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

Table of Contents

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. Checks need to be deposited or cashed and there is much more risk in keeping all funds as cash.[i]

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 Driver 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.

Account data must be stored in a data file where each field is separated such that it looks like: 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

Clients 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. Driver 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 stored in the Driver class.

3.1.2.1.2 The second interface, Login Prompt, requires the user to enter their username and password. If the teller processes, the teller must enter their ID to be verified.

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 own 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 ServerClient reference.

3.1.2. Server Client Module Requirements:

3.1.2.1 Users should be allowed to log in to their respective User accounts with their chosen username and password, both of which are alphanumeric strings between 6 and 20 characters in length.

3.1.2.2 The server 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 Server client can make new user accounts by communicating with the server to make new files associated with credentials

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

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 stores user, account, and teller information in text files

3.1.3.3 Server communicates with Server Clients to send and receive data to be stored in text files

3.1.4. User Module Requirements:

3.1.4.1 Users are instantiated by ServerClient module upon User information being received from 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 User class

3.1.4.6 User communicates to ServerClient 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

3.2. External Interface Requirements

3.2.1 Information of each of a User’s accounts is only displayed after 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 accounts 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 Its 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 party from accessing any information(teller has their login to access their interface).

4.1.2 The Files containing customers 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, 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 Thes 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 accounts information and allow the customer or the teller to login.

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 is in possession of 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 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.
Employee enters customer information and selects the account type.
Banking system validates the information provided and checks for any existing customer records to avoid duplicates.
System creates the account, assigning a unique account number.
System generates a welcome package for the customer, including account details and information on how to use banking services.
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).
Banking system validates the updated information.
System updates the account information as per the changes made.
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
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.
Employee initiates the account closure process if there are no impediments.
Banking system processes the closure, transferring any remaining funds to an account specified by the customer.
System archives or deletes the account information according to the bank's policy on customer data.
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.

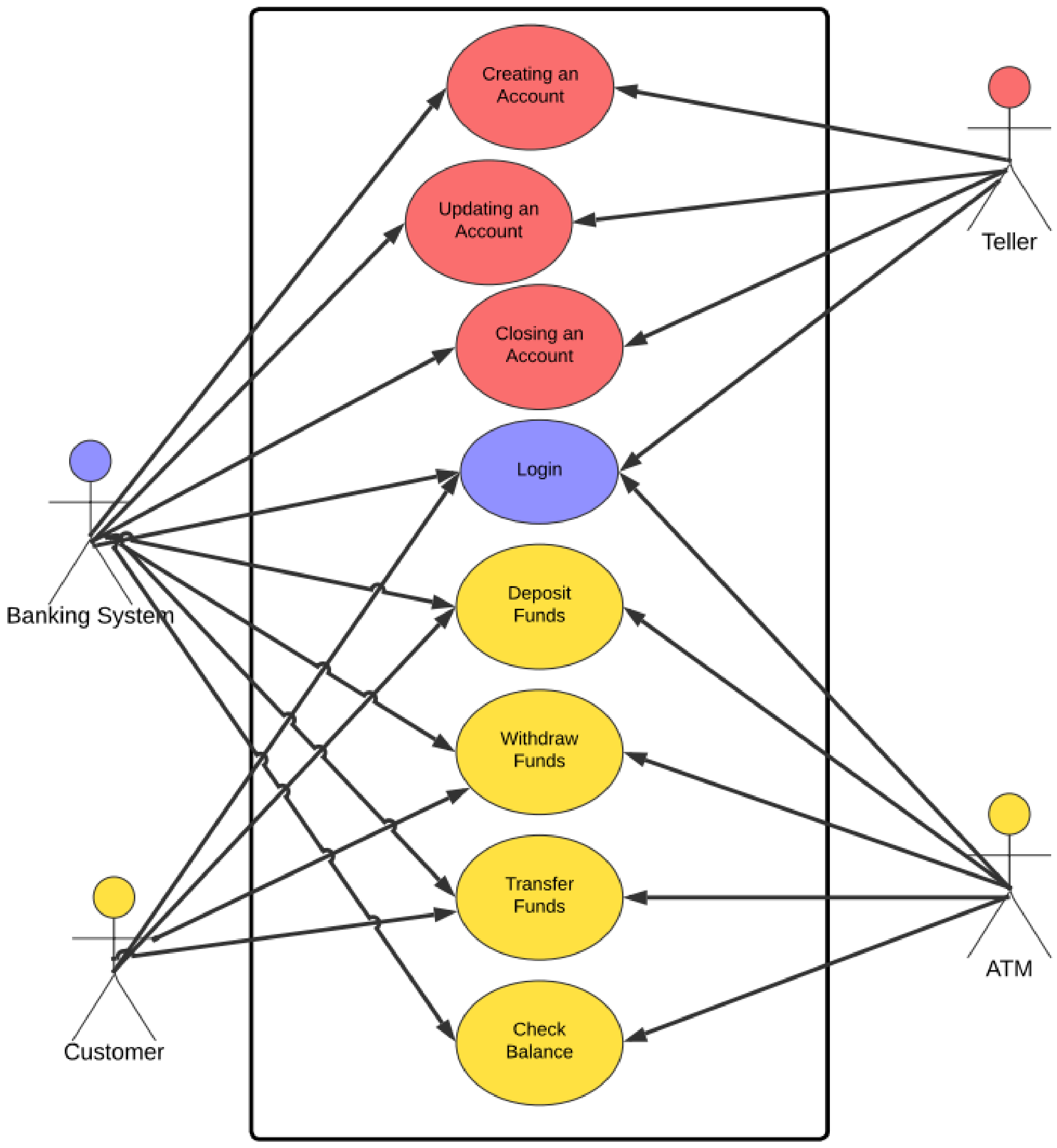
Use Case ID: 006
Use Case Name: Withdrawals
Requirements:
Actors
Primary Actor: Customer
Supporting Actors: ATM, Banking System
Pre-conditions
The customer is authenticated.
The account from which the withdrawal will be made exists, is active, and has sufficient funds.
Post-conditions
The account balance 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.

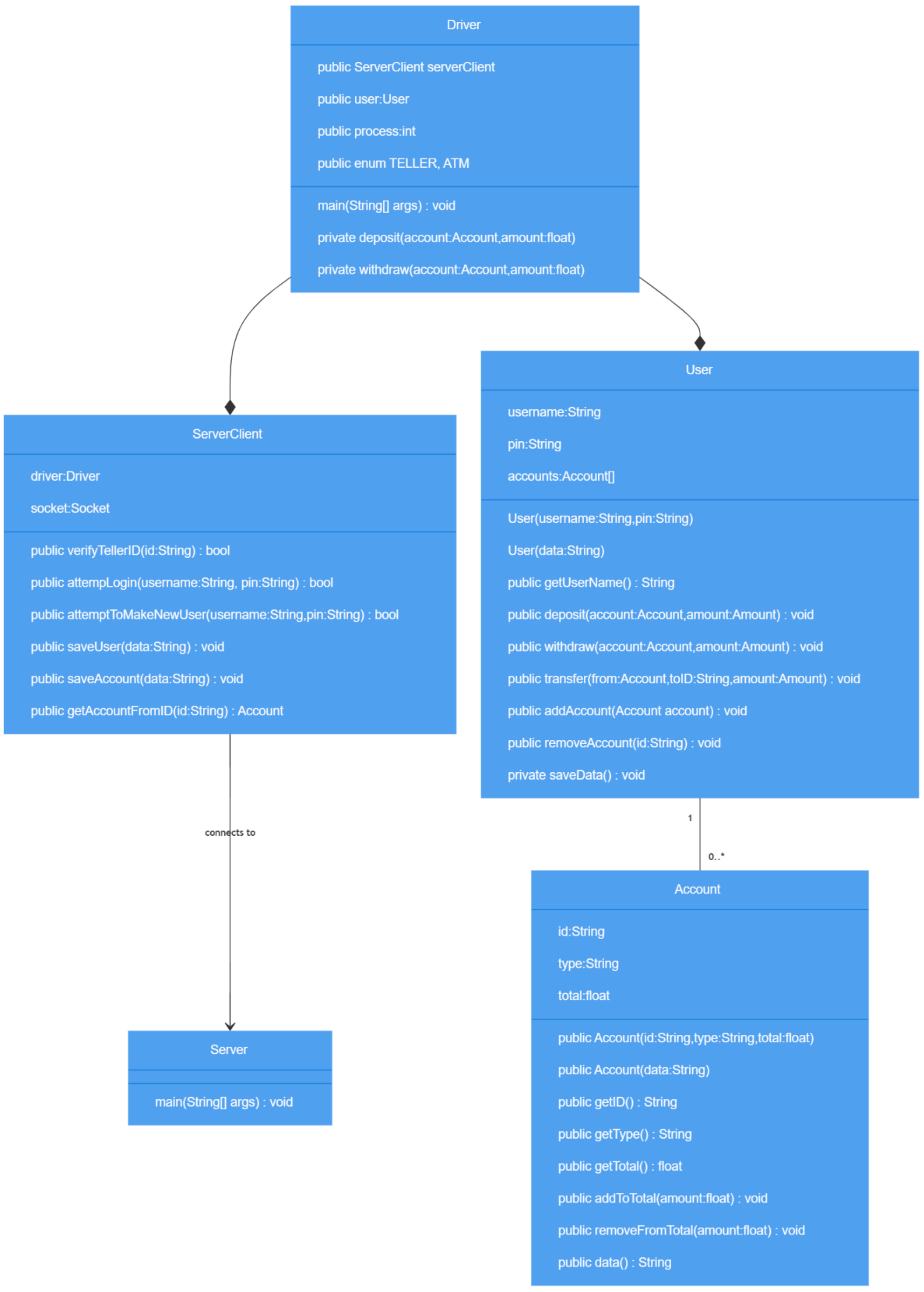
Use Case ID: 007
Use Case Name: Transfers
Requirements:
Actors
Primary Actor: Customer
Supporting Actors: Banking System
Pre-conditions
Both the sending and receiving accounts exist and are active.
The sending account has sufficient funds for the transfer.
Post-conditions
The sending account balance 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.

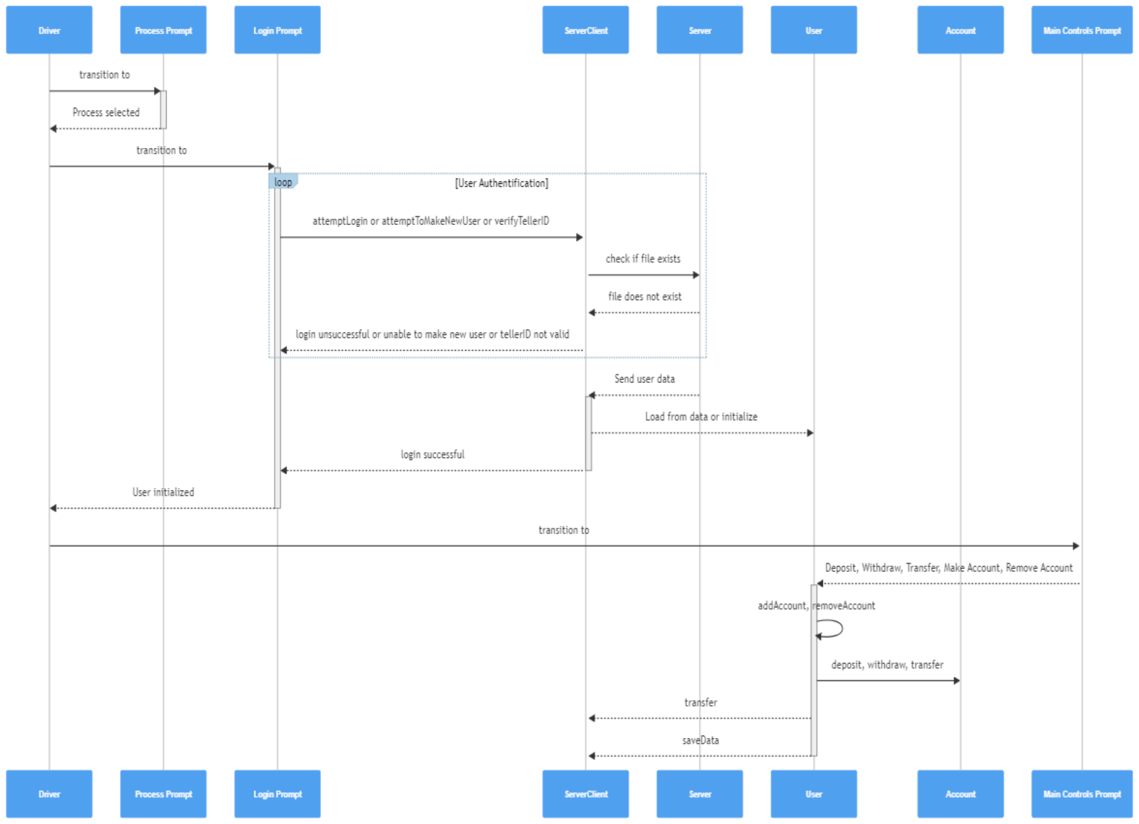
6. Use Case Diagrams



7. Class Diagram


Driver
public ServerClient serverClient
public user:User
public process:int
public enum TELLER, ATM
main(String[] args) : void
private deposit(account:Account,amount:float)
private withdraw(account:Account,amount:float)
ServerClient
driver:Driver
socket:Socket
public verifyTellerID(id:String) : bool
public attempLogin(username:String, pin:String) : bool
public attemptToMakeNewUser(username:String,pin:String) : bool
public saveUser(data:String) : void
public saveAccount(data:String) : void
public getAccountFromID(id:String) : Account
User
username:String
pin:String
accounts:Account[]
User(username:String,pin:String)
User(data:String)
public getUserName() : String
public deposit(account:Account,amount:Amount) : void
public withdraw(account:Account,amount:Amount) : void
public transfer(from:Account,toID:String,amount:Amount) : void
public addAccount(Account account) : void
public removeAccount(id:String) : void
private saveData() : void
connects to
1
0..*
Server
main(String[] args) : void
Account
id:String
type:String
total:float
public Account(id:String,type:String,total:float)
public Account(data:String)
public getID() : String
public getType() : String
public getTotal() : float
public addToTotal(amount:float) : void
public removeFromTotal(amount:float) : void
public data() : String

8. Sequence Diagram


Driver
Process Prompt
Login Prompt
ServerClient
Server
User
Account
Main Controls Prompt
transition to
Process selected
transition to
loop
[User Authentification]
attemptLogin or attemptToMakeNewUser or verifyTellerID
check if file exists
file does not exist
login unsuccessful or unable to make new user or tellerID not valid
Send user data
Load from data or initialize
login successful
User initialized
transition to
Deposit, Withdraw, Transfer, Make Account, Remove Account
addAccount, removeAccount
deposit, withdraw, transfer
transfer
saveData
Driver
Process Prompt
Login Prompt
ServerClient
Server
User
Account
Main Controls Prompt