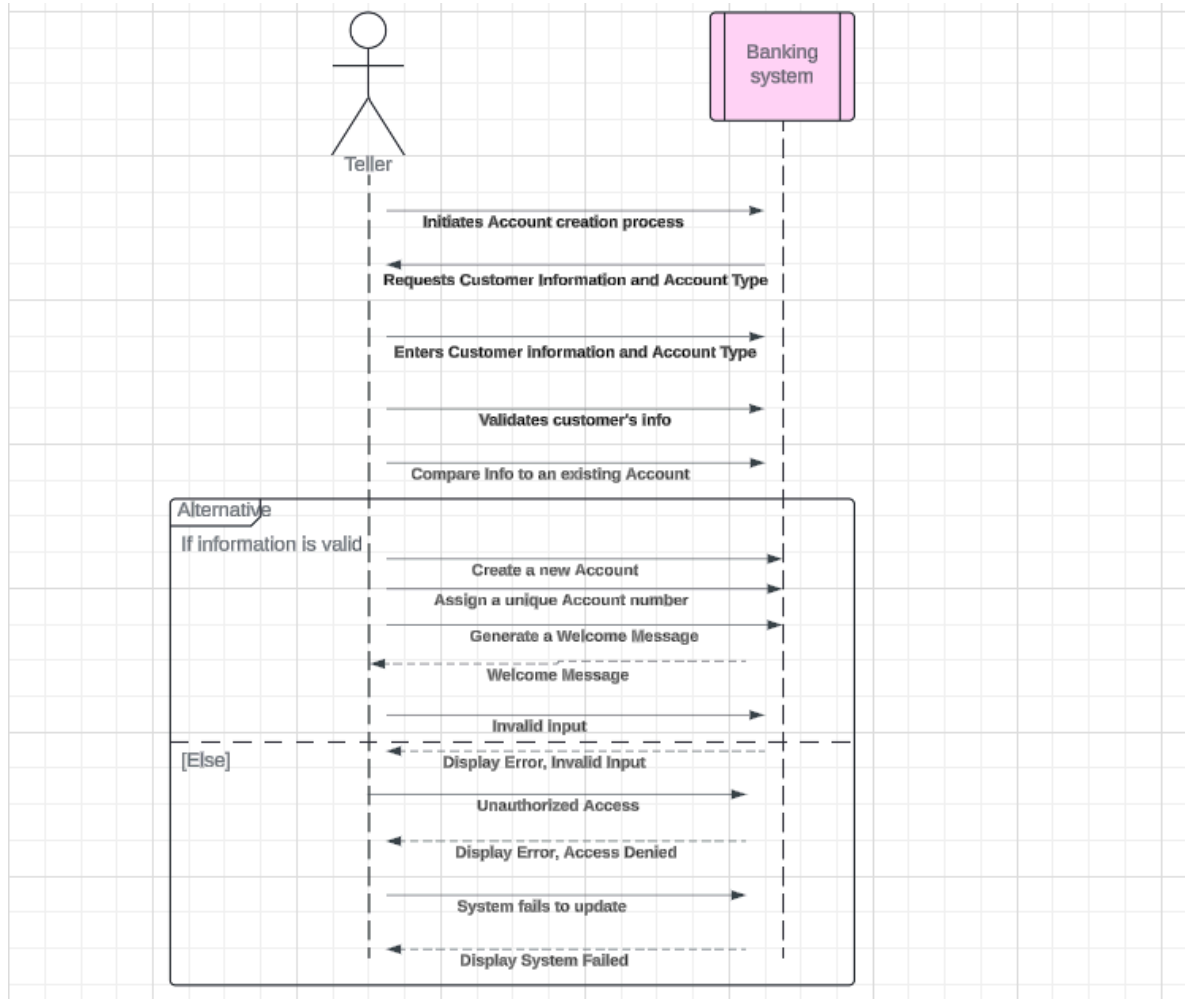
10.1 Withdrawal



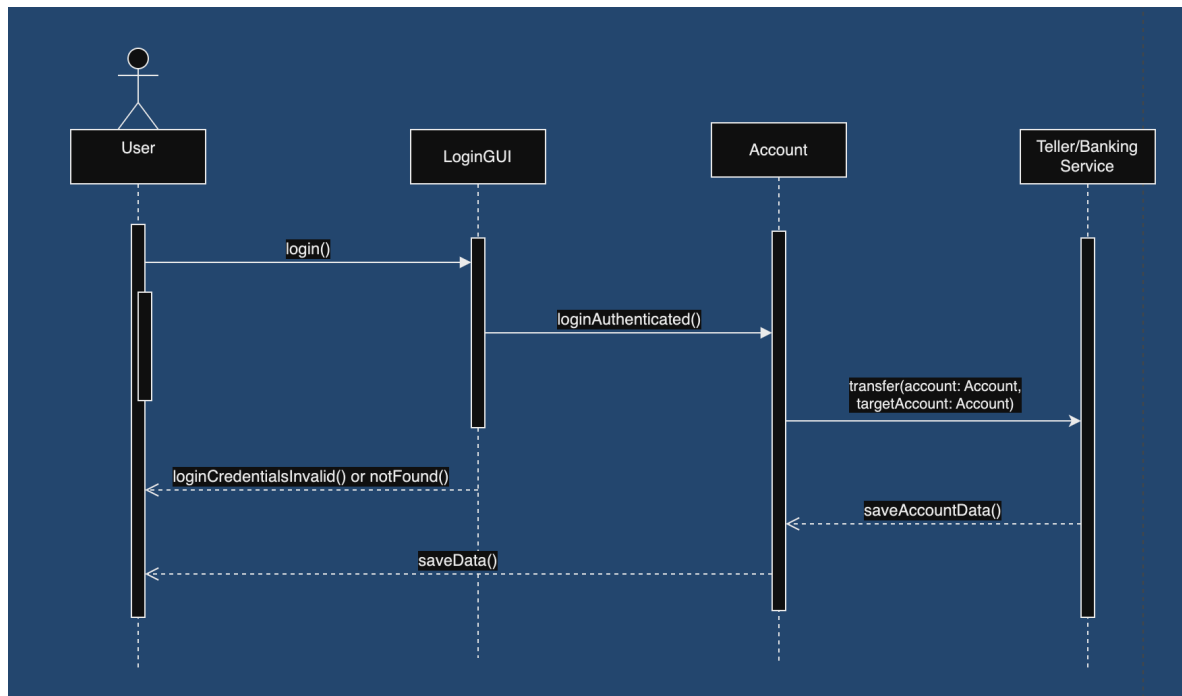
10.2 Deposit



10.3 Create An Account



10.4 Transfer



10.5 User + Account

