Blackjack Design Document

CS 401 Group 2

Spring 2025

Revision History

Date	Revision	Description	Author
3/24/2025	1.0	Start of Design Document	Mahlet Bekele, Brandom Wall, Cristian Gonzales, Kenny Yu
4/6/2025	1.1	Added new graphics in interface design	Armand Tirado
4/6/2025	1.2	Created Mock-up images for UI Design, System Architecture, Dealer's point of view in game, and Player's point of view in game	Brandon Wall
4/6/2025	1.3	Worked on use case diagram	Mahlet Bekele
4/7/2025	1.4	Continued working on mock-up images	Brandon Wall
4/7/2025	1.5	Worked on UML class diagrams	Brandon Wall
4/7/2025	1.6	Worked on UML class diagrams	Mahlet Bekele
4/8/2025	1.7	Worked on UML class diagrams	Brandon Wall, Kenny Yu, Cristian Gonzales
4/8/2025	1.8	Worked on login, Main menu and fund manager UI	Mahlet Bekele
4/9/2025	1.9	Worked on the design document	Brandon Wall
4/9/2025	2.0	Worked on design document	Brandon Wall
4/9/2025	2.1	Worked on client server sequence diagram and use cases	Cristian Gonzales
4/10/2025	2.2	Worked on design document	Cristian Gonzales
4/10/2025	2.3	Worked on the design document	Brandon Wall
4/10/2025	2.4	Worked on UML class diagrams	Brandon Wall

Table of Contents

1. Purpose

1.1. Scope

This document will catalog the user, system, and hardware requirements for the Multiplayer Blackjack Gaming System. It will define what the system must accomplish but will not detail how these requirements will be implemented.

1.2. Definitions, Acronyms, Abbreviations

1.2.1. Hand: The collection of cards each player possesses during the game.
1.2.2. Table: A game room consisting of one dealer and 0-6 players.
1.2.3. Dealer: The person who distributes cards to active players and plays against them.
1.2.4. Blackjack: A hand consisting of two cards whose sum equals 21.
1.2.5. Bet: The amount of money the player wishes to wager that they'll win against the dealer
1.2.6. Table Limit: The minimum and maximum amount players can bet at a table.
1.2.7. Hit: An action made by a player that will adds the top card of the deck into the player's hand.
1.2.8. Stand: An action made by the player when the player wishes to end their turn.
1.2.9. Fold: When a player decides to forfeit the game, losing all money they've bet for that game.
1.2.10. Bust: When a player hand total is over 21 resulting in a loss.
1.2.11. Card: An object that has a numerical value and a card symbol that is either a spade, diamond, heart, or club. Numerical values range from 1-11.
1.2.12. Card Symbol: Each card has a card symbol, which is either a spade, diamond, heart, or club.
1.2.13. Ace Card: A variant of a normal card that has a numerical value of either 1 or 11 by the player's choice and has a card symbol.
1.2.14. Face Card: A variant of a normal card but has a numerical value of 10 and is called either a Jack Queen or King. Face cards also have one of the four card symbols.
1.2.15. Deck: A collection of 52 unique cards, where there are four versions of each card in the deck, each with one of the four card symbols.
1.2.16. Shoe: A collection of multiple decks.
1.2.17. Push: When the sum of the player's hand is equal to the sum of the dealer's hand.
1.2.18. TCP/IP (Transmission Control Protocol/Internet Protocol): a group of network protocols that let the client and the server communicate.
1.2.19. GUI (Graphical User Interface): the game's visual interface that allows players to interact with it.

1.3. References

1.3.1. UML Class Diagrams: Refer to Blackjack-UML-Class Diagrams.pdf
1.3.2. GitHub Repository: Refer to: https://github.com/bwalldev/CS401_BlackJackGroupProject
1.3.3. Gantt Chart: Refer to Gantt project planner.xlsx

1.4. Overview

1.4.1. The multiplayer blackjack gaming system is a Java-based application with a GUI, enabling real players to join virtual tables, place bets, and play under standard blackjack rules. It operates over TCP/IP, ensuring real-time gameplay, fund transactions, fair play, and minimal cheating.

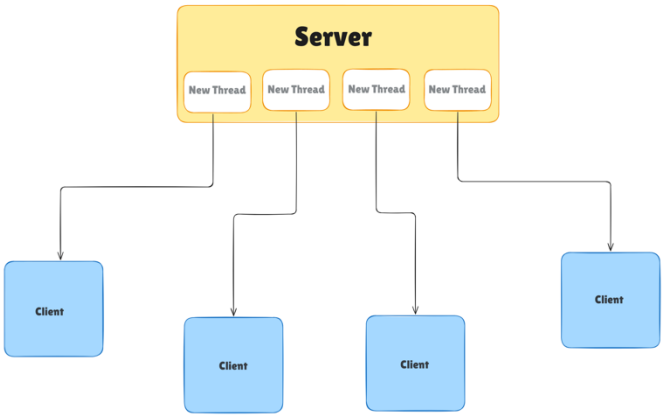
2. System Architecture

2.1. Network

2.1.1. The system will have a Client-Server design pattern to handle the network needs of the application.

2.2. Server

2.2.1. Game logic will take place on the server.
2.2.2. The Server with be multithreaded, which will spawn a new thread for each client, so that the application supports multiple players playing simultaneously. The image below illustrates the visualization:



2.2.3. The Servers main thread will handle incoming network connections.

2.3. Client

2.3.1. The Client will be multithreaded so that the UI and game properties update in real time.
2.3.2. A thread for handling connections to the Server will also be needed.
2.3.3. The Client will retrieve data from the Server by sending and receiving messages from the Server.
2.3.4. The Client will communicate with the GUI to update game properties.

2.4. Data Storage

2.4.1. Player and Dealer login credentials will be stored on the server in a text file. When a client initializes to login, a message will be sent from the client containing their login credentials to the server for authorization.
2.4.2. Only the server will have access to stored user data files

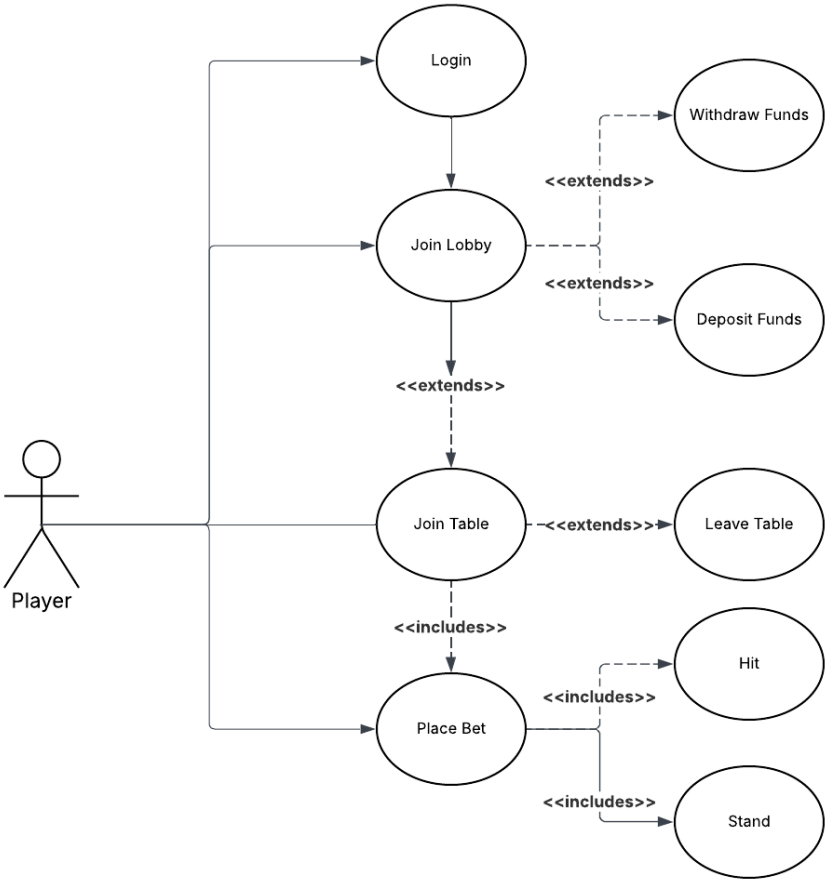
3. UML Class Diagrams

A large-scale visualization of the class diagrams is provided and can be found in an external file titled "Blackjack-UML-Class-Diagram.pdf", referred to in section 1.3.1 of the references.

4. Use Case Diagrams

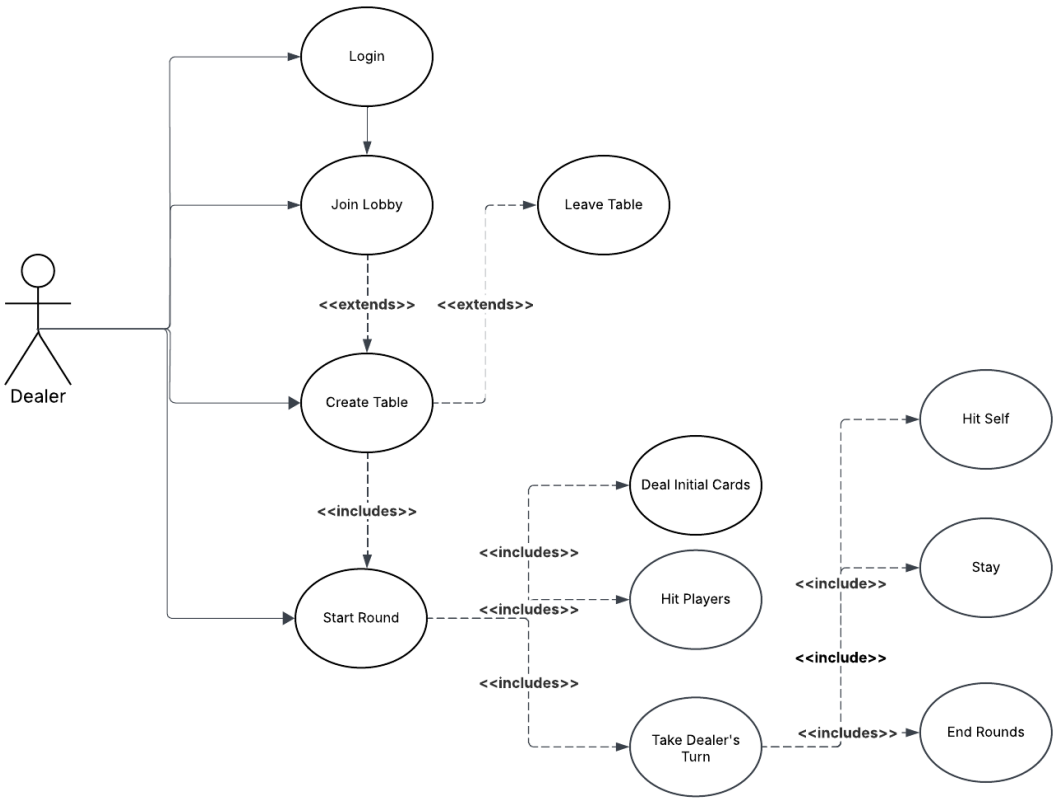
4.1. Player Use Case Diagram

4.1.1. This Player use case diagram shows the different actions a player can do when interacting with the server. It starts with authentication of user credentials to participate in an actual Blackjack lobby and joining a table. When at a table, the player must place a bet to play the game. They are then able to hit/stay during their turn.



4.2. Dealer Use Case Diagram

4.2.1. The dealer use case diagram shows the use cases for the dealer. It begins by having the dealer log in to the server and join the lobby. The dealer will then create a table, which they can leave and start the round of Blackjack. During the round start, they will have to deal cards to the players when they request it. After all players' turns are over, the dealer will play with their own hand, where they can hit/stay following Blackjack rules. The round then ends when the dealer's hand is finished, and players will be paid out.



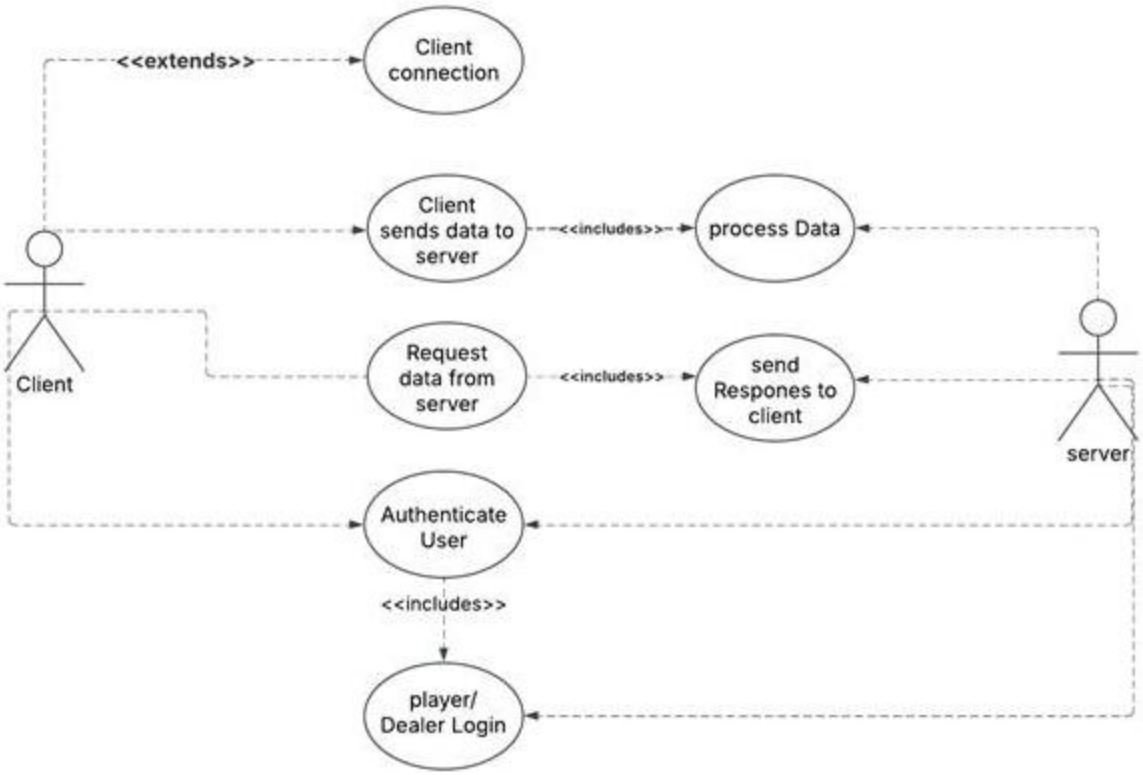
4.3. Client-Server Use Case Diagram:

4.3.1. The Client starts the connection, authenticates, sends, and requests data.

4.3.2. The Server handles authentication, stores or processes the received data, and responds to client requests.

4.3.3. <<includes>> - shows relationships of dependent use cases that are part of a larger action.

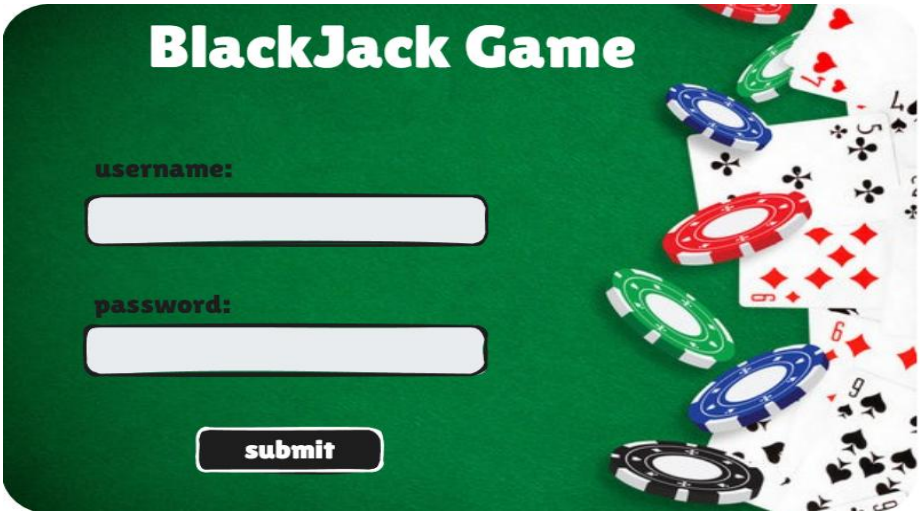
4.3.4. <<extends>>—represents optional, conditional behavior that enhances or adds extra functionality.



5. Interface Mock-up Designs

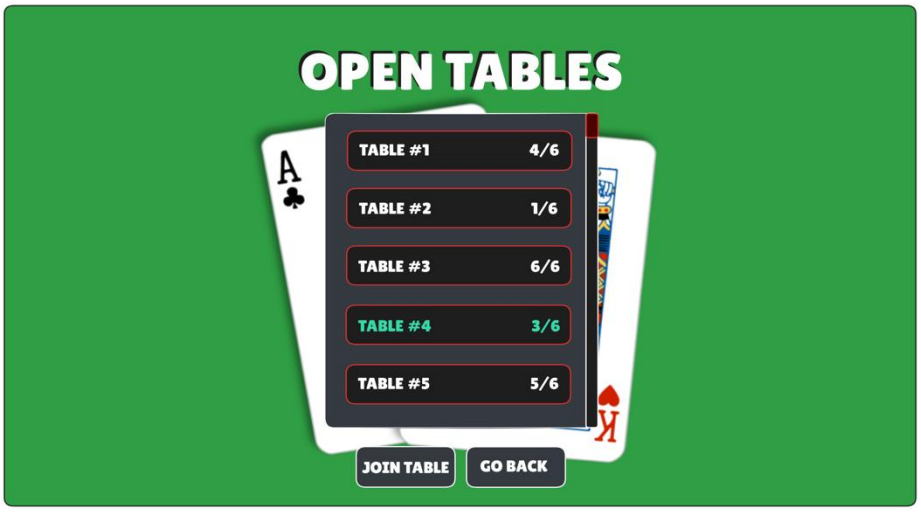
5.1. Launch Menu

5.2. Login screen



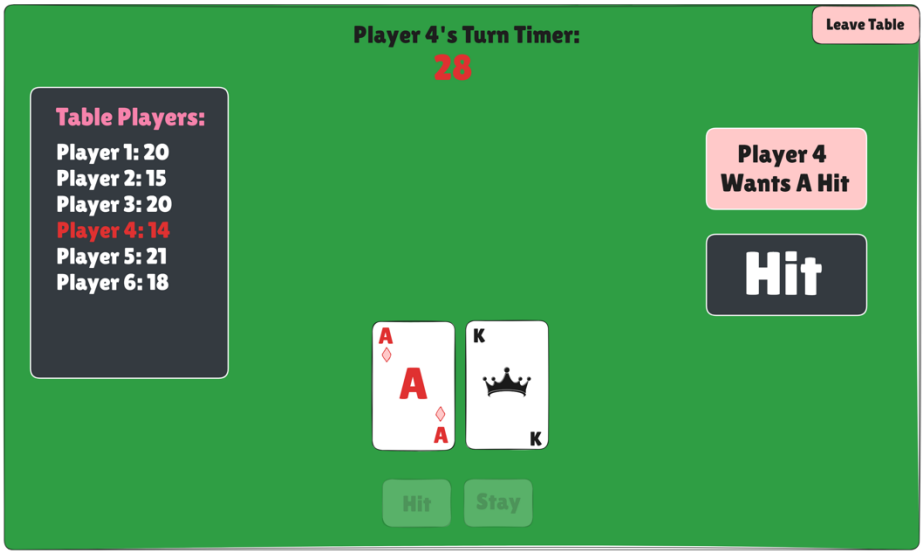
5.3. Player's / Dealer's Main menu

5.4. Lobby Menu

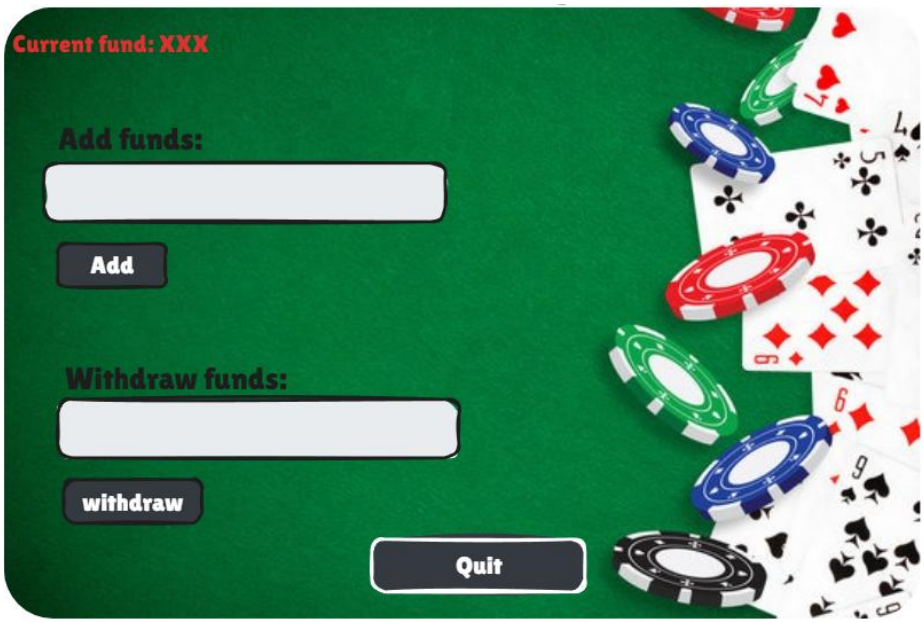


5.5. Player's point of view

5.6. Dealer's point of view



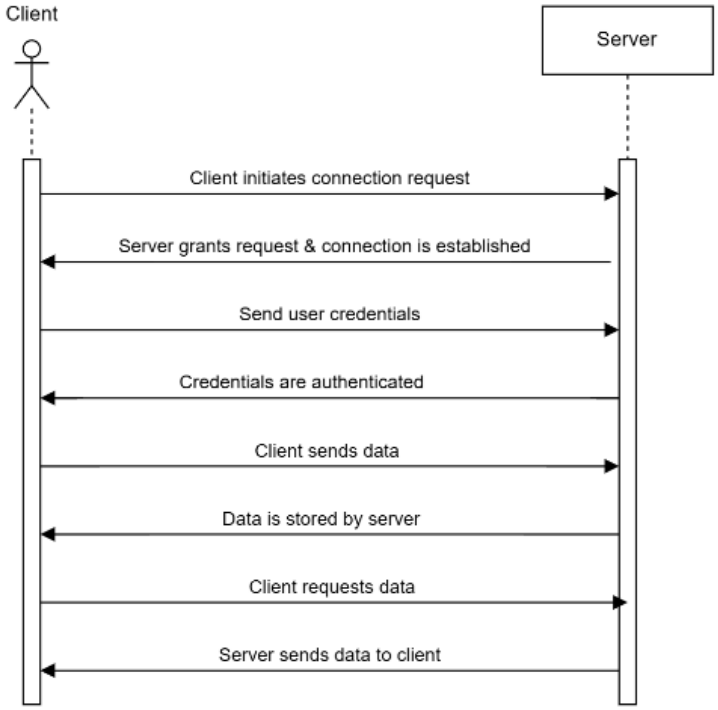
5.7. Players Fund Management



6. Sequence Diagrams

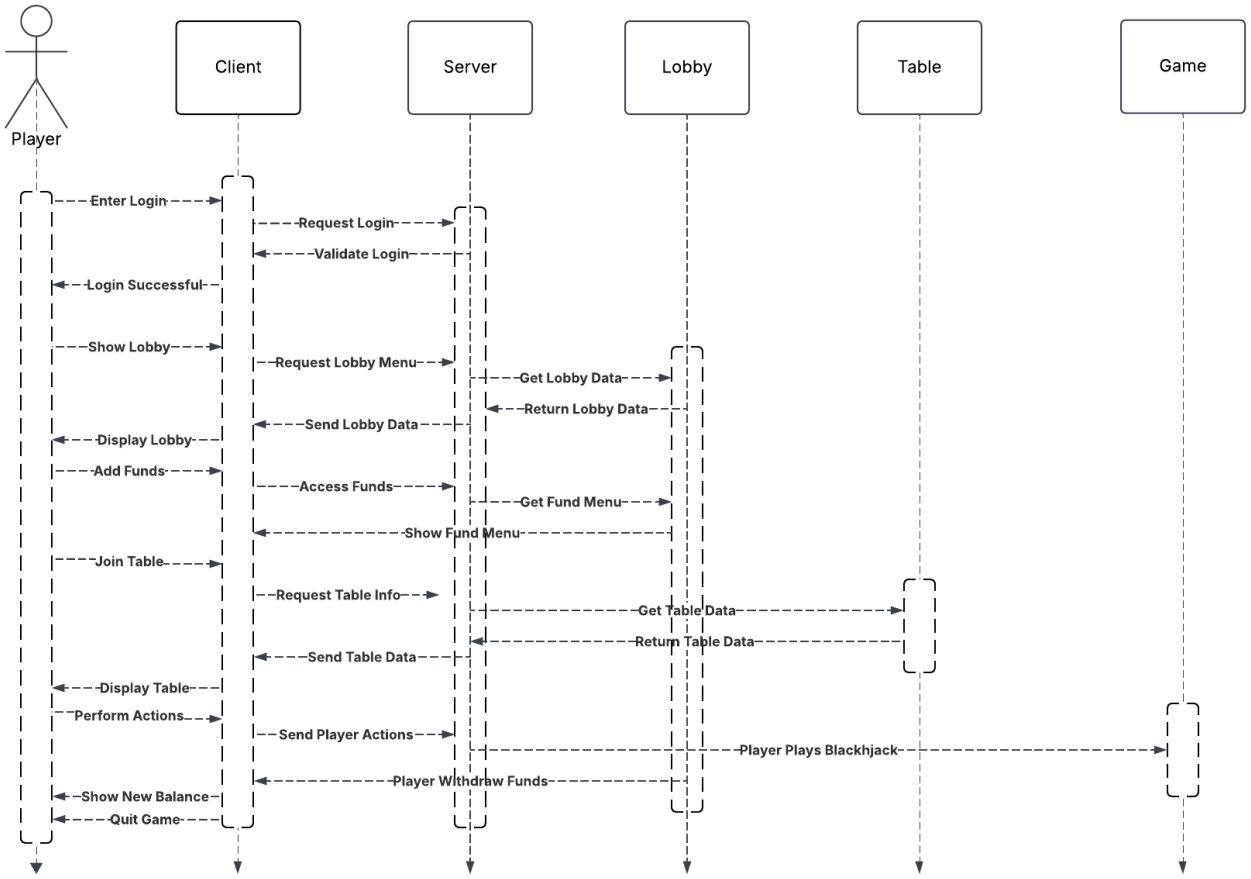
6.1. Client-Server Sequence

6.1.1. Client initiates interaction by sending a request to the server
6.1.2. Server acknowledges request confirming the connection is made
6.1.3. Client submits credentials
6.1.4. Server processes credentials to show if it was a success or failure
6.1.5. Once authenticated client sends or requests data
6.1.6. The server responds by confirming that it has received sent data or responds with the data requested from client



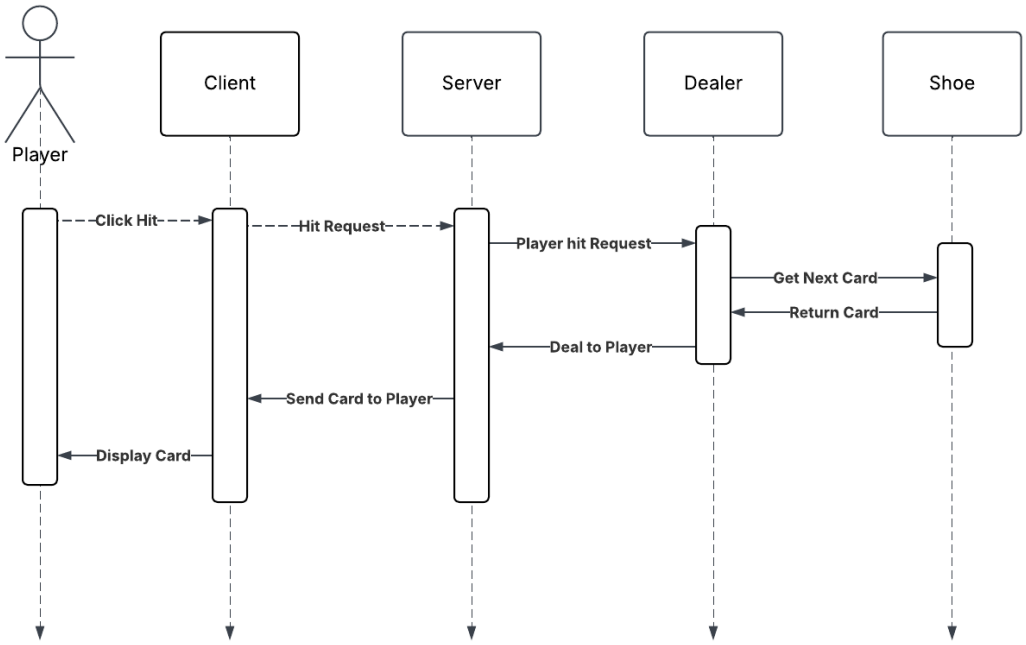
6.2. Player Login Sequence

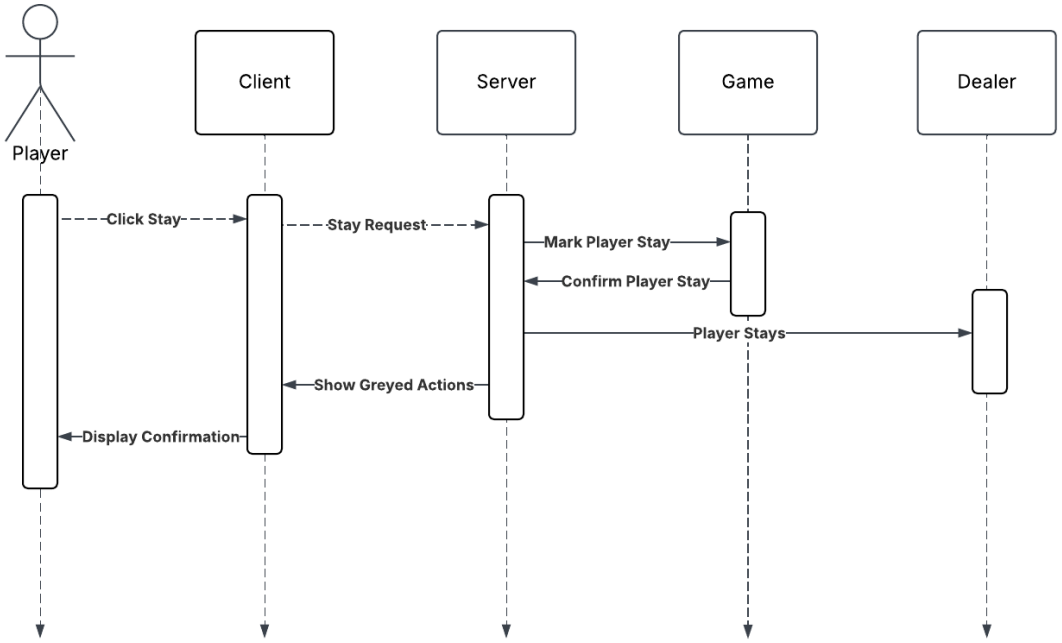
6.2.1. The following diagram shows the player login sequence. Player first requests to log in through the client, and the client sends a request for authentication to the server. Every action that player makes goes through the client, then to the server, then to the modules that the player wants to interact with. Then it will send it back to the client and eventually back to the player.

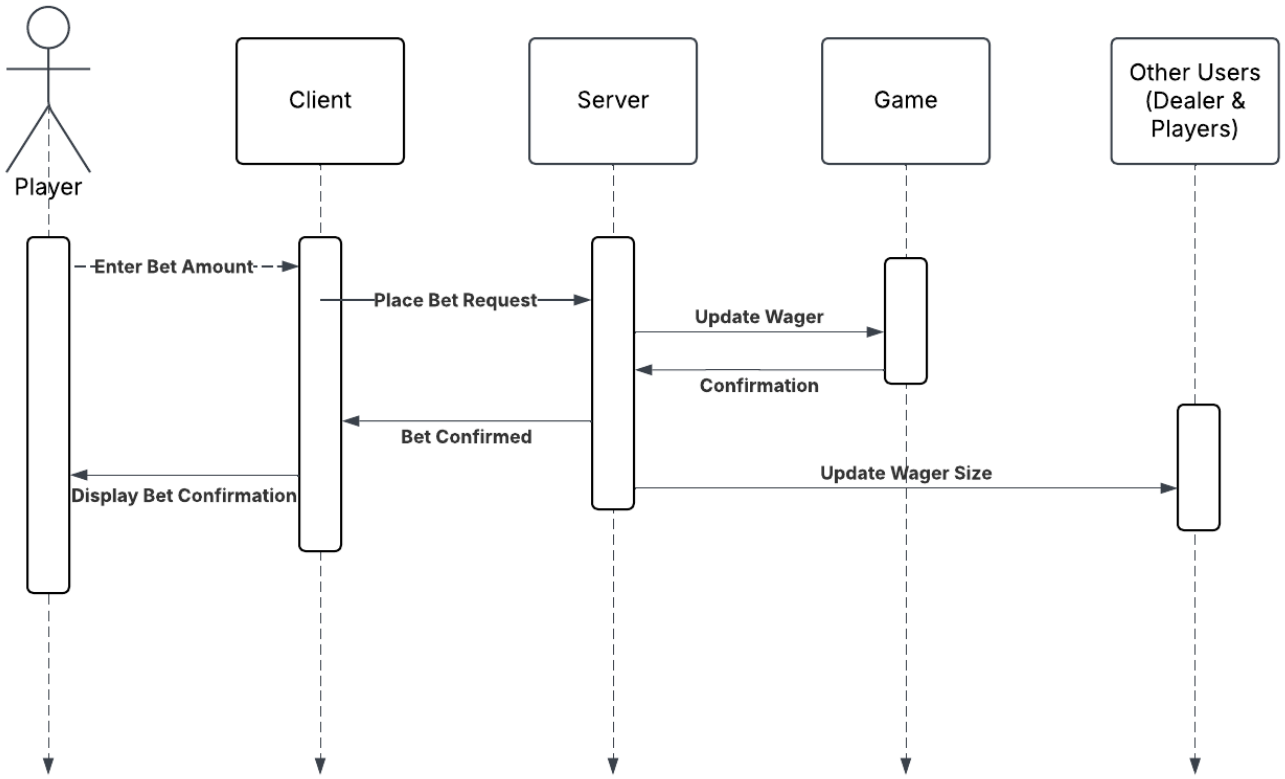


6.3. Player Round Sequence

6.3.1. The following sequence diagrams show the interactions of the player's turn and the possible moves that they can make. One is for the player to request a hit to the client, which then sends a message to the server for a hit request and then asks the dealer to hit. It will send it back to the player and display the card. Another is for the player to stay, which is like hit, but instead of receiving a card, it will just confirm by greying out the hit button after they choose to stay. Lastly, the player places a bet, which then sends a message to the server from the player's client. The server will also update the bet size/pool to the other users, which are the dealer and players.

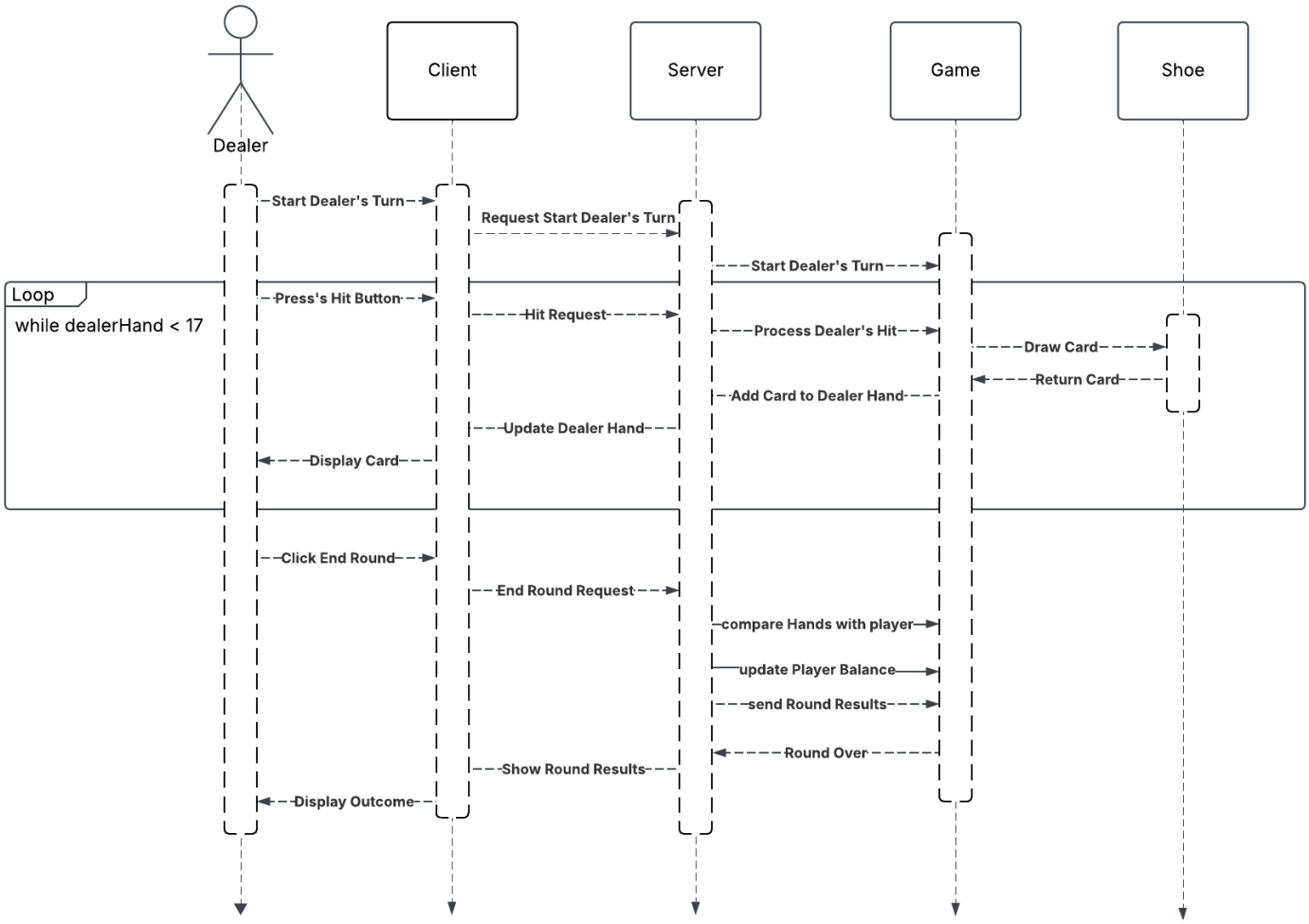

Player
Client
Server
Dealer
Shoe
Click Hit
Hit Request
Player hit Request
Get Next Card
Return Card
Deal to Player
Send Card to Player
Display Card


Player
Client
Server
Game
Dealer
Click Stay
Stay Request
Mark Player Stay
Confirm Player Stay
Player Stays
Show Greyed Actions
Display Confirmation



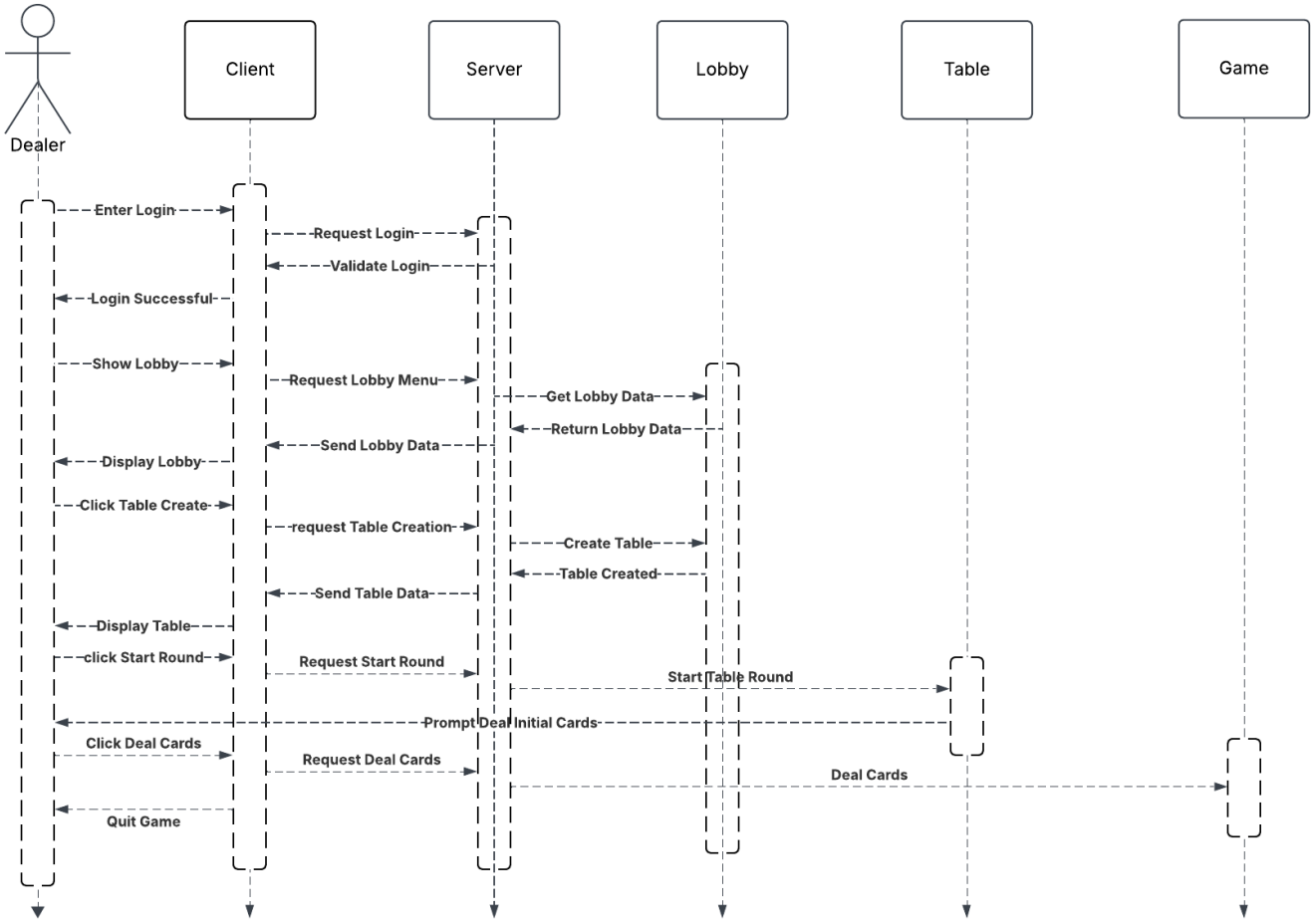
6.4. Dealer Round Sequence

6.4.1. The following sequence diagram shows the interactions of the dealer's turn. After the end of the last player's turn, the dealer would start its own turn. In here there is a while loop that when the dealer's hand is less than 17, the dealer must continuously request a hit for a new card. After that is finished and the cards are displayed to the dealer's client, the round will end, and the dealer's and player's hands will be compared. It will then end the round and display the outcome.



6.5. Dealer Login Sequence

6.5.1. The following sequence diagram shows the dealer login sequence. Dealer first requests to log in through the client, and the client sends a request for authentication to the server. Every action that dealer makes goes through the client, then to the server, then to the modules that the dealer wants to interact with. Then it will send it back to the client and eventually back to the dealer. The main difference here is the dealer requests to create a table and deals out the initial cards for the players.



7. Requirement Adjustments

7.1. Requirements Removed

7.1.1. The GUI should have animations for shuffling cards, card dealing, and chip movements to enhance player immersion.

7.1.2. The system must have an immersive and visually appealing GUI for players to use throughout the game to keep them enthralled.

7.2. Requirements Modified

7.2.1. The internal interface must support all possible player actions, such as hit, stand, double down, split, and leave game.

7.2.1.1. The requirement is now: The internal interface must support Player actions such as hit, stand, leave game, join table, add funds, and withdraw funds.