CS-401 Group 2 Project
Blackjack

Software Requirements Specification

Authors: Haoze (Albert) Chen, Thomas Low, Jievy Lance Jumawan, Flora Luo, Parsa Majboleh

Revision History

Date	Revision	Description	Author
2/12/2023	1.0	Initial Version	Chen
2/17/2023	1.1	Initiated descriptions for sections 1, 2, 3	Flora, Chen, Parsa, Jievy, Low
2/17/2023	1.2	4.1 & 4.2 & 4.3 started	Parsa
2/18/2023	1.3	3.3. & 3.2 started	Low
2/18/2023	1.4	Initiated section 3.1.1	Chen
2/19/2023	1.5	Added Environmental Reqs	Jievy
2/19/2023	1.6	Initiated description to section 3.2.1 and 3.1.2	Flora
2/19/2023	1.7	4.1 & 4.2 & 4.3 finished	Parsa
2/24/2023	1.8	Completed the SRS except for the reference part	Flora, Chen, Jievy, Low
3/1/2023	1.9	Completed UML Use Case Diagrams	Flora
3/1/2023	2.0	Completed UML Sequence Diagrams	Low
3/1/2023	2.1	Completed UML Class Diagrams	Chen, Low, Jievy, Flora

4/5/2023	2.2	Updated UML Use Case Diagram and removed “Lobby” related specifications from the document	Albert (Haoze Chen)

Table of Contents

1. Purpose
    - 1.1. Scope 4
    - 1.2. Definitions, Acronyms, Abbreviations 4
    - 1.3. References 5
    - 1.4. Overview 5
2. Overall Description
    - 2.1. Product Perspective 5
    - 2.2. Product Architecture 5
    - 2.3. Product Functionality/Features 5
    - 2.4. Constraints 6
    - 2.5. Assumptions and Dependencies 6
3. Specific Requirements
    - 3.1. Functional Requirements 6
    - 3.2. External Interface Requirements 8
    - 3.3. Internal Interface Requirements 8
4. Non-Functional Requirements
    - 4.1. Security and Privacy Requirements 9
    - 4.2. Environmental Requirements 9
    - 4.3. Performance Requirements 9

1. Purpose

This document outlines the requirements and design for a Black Jack Game.

1.1. Scope

This document will catalog the user and system requirements for the Black Jack Game. Implementation of the requirements will be described in referenced documents.

1.2. Definitions, Acronyms, Abbreviations

Aces: Count as the value one or eleven.

Blackjack: A gambling card game in which players try to acquire cards with a total value as close to 21 as possible.

Bust: A hand with a total value higher than 21.

Double down: Double your bet in the middle of a hand (equal to your ante) in return for only one extra card.

Player: A person who plays the game.

Dealer: An automated program that the players play against in the game.

Face Cards: Cards with faces count as 10 each.

Hand: A set of cards to a player or a dealer.

Hit: An instruction is given by players to the dealer to request an extra card.

Insurance Bet: A side bet offered to the player if the dealer's up-card is an ace.

Split: When there are two cards of the same denomination, they can be split into two hands, with the original bet to the original card and an equal amount is placed on the other card.

Stand: To hold your total and end your turn.

Surrender: When you elect to throw away your cards rather than play on or stand.

1.3. References

1. Use Case Specification Document (Page 10)
2. UML Use Case Diagrams Document (Page 14)
3. UML Class Diagrams Document (Page 15)
4. UML Sequence Diagrams Document (Page 16)

1.4. Overview

Designed for people to play Black Jack Games online. The game maintains accounts with individual identifiers and balance, for players to find entertainment anywhere at their convenience in a virtual environment.

2. Overall Description

2.1. Product Perspective

The game is independent and self-contained.

2.2. Product Architecture

The system will be organized into 3 major modules: the common, client, and server modules.

2.3. Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

- User account creation
- User balance, winnings, and losses
- Withdrawal and/or addition to balance
- Display user and dealer hand

- In-game actions - bet, double, fold, stay, leave game
- In-game data - timer, rounds, funds, card total
- Player per table limit
- Dealer - Deal cards, shuffle the deck, collect or give funds

2.4. Constraints

2.4.1. Compatible with all supported OS

2.4.2. The user needs access to the internet

2.4.3. The user needs some form of input (mouse/ keys)

2.4.4. The maximum number of users the game system can handle at once is dependent on the server. For common consumer-based computers hosting the server, the maximum number of users is limited to 15.

2.5. Assumptions and Dependencies

2.5.1 It is assumed that the maximum number of players at a given time is 15.

2.5.2 It is assumed that players have a stable internet connection.

2.5.3 It is assumed that the servers will not crash.

2.5.4 It is assumed that the power source is stable.

2.5.5 It is assumed that the hardware is scalable.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements

3.1.1.1. The system should be implemented with Java using the client-server architecture pattern over the TCP/IP protocol. The networking should be multithreaded and bi-directional.

3.1.1.2. The system should have a Java client application to display the user interface for all the features mentioned in 2.3 and communicate with the server.

3.1.1.3. The system should have a Java server application to serve the client and handle the data.

3.1.1.4. Users should be allowed to log in using their username and password. System checks with the server that if the account is currently in access, the user cannot log in to the system through the current application.

3.1.2. Client Module Requirements

3.1.2.1 The system should process funds between a user's personal financial account and the system's user's account. The user can add or transfer any amount in whole dollars into the system's account and withdraw up to $10,000 or the account' balance, whichever is less, out of the system. Send request.

3.1.2.2 The action to join a game is inaccessible to the user if the user's balance in the system is zero.

3.1.2.3 User Login Interface: Refer to 3.1.1.4. It contains the function to create a new account for new users.

3.1.2.4 New User Interface: A user is prompted to enter their legal first and last names, a unique username, and a password. Username and password accepts alphanumeric between 6 to 20 characters. Username must be unique to the user. User information is checked and saved by the server if the username is unique. If the username is taken, the interface will notify the user to try a new username.

3.1.2.5 User Home Page Visual: Display the user's name, username, current balance in the system, actions accepted by the systems in reference to 3.1.2.6. The system will display the balance, tracked by the server, as the user lands on the Home Page. The User may come from the login interface or from the termination of a gameplay.

3.1.2.6 User Home Page Action Interface: On this page, the user can join a game, add and withdraw funding to and from the system, log out of the game. Actions taken will be communicated to the server to implement.

~~3.1.2.7 User Lobby Page: Display a waiting indicator while the player waits to be added to a game. The system pulls information from the server and displays the number of users who are concurrently waiting to join an available seat at a Black Jack Game table and the player's current position.~~

3.1.2.8 User In-Game Action Interface: While in the game, the player can bet, hit, stand, double down, split, insurance bet, surrender, and exit the game. Actions taken will be communicated to the server to implement.

3.1.2.9 User In Game Visual: to facilitate a game in motion, the following is displays to user's application: all player's username, dealer's title, each players and dealer's face up cards, current balance of the user excluding the amount in bet, current bet for the game in play, all action functions in reference to 3.1.2.9.

3.1.3. Server Module Requirements:

3.1.3.1. The server application should be initiated before the client application.

3.1.3.2. Game initialization: The server initialization should find and open the correct data file in the system and load the history data stored in the system file. The server should initialize the Java class objects, including "house" and "player."

3.1.3.2. User authentication: The server should check the username and password input sent from the client. If the username and/or password don't match the existing data, the server will send an error message to the client. Otherwise, the server should load the logged-in user's data from the system file, and send the user data and an initialization signal to the client to initialize the in-game GUI.

3.1.3.3. User account management: The server should update the user's funds when receiving requests for fund charging or fund withdrawing from the client.

3.1.3.4. Game management: When the client requests to start a game the server should load players' data and initialize the in-game data and then send the initialized data back to the client. Player data include each player's username, account balance, and an assigned number based on the alphanumerical order of their username for the order of each round. The server should also generate random card decks and deal two cards to each player and the dealer.

3.1.3.5. Deck management: The server should manage the deck of cards, including shuffling the deck and dealing cards to players.

3.1.3.6. Table management: The server should be able to manage the number of players per table and handle the distribution of cards to each player.

3.1.3.7. Dealer management: The server should be responsible for managing the dealer, including collecting or giving funds and dealing cards to players.

3.1.3.8. Error handling: The server should handle any errors that occur during gameplay, such as when a player tries to place a bet they cannot afford or when a player tries to join a full table.

3.2. External Interface Requirements

3.2.1 The user can send input via keyboard or mouse click. The GUI is native to the arrow, while key binding will read to the council.

3.3. Internal Interface Requirements

3.3.1 (Login/ home screen) The user will be greeted and asked to enter their login data. Once a successful login is made, the client will request a connection with the server. Now with a connection, the user will be on a home screen. Here users can add/withdraw funds or enter a game. If the user lacks funds and attempts to join, they will be met with an error screen and sent back to the home screen.

3.3.2 (Gameplay - needs editing bc I haven't looked at all the rules) Once loaded into a game, the player will interact with 3 GUIs, betting, action, and continue. Betting will have 4 options from 50, 100, 500, and 1000 per round. Action will have the classic hit or stand. Continue, if you wish to leave after the round.

3.3.3 (Add funds) Adding funds will open a transaction window. This window will allow the user to pick from a selected amount. They will be asked for a card number? Then another confirmation window will pop up with a yes or no option. If the card number fails or the confirmation window is met with a no, the transaction will be canceled. Withdrawing is another story.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1 The program must not transmit any personally identifiable information of the players. No cookies and no third-party services that collect information.

4.1.2 The program must not allow multiple logins at once.

4.1.3 The game shall use secure connections for all data transmissions. More specifically, the data being transmitted between the user’s device and the game’s server should be using TCP/IP protocol.

4.2. Environmental Requirements

4.2.1 The game will require a computer with sufficient processing power, memory, and storage to run the software.

4.2.2 The game will require a reliable and stable network connection to support real-time gameplay.

4.2.3 The game will require reliable and stable power to support real-time gameplay.

4.3. Performance Requirements

4.3.1 The game shall be able to handle an appropriate amount of concurrent users based on the system and the hardware.

4.3.2 The game shall have a response time of less than 1 second for user actions. To be more specific 1.0 seconds is about the limit for the user's flow of thought to stay uninterrupted.

Reference 1: Use Case Specification

Use Case ID: 1

- Use Case Name: Sign Up
- Relevant Requirements: Log In
- Primary Actor: Player
- Pre-conditions: The player is not signed up and is on the account page
- Post-conditions: The player is signed up
- Basic Flow or Main Scenario:
    1. The player enters the game
    2. The system displays the account page
    3. The player clicks the "Sign Up" button on the account page
    4. The player enters the username and password on the account page
    5. The player clicks the "Submit" button on the account page

Use Case ID: 2

- Use Case Name: Log In
- Relevant Requirements: Login, Player
- Primary Actor: Player
- Pre-conditions: The player is not logged in and is on the account page
- Post-conditions: The player is logged in and is on the home page

- Basic Flow or Main Scenario:
    1. The player enters their username and password
    2. The system verifies the player's username and password.
    3. If the username and password are correct, the system directs the player to the home page; if the username and passwords are incorrect, the system displays an error message.

Use Case ID: 3

- Use Case Name: Add Funds
- Relevant Requirements: Home Screen, Player
- Primary Actor: Player
- Pre-conditions: The player is logged in and is on the home page
- Post-conditions: The player's account balance is increased
- Basic Flow or Main Scenario:
    1. The player clicks on the "Add Funds" button on the home page
    2. The player enters a number
    3. The system adds the number to the player's account

Use Case ID: 4

- Use Case Name: Check Funds
- Relevant Requirements: Home Screen, Player
- Primary Actor: Player
- Pre-conditions: The player is logged in
- Post-conditions: The player's balance is displayed on the screen
- Basic Flow or Main Scenario:
    1. The player clicks on the "Check Funds" button
    2. The system displays the player's balance

Use Case ID: 5

- Use Case Name: Cash Out
- Relevant Requirements: Home Screen, Player
- Primary Actor: Player
- Pre-conditions: The player is logged in and their account's balance is greater than zero
- Post-conditions: The player's balance in their system account is decreased
- Basic Flow or Main Scenario:
    1. The player clicks on the "Cash Out" button
    2. The player enters the number of funds to cash out
    3. The system deducts the number the player input from their account

Use Case ID: 6

- Use Case Name: Log Out
- Relevant Requirements: Home Screen, Player

- Primary Actor: Player
- Pre-conditions: The player is logged in
- Post-conditions: The player is logged out of the system and is exited from the game
- Basic Flow or Main Scenario:
    1. The player clicks on the "Log Out" button
    2. The system removes the player from the game and sets the account status to “offline”

Use Case ID: 7

- Use Case Name: Make Bet
- Relevant Requirements: Game Play Screen, Player
- Primary Actor: Player
- Pre-conditions: The player has enough balance to make the bet
- Post-conditions: A bet is made and the bet is deducted from the player’s balance
- Basic Flow or Main Scenario:
    1. The player clicks on the “Make Bet” button
    2. The system verifies if the player has sufficient funds to make the bet
    3. If the player has sufficient funds to make the bet, the system places bet on the player’s current hand. Otherwise, the system displays an error message

Use Case ID: 7

- Use Case Name: Double Down
- Relevant Requirements: Game Play Screen, Player
- Primary Actor: Player
- Pre-conditions: The player has enough balance to double down
- Post-conditions: A doubled bet is made and the bet is deducted from the player’s balance
- Basic Flow or Main Scenario:
    1. The player clicks on the “Double Down” button
    2. The system verifies if the player has sufficient funds to double down
    3. If the player has sufficient funds to double down, the system places a doubled bet on the player’s current hand. Otherwise, the system displays an error message

Use Case ID: 8

- Use Case Name: Hit
- Relevant Requirements: Game Play Screen, Player
- Primary Actor: Player
- Pre-conditions: The number of cards left in the deck is greater than zero
- Post-conditions: The cards and hand values of the player are updated
- Basic Flow or Main Scenario:
    1. The player clicks on the “Hit” button

2. The system checks if the player's hand's value is greater than 21
3. If the player's hand's value is greater than 21, the player busts. Otherwise, the dealer deals a card for the

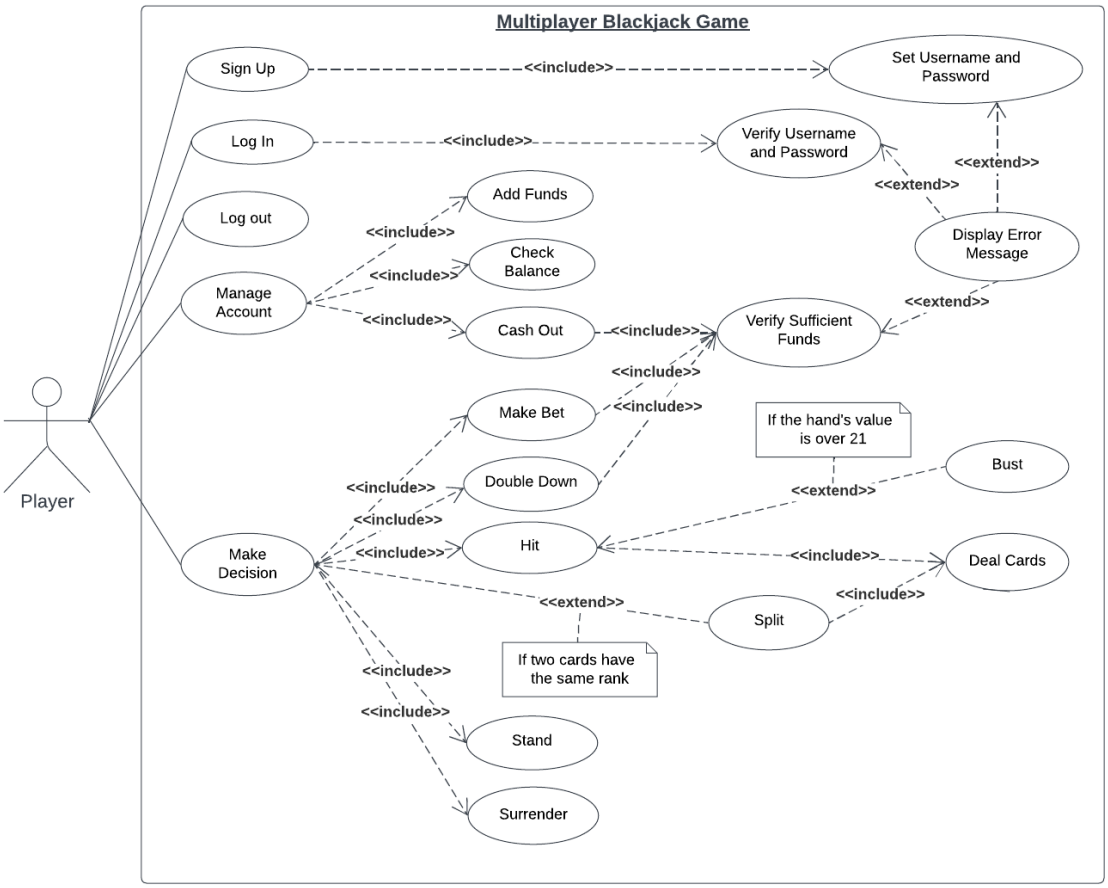
Use Case ID: 8

- Use Case Name: Split
- Relevant Requirements: Game Play Screen, Player
- Primary Actor: Player
- Pre-conditions: The player has two cards of the same rank
- Post-conditions: The player's hand is split into two, a new card is dealt to the new hand, and a bet of the same value as the original bet is placed on the new hand
- Basic Flow or Main Scenario:
    1. The player clicks on the "Split" button
    2. The system splits the player's hand into two–left hand and right hand and places the original bet on both hands
    3. The player plays two hands separately until one of the two hands busts

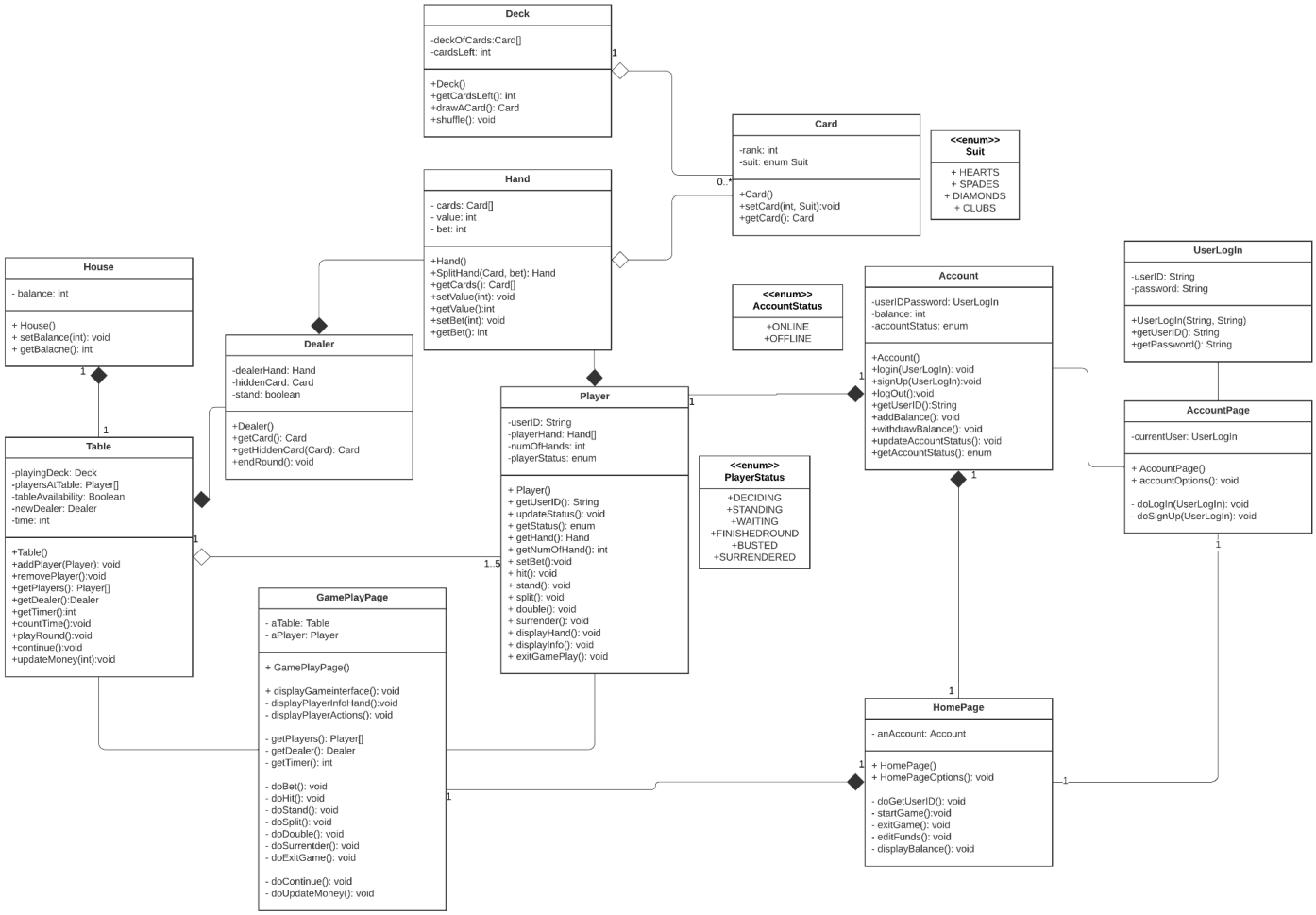
Use Case ID: 9

- Use Case Name: Stand
- Relevant Requirements: Game Play Screen, Player
- Primary Actor: Player
- Pre-conditions: The player is not busted
- Post-conditions: The dealer skips the player and deals with the next player
- Basic Flow or Main Scenario:
    4. The player clicks on the "Stand" button
    5. The dealer skips the player and deals with the next player

Reference 2: UML Use Case Diagram

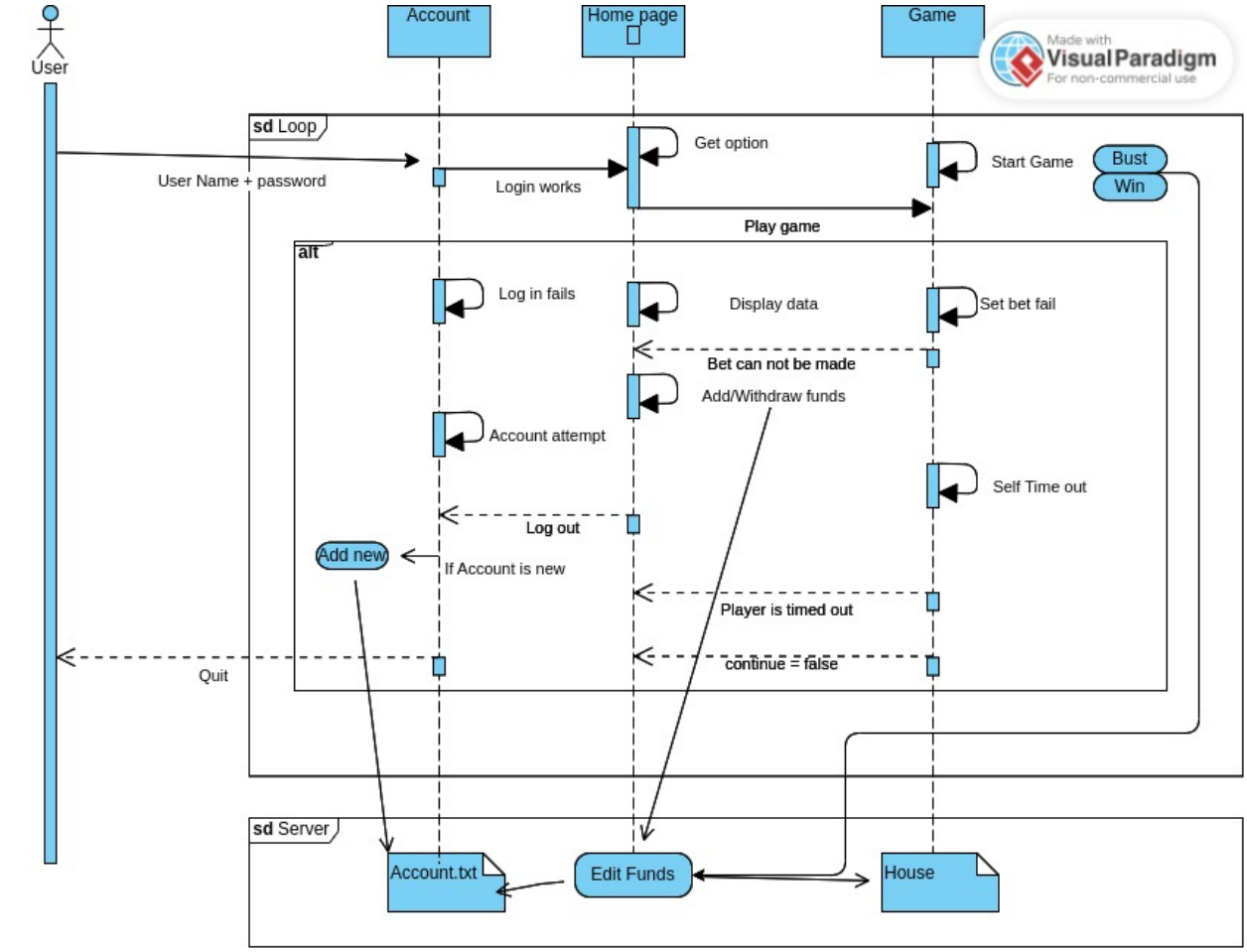


Reference 3: Class Diagrams

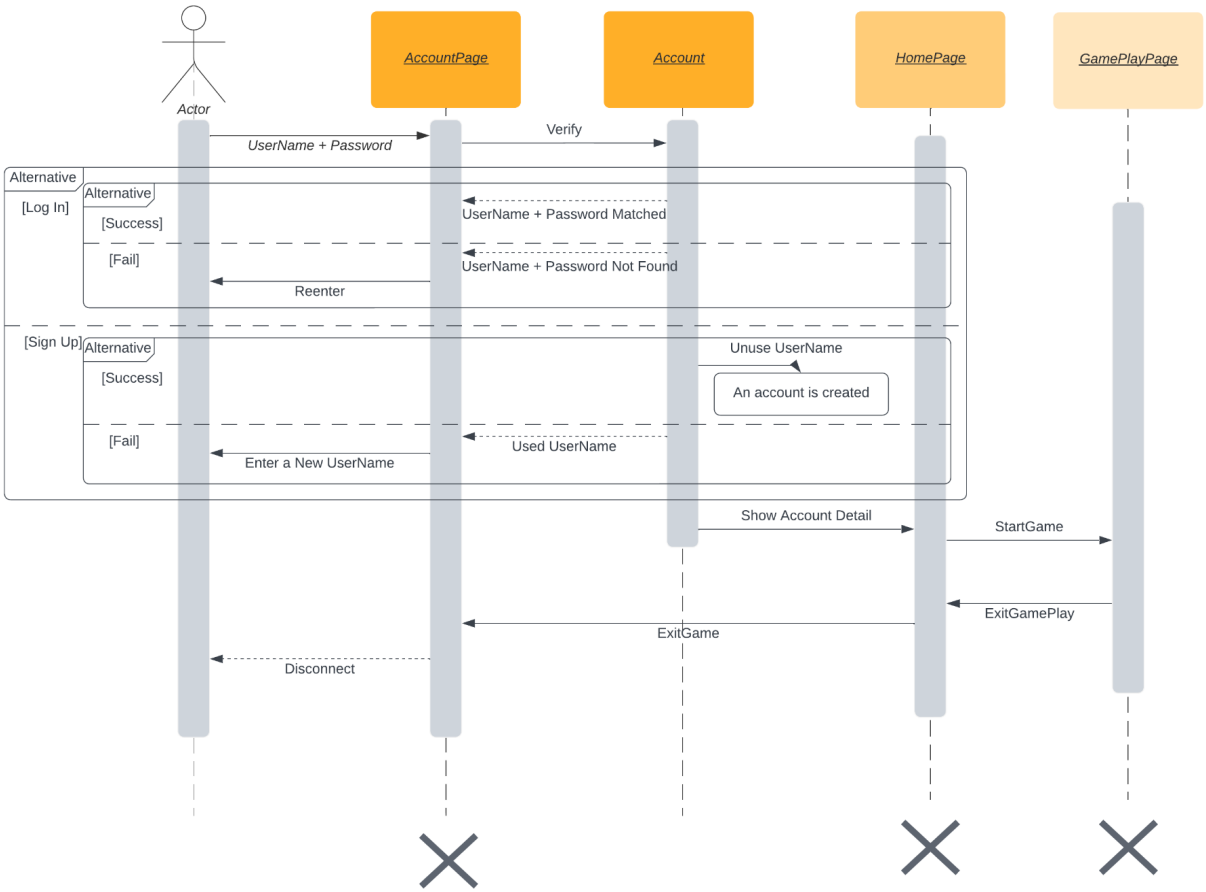


Reference 4: Sequence Diagrams

4.1 High Level View of Entire Program



4.2 Log In And Sign Up Sequence



4.3 Game Play Sequence