# Multiplayer Blackjack Game

Software Requirements Specification

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 02/25/2025 | 1.0 | Worked on section 1 | Basim Shahzad |
| 02/26/2025 | 1.1 | Worked on section 2 | Basim Shahzad |
| 02/26/2025 | 1.2 | 3.1.1 Common Requirements  3.1.2 Multiplayer Module Starter  Added 3.1.2.1 – 3.1.2.5  3.1.3 Stats Tracker Module Starter  Added 3.1.3.1 | Riley Fischer |
| 02/26/2025 | 1.3 | Worked on Section (4.1 & 4.2) | Basim Shahzad |
| 02/26/2025 | 1.4 | Added 3.2.1, 3.2.2 and 3.3.1 | Riley Fischer |
| 02/27/2025 | 1.5 | Added 3.1.3, 3.1.4 | John Nguyen |
| 02/28/2025 | 1.6 | 3.1.5 Game Module Requirements  Added 3.1.4.2, 3.1.4.5 – 3.1.4.8  Removed 3.1.4.1  Added 3.1.3.4, 3.1.3.5 | Martin Garcia |
| 02/28/2025 | 1.7 | Added intial ideas for user case document | John Nguyen |
| 03/03/2025 | 1.8 | Worked on section 4.3 | Martin Garcia |
| 03/03/2025 | 1.9 | Finished Use Case Specifications Document | Basim Shahzad |
| 03/06/2025 | 1.10 | Filled in the references section to include Sequence, Use Case, and UML class diagrams. | Riley Fischer |
| 03/06/2025 | 1.11 | Inserted Sequence Diagrams into the document | Michelle Nguyen |
| 03/06/2025 | 1.12 | Reformatted the references section | Riley Fischer |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Purpose iv](#_Toc192177534)

[1.1. Scope iv](#_Toc192177535)

[1.2. Definitions, Acronyms, Abbreviations iv](#_Toc192177536)

[1.3. References iv](#_Toc192177537)

[Use Case Specification – Blackjack Use Case Doc iv](#_Toc192177538)

[1.4. Overview iv](#_Toc192177539)

[2. Overall Description v](#_Toc192177540)

[2.1. Product Perspective v](#_Toc192177541)

[2.2. Product Architecture v](#_Toc192177542)

[2.3. Product Functionality/Features v](#_Toc192177543)

[2.4. Constraints v](#_Toc192177544)

[2.5. Assumptions and Dependencies v](#_Toc192177545)

[3. Specific Requirements vi](#_Toc192177546)

[3.1. Functional Requirements vi](#_Toc192177547)

[3.1.1. Common Requirements: vi](#_Toc192177548)

[3.1.2. Multiplayer Module Requirements: vi](#_Toc192177549)

[3.1.3. Stats Tracker Module Requirements vi](#_Toc192177550)

[3.1.4. Cheating Mitigation Module Requirements: vi](#_Toc192177551)

[3.2. External Interface Requirements vii](#_Toc192177552)

[3.3. Internal Interface Requirements vii](#_Toc192177553)

[4. Non-Functional Requirements viii](#_Toc192177554)

[4.1. Security and Privacy Requirements viii](#_Toc192177555)

[4.2. Environmental Requirements viii](#_Toc192177556)

[4.3. Performance Requirements viii](#_Toc192177557)

[5. Use Case Specifications for Blackjack Application 9](#_Toc192177558)

[6. UML Use Case Diagrams 14](#_Toc192177559)

[6.1. Player/Dealer Login 14](#_Toc192177560)

[6.2. Manage Account Funds 15](#_Toc192177561)

[6.3. Join a Game 16](#_Toc192177562)

[6.4. Make a Bet 17](#_Toc192177563)

[6.5. Play a Turn(Hit/Stand/Double/Down/Split 18](#_Toc192177564)

[6.6. Dealer Turn Execution 18](#_Toc192177565)

[6.7. Determine Game Outcome 19](#_Toc192177566)

[6.8. Update Leaderboard 19](#_Toc192177567)

[6.9. Detect and Prevent Cheating 20](#_Toc192177568)

[6.10. Player/Dealer Disconnect 20](#_Toc192177569)

[7. UML Class Diagram(s) 21](#_Toc192177570)

[8. Sequence Diagrams 23](#_Toc192177571)

[9. GitHub Repo Link 26](#_Toc192177572)

[9.1. GitHub Repo Link: https://github.com/CSUEB-Black-Jack-Group-5/CS401-BlackJack.git 26](#_Toc192177573)

# Purpose

The purpose of this document is to outlines the software requirements for the Multiplayer Blackjack Game. The document details functional and non-functional requirements, system architecture, and constraints.

## Scope

The Multiplayer Blackjack Game is a Java-based application with a GUI that operates over TCP/IP. It will include the following core features:

* Multiplayer functionality (6 players max per game)
* Real players as dealer and players (no bots)
* Account management (fund deposits & withdrawals)
* Anti-cheating mechanisms (card counting detection, random shoe replacement)
* Game logic( betting, turns, win/lose conditions)
* Leaderboard for tracking player stats
* Lobby system for concurrent games

## Definitions, Acronyms, Abbreviations

* GUI: Graphical User Interface
* TCP/IP: Transmission Control Protocol/ Internet Protocol
* Shoe: A collection of multiple decks used in Blackjack to prevent card counting
* Dealer: The player responsible for dealing cards in the game
* Card Counting: A cheating strategy where players keep track of the cards already dealt from a deck, to determine their betting decisions.

## References

## Use Case Specification [Section 5, Page 9]

UML Use Case Diagram(s) [Section 6, Page 14]

Class Diagram(s) [Section 7, Page 21]

Sequence Diagram(s) [Section 8, Page 23]

## Overview

This document provides a detailed overview of the system architecture, constraints, functional and non-functional requirements for the Multiplayer Blackjack Game.

# Overall Description

## Product Perspective

This application is a multiplayer game that runs on client-server architecture. The system will consist of:

* Client application: GUI based interface for players and dealers.
* Server Application: Handles the current state of the game, player action, and turn management.

## Product Architecture

The blackjack game will consist of 5 modules: the multiplayer module, the stats tracker module, the cheating mitigation module, and the game module.

## Product Functionality/Features

* Multiplayer Gameplay: Players can join games and compete in Blackjack.
* Turn-Based System: Players take turns in sequential order, and players must make their move in a certain time limit.
* Funds Management: Players can deposit and withdraw money
* Game Logic: Betting, hitting, standing, and winning will be accurately implemented
* Anti-Cheating Logic: Card counting detection, and randomized shoe replacement.
* Leaderboard: Tracks player earnings and win/loss records

## Constraints

* No bots allowed; all participants must be real people
* Maximum of 6 players per game
* Only as many tables as there are available dealers
* No spectator mode
* No communication amongst players

## Assumptions and Dependencies

* Players will follow the rules outlined to them
* Players will have a stable internet connection

# Specific Requirements

## Functional Requirements

### Common Requirements:

3.1.1.1 Users should be able to log in as either players or dealers.

3.1.1.2 The game should be fully GUI driven.

3.1.1.3

### Multiplayer Module Requirements:

3.1.2.1 No bots are allowed to be playing. It should be played purely by humans.

3.1.2.2 There should be exactly 1 dealer and 6 players per table.

3.1.2.3 There should always be the same number of active tables as there are dealers.

3.1.2.4 The gameplay should follow a turn-based structure. One player must be active at a time with the dealer.

3.1.2.5

### Stats Tracker Module Requirements

3.1.3.1 Each player should have full history of all the games they’ve played.

3.1.3.2 Transaction history like add funds or withdrawl money from players.

3.1.3.3 Players, dealers lose or win history

3.1.3.4 The system should record and display the following stats of each player:

* Total games played
* Total wins, losses, and pushes
* Win/loss percentages
* Highest winning in a single game
* Average bet per game

3.1.3.5 Game stats should be stored in a database (CSV file).

### Cheating Mitigation Module Requirements:

3.1.4.1 The System should prevent card counting by implementing an automatic reshuffle when 50% of the second deck has been used

3.1.4.2 Switching dealer for every table every 30 – 40 rounds.

3.1.4.3 Players should not be able to modify or inject any client-side code to manipulate the game state, card distribution, or betting system.

3.1.4.4 The system should monitor betting patterns and flag suspicious activity.

3.1.4.5 Players should not be allowed to disconnect or refresh from the game to manipulate outcomes.

3.1.4.6 The system should ban players who break any of these rules.

* + 1. **Game Module Requirements**

3.1.5.1 The system should follow standard blackjack rules, where players attempt to reach a hand value of 21 without exceeding it.

3.1.5.2 The dealer must follow predefined rules, such as hitting on 16 or less and standing on 17 or higher.

3.1.5.3 Each game should support the following player actions:

* Hit: Draw an additional card.
* Stand: End turn without drawing more cards.
* Double Down: Double the bet and receive exactly one more card.
* Split: If the player has two cards of the same rank, they can split them into two separate hands (if the betting balance allows).
* Bust: Player exceeds 21

3.1.5.4 Players should be able to place bets within a predefined betting range before the game starts.

3.1.5.5 The game should handle blackjack payouts based on a 3:2 ratio and normal win payouts based on a 1:1 ratio.

3.1.5.6 A tie (push) should result in the player's bet being returned.

3.1.5.7 The system should enforce time limits for player actions to ensure smooth gameplay and prevent stalling.

3.1.5.8 The game state should persist in case of player disconnection. Forfeiting players bets.

3.1.5.9 The game should maintain a clear UI displaying:

* Player’s current hand and value.
* Dealer’s visible card(s).
* Available player actions.
* Current bet and balance.

## External Interface Requirements

3.2.1 The user interface for the client application must display all relevant information and provide means for the user to log-in, log-out, choose their user type (player or dealer), join a table, play blackjack.

3.2.2 The interface for the server application must log the activities of the active blackjack games.

## Internal Interface Requirements

3.3.1 The server should be responsible for processing player actions and inform the other players of what each other are doing. This will keep the game in sync.

# Non-Functional Requirements

## Security and Privacy Requirements

4.1.1 The SR8 system must validate user credentials before allowing logins for players or dealers

4.1.2 The player should not be able to modify their own or any other player’s game stats

4.1.3 Players should not be able to join during a game in progress, but players already in

a game should be able to leave.

## Environmental Requirements

4.2.1 The system must be a cross-platform application.

4.2.2 Main server will be hosted on one of the group member’s computers. SR25

## Performance Requirements

4.3.1 The system should respond to player and dealer actions within 500 milliseconds for smooth gameplay.

4.3.2 The game interface should load within 3 seconds with 30 Mbps or higher.

4.3.3 Game animations should not cause the game to lag in any way.

4.3.4 The system should handle at least 10 games with 6 players in each game without any noticeable lag.

4.3.5 Player stats, game history, bank access should be retrievable within 500 milliseconds.

# Use Case Specifications for Blackjack Application

**1. Use Case Overview**

**System**: Multiplayer Blackjack Game  
**Actors**: Player, Dealer, Game Server  
**Primary Goal**: Facilitate a multiplayer blackjack experience with betting, turn-based gameplay, and anti-cheating mechanisms.

**2. Use Case List**

1. **UC1: Player Login**
2. **UC2: Manage Account Funds**
3. **UC3: Join a Game**
4. **UC4: Place a Bet**
5. **UC5: Play a Turn (Hit/Stand/Double Down/Split)**
6. **UC6: Dealer Turn Execution**
7. **UC7: Determine Game Outcome**
8. **UC8: Update Leaderboard**
9. **UC9: Detect and Prevent Cheating**
10. **UC10: Player/Dealer Disconnect**

**3. Detailed Use Cases**

**UC1: Player Login**

**Actors**: Player  
**Preconditions**: The player must have an existing account.  
**Basic Flow**:

1. The player enters login credentials.
2. The system validates credentials.
3. The player is authenticated and enters the main lobby.

**Alternative Flows**:

* **Invalid Credentials**: System prompts re-entry of login information.
* **Account Locked**: System denies access after multiple failed attempts.

**UC2: Manage Account Funds**

**Actors**: Player  
**Preconditions**: Player is logged in.  
**Basic Flow**:

1. The player selects deposit or withdrawal.
2. The player enters an amount.
3. The system processes the transaction.
4. The player’s balance updates.

**Alternative Flows**:

* **Insufficient Funds**: System notifies the player.
* **Transaction Error**: System notifies the player of failure.

**UC3: Join a Game**

**Actors**: Player, Game Server  
**Preconditions**: Player is logged in with sufficient funds.  
**Basic Flow**:

1. The player selects a game lobby.
2. The system assigns the player to a game.
3. The game starts when enough players join.

**Alternative Flows**:

* **Lobby Full**: Player is directed to another table.
* **No Available Dealers**: Player is notified and waits.

**UC4: Place a Bet**

**Actors**: Player, Game Server  
**Preconditions**: The player is in an active game.  
**Basic Flow**:

1. The player selects an amount to bet.
2. The system deducts the bet amount from their balance.
3. The system confirms the bet and deals initial cards.

**Alternative Flows**:

* **Insufficient Balance**: Player is notified and must adjust the bet.

**UC5: Play a Turn (Hit/Stand/Double Down/Split)**

**Actors**: Player  
**Preconditions**: The player has placed a bet.  
**Basic Flow**:

1. The player selects an action (Hit, Stand, Double Down, Split).
2. The system updates the player’s hand accordingly.
3. The turn moves to the next player.

**Alternative Flows**:

* **Bust**: Player loses and turn ends.
* **Split Not Possible**: System denies the action.

**UC6: Dealer Turn Execution**

**Actors**: Dealer  
**Preconditions**: All players have completed their turns.  
**Basic Flow**:

1. The dealer reveals their hand.
2. The dealer follows predefined Blackjack rules.
3. The dealer plays their turn automatically.

**UC7: Determine Game Outcome**

**Actors**: Game Server  
**Preconditions**: Dealer’s turn is complete.  
**Basic Flow**:

1. The system compares all hands.
2. The system determines winners and losers.
3. Payouts are distributed accordingly.

**UC8: Update Leaderboard**

**Actors**: Game Server  
**Preconditions**: A game round is complete.  
**Basic Flow**:

1. The system updates player rankings based on earnings.
2. The leaderboard displays the updated stats.

**UC9: Detect and Prevent Cheating**

**Actors**: Game Server  
**Preconditions**: Game is in progress.  
**Basic Flow**:

1. The system monitors player actions for patterns.
2. If cheating is detected, the system intervenes (e.g., reshuffling deck, removing player).

**UC10: Player/Dealer Disconnect**

**Actors:** Player, Dealer, Game Server

**Preconditions:** The player or dealer is connected to an active game session

**Basic Flow:**

1. A player or dealer disconnects.

2. The Game Server detects the disconnection.

3. If a player disconnects:

* The Game Server temporarily holds their spot.
* If they reconnect within a time limit, they rejoin the game.
* If the time limit expires, they are removed from the game.

4. If the dealer disconnects:

* The Game Server assigns a new dealer or waits for reconnection.
* If no dealer available, the round is paused and players are notified.

5. The game state is updated accordingly.

**Alternative Flows:**

 Player Reconnects: The player resumes their session seamlessly.

 Dealer Reconnects: The game resumes as normal.

 Player Does Not Reconnect: The game continues without them, and they forfeit any bets.

 Dealer Does Not Reconnect: The system assigns a new dealer or ends the game.

**4. Summary**

This document provides an in-depth examination of the primary use cases in the Multiplayer Blackjack Game. The system ensures a fair and engaging gameplay experience by incorporating key features such as account management, multiplayer interactions, betting mechanics, dealer execution, and automated game outcome determinations.

Additionally, measures for anti-cheating detection and leaderboard updates enhance the integrity and competitiveness of the game. The game server enforces rules and fair play through systematic tracking and intervention when necessary.

By implementing these well-defined use cases, the Multiplayer Blackjack Game will deliver a smooth and enjoyable user experience, ensuring both fairness and competitive engagement. The next steps involve validating these use cases against system requirements and refining user interactions through iterative testing.

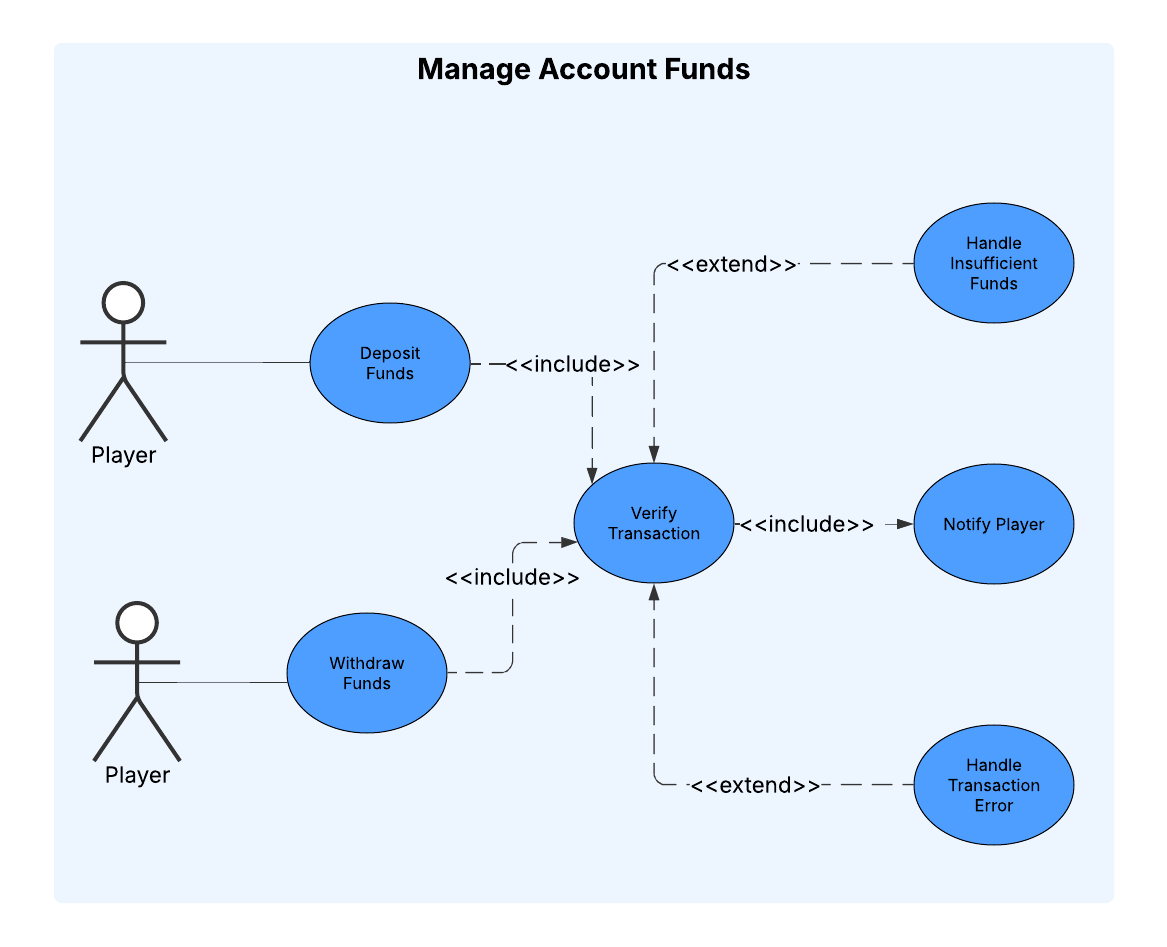
# UML Use Case Diagrams

## Player/Dealer Login

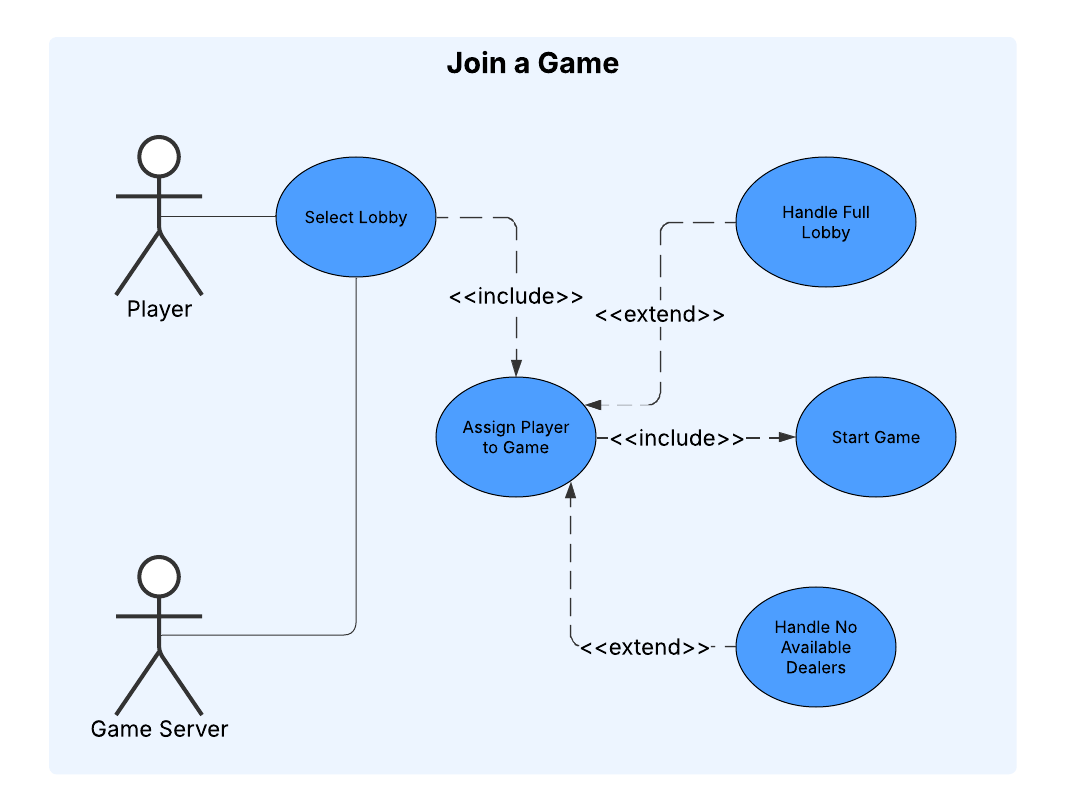
A diagram of a user interface

AI-generated content may be incorrect.

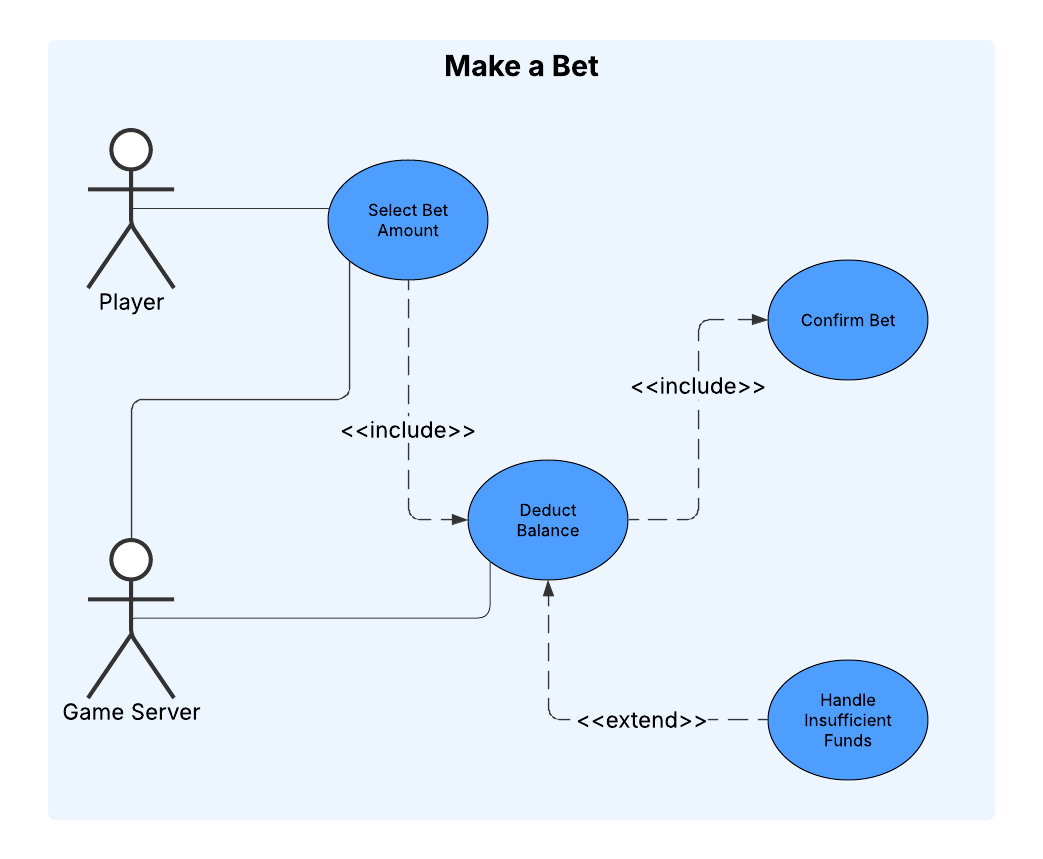
## Manage Account Funds



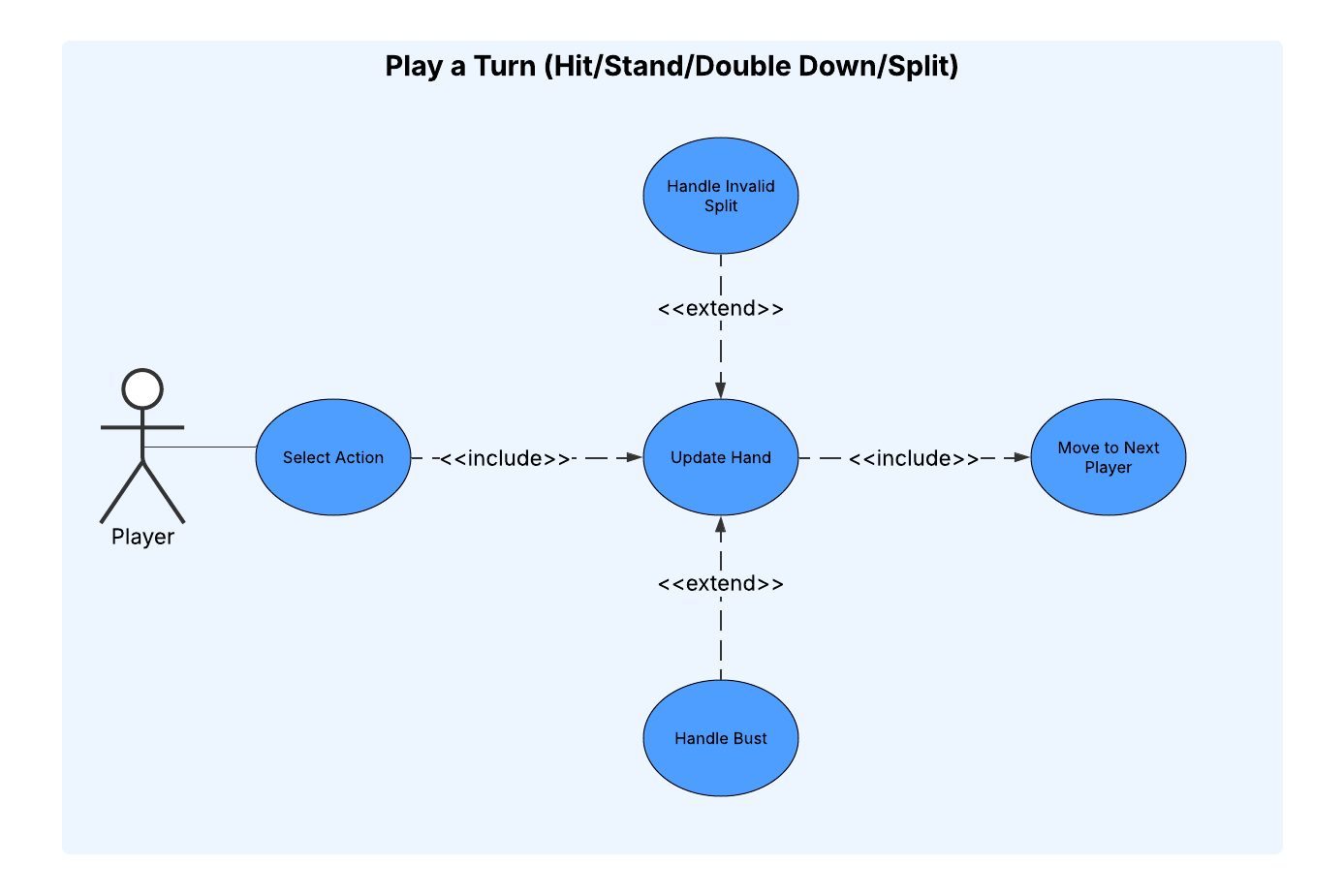
## Join a Game



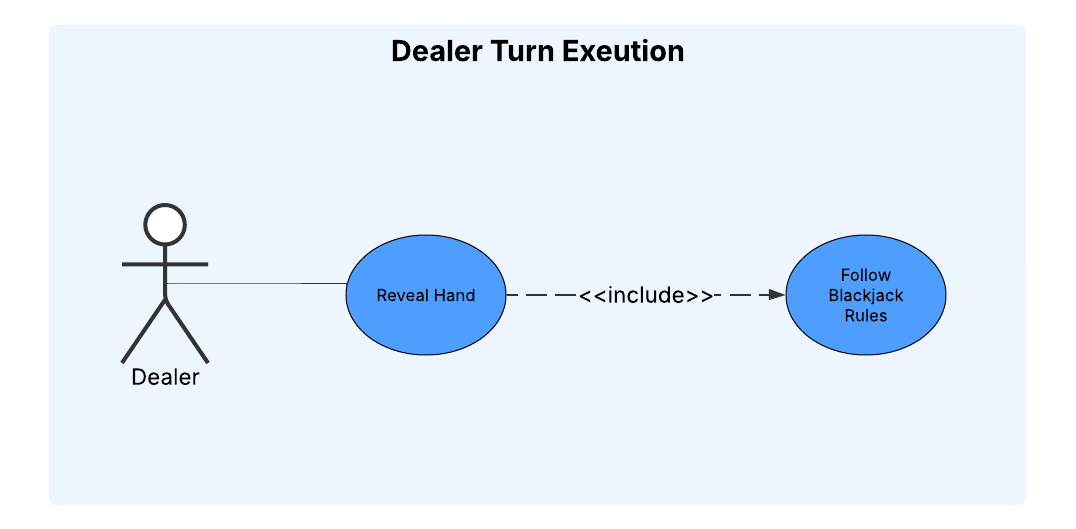
## Make a Bet



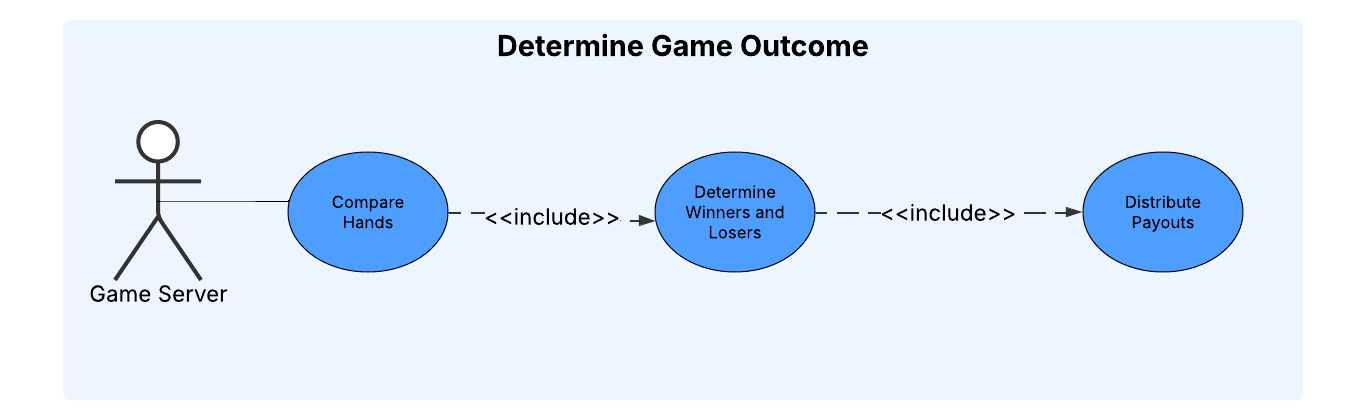
## Play a Turn(Hit/Stand/Double/Down/Split



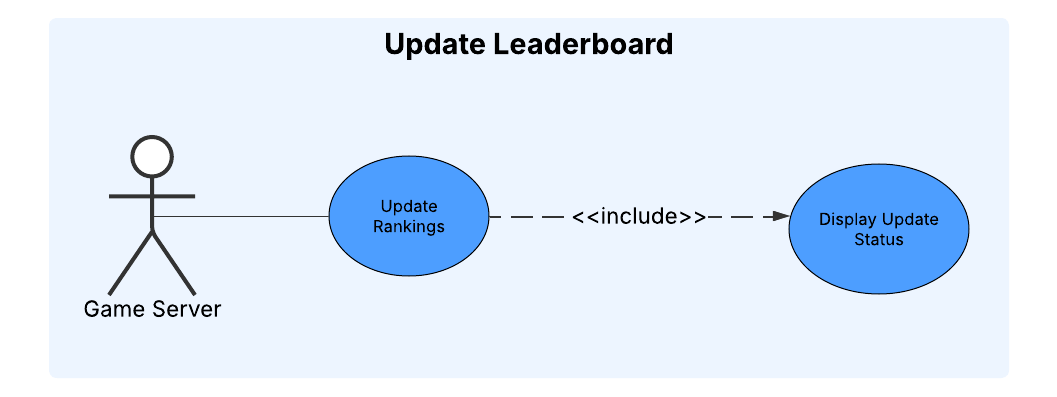
## Dealer Turn Execution



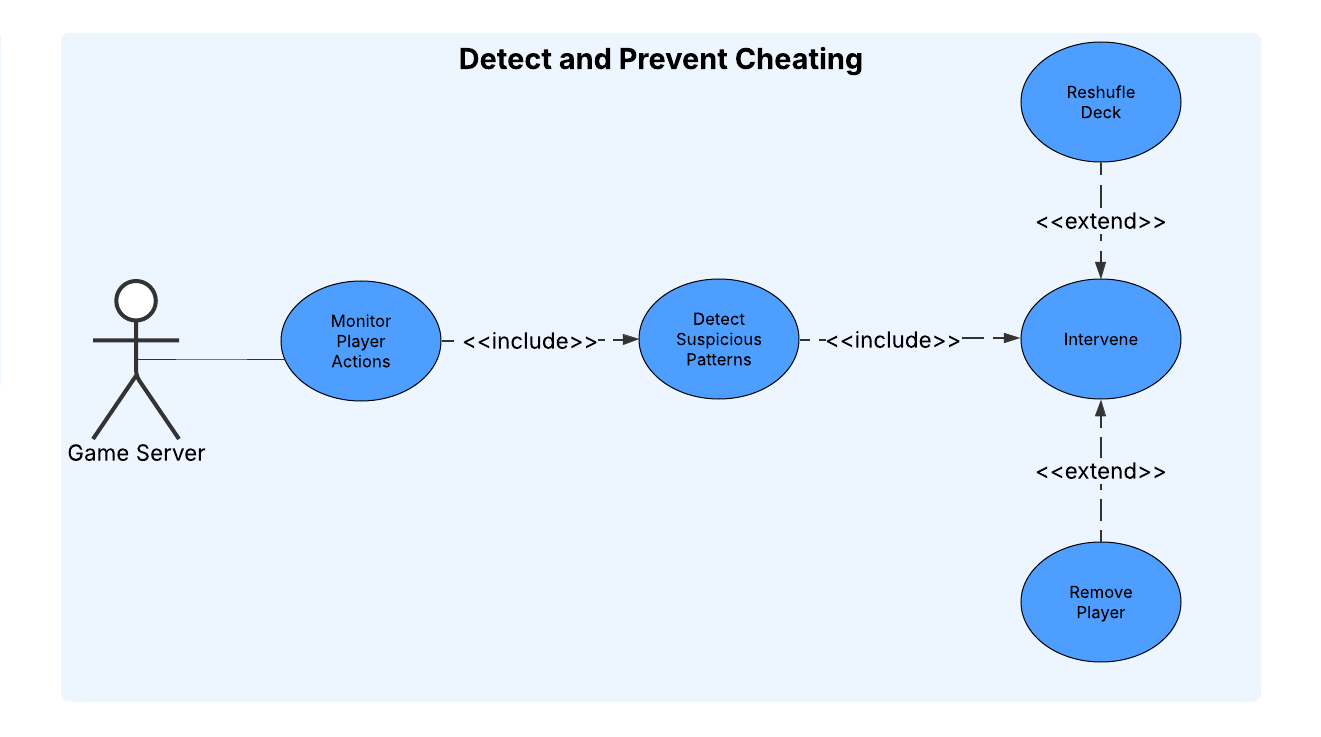
## Determine Game Outcome



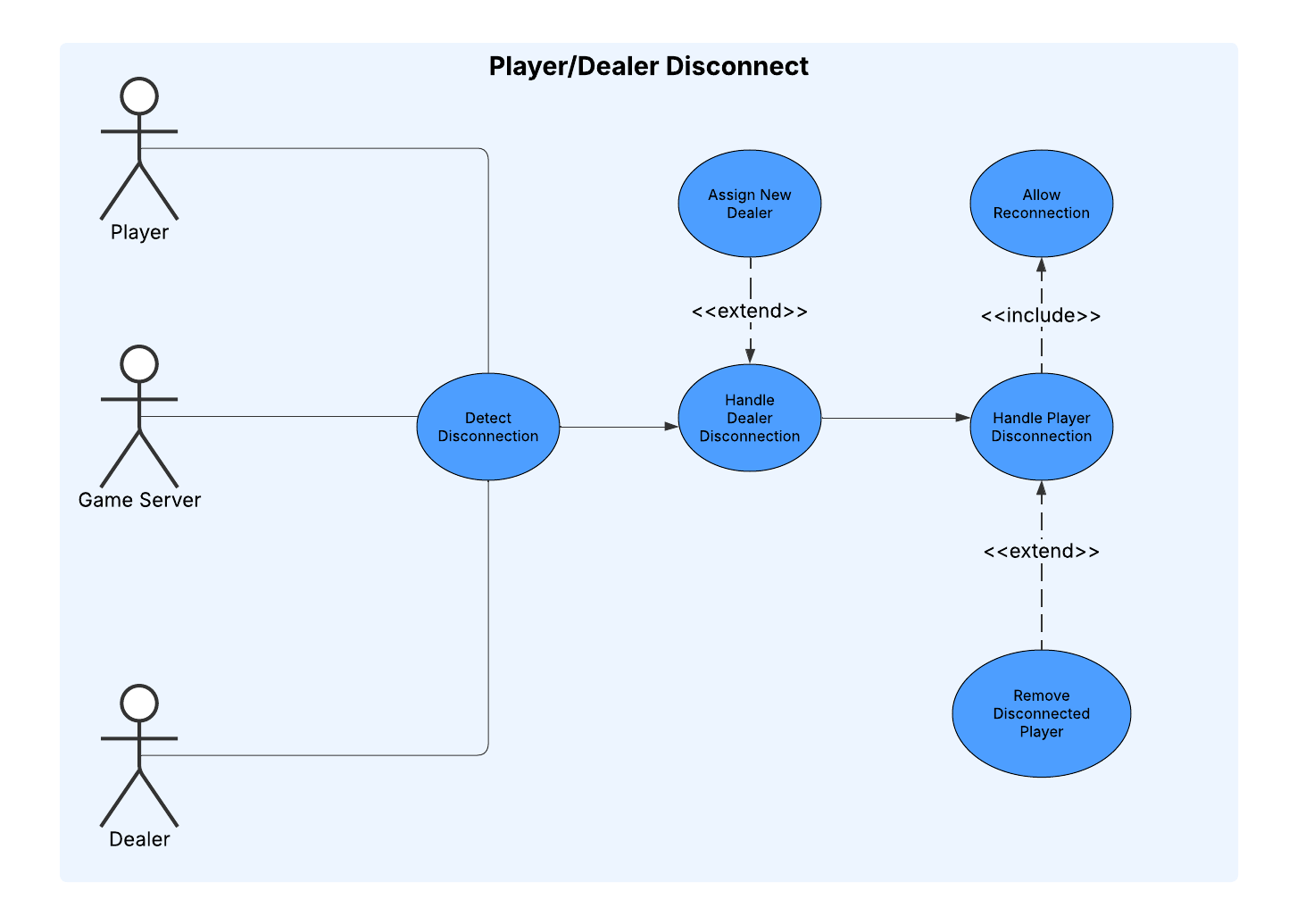
## Update Leaderboard



## Detect and Prevent Cheating



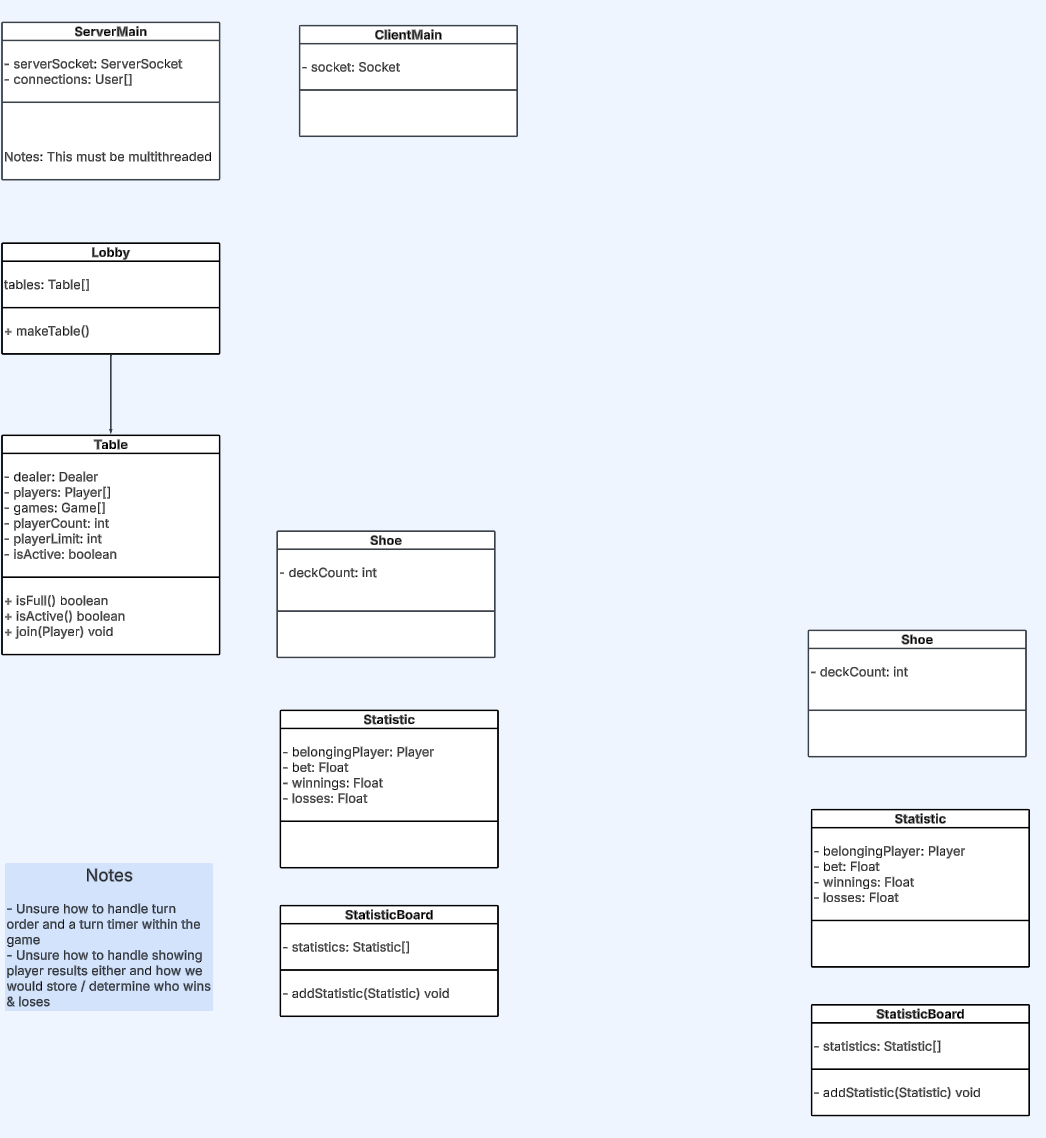
## Player/Dealer Disconnect



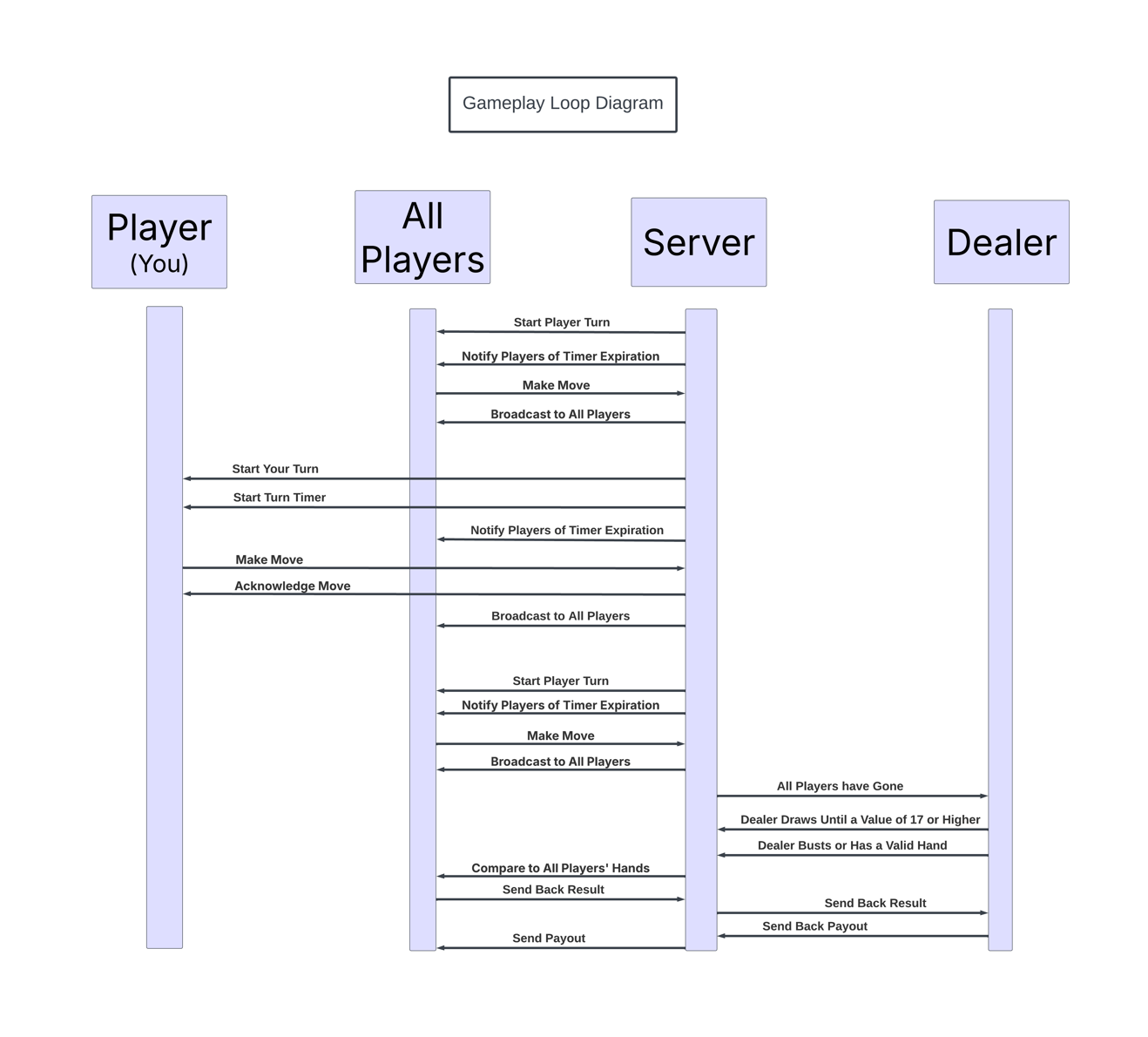
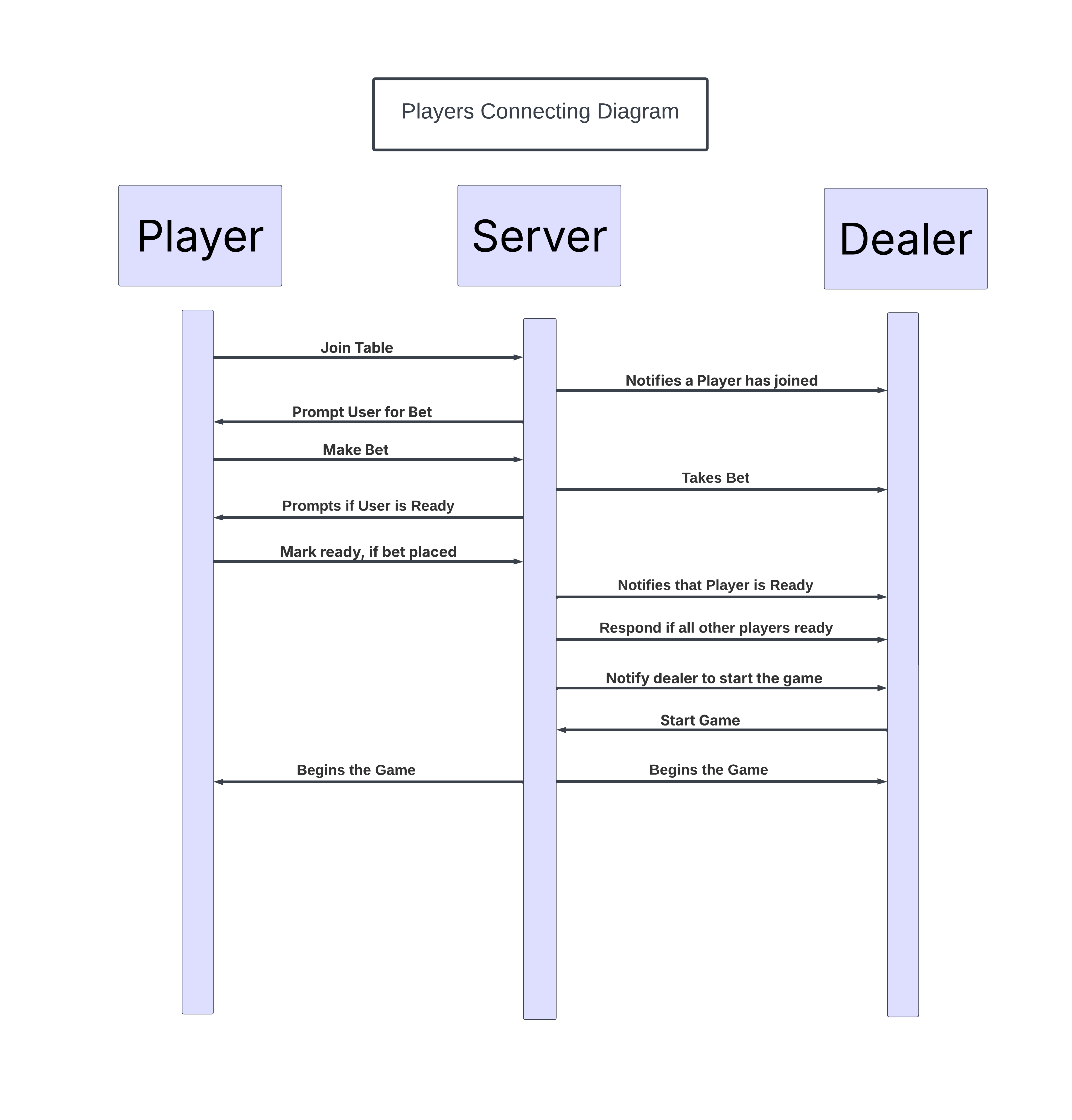
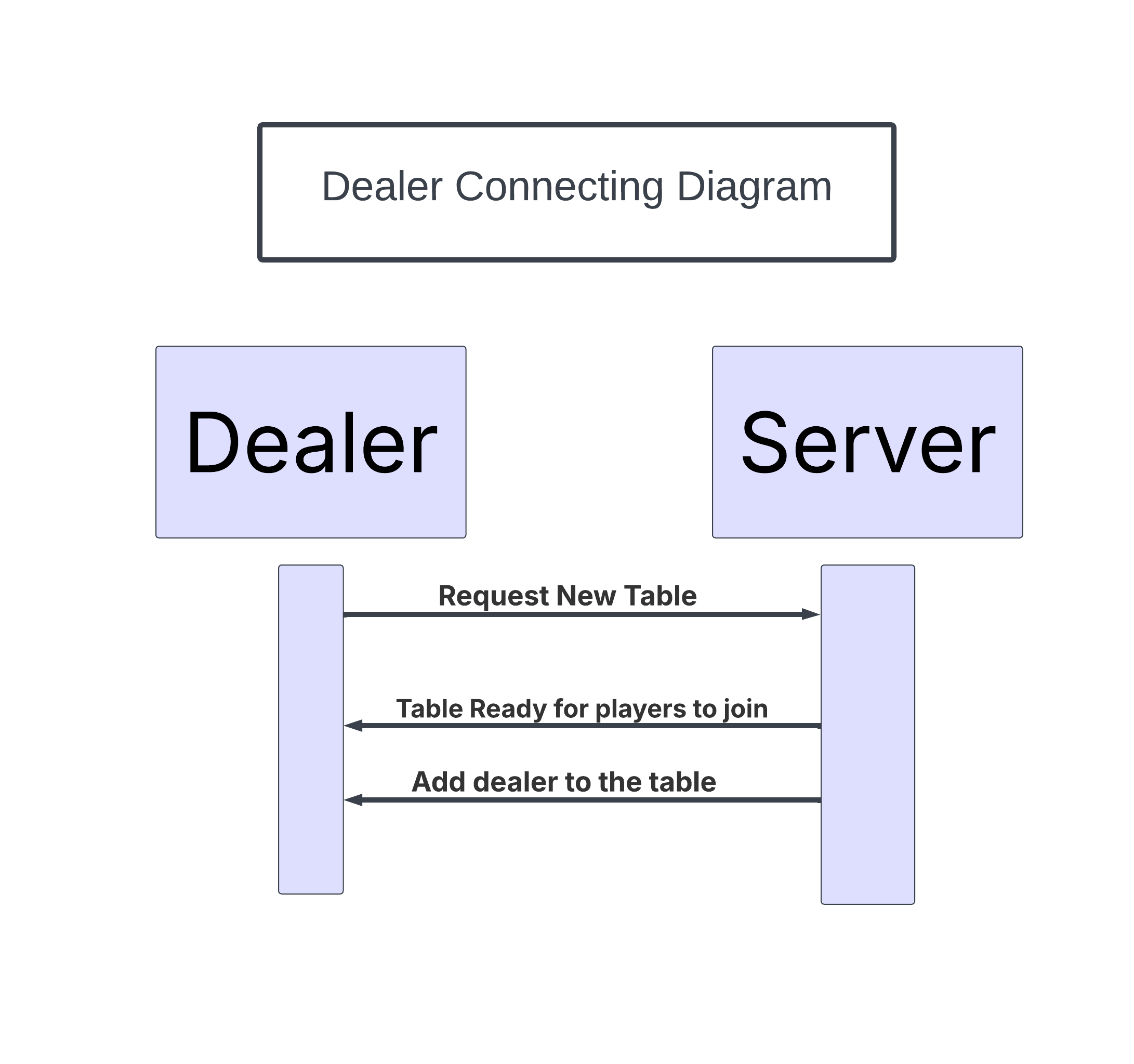
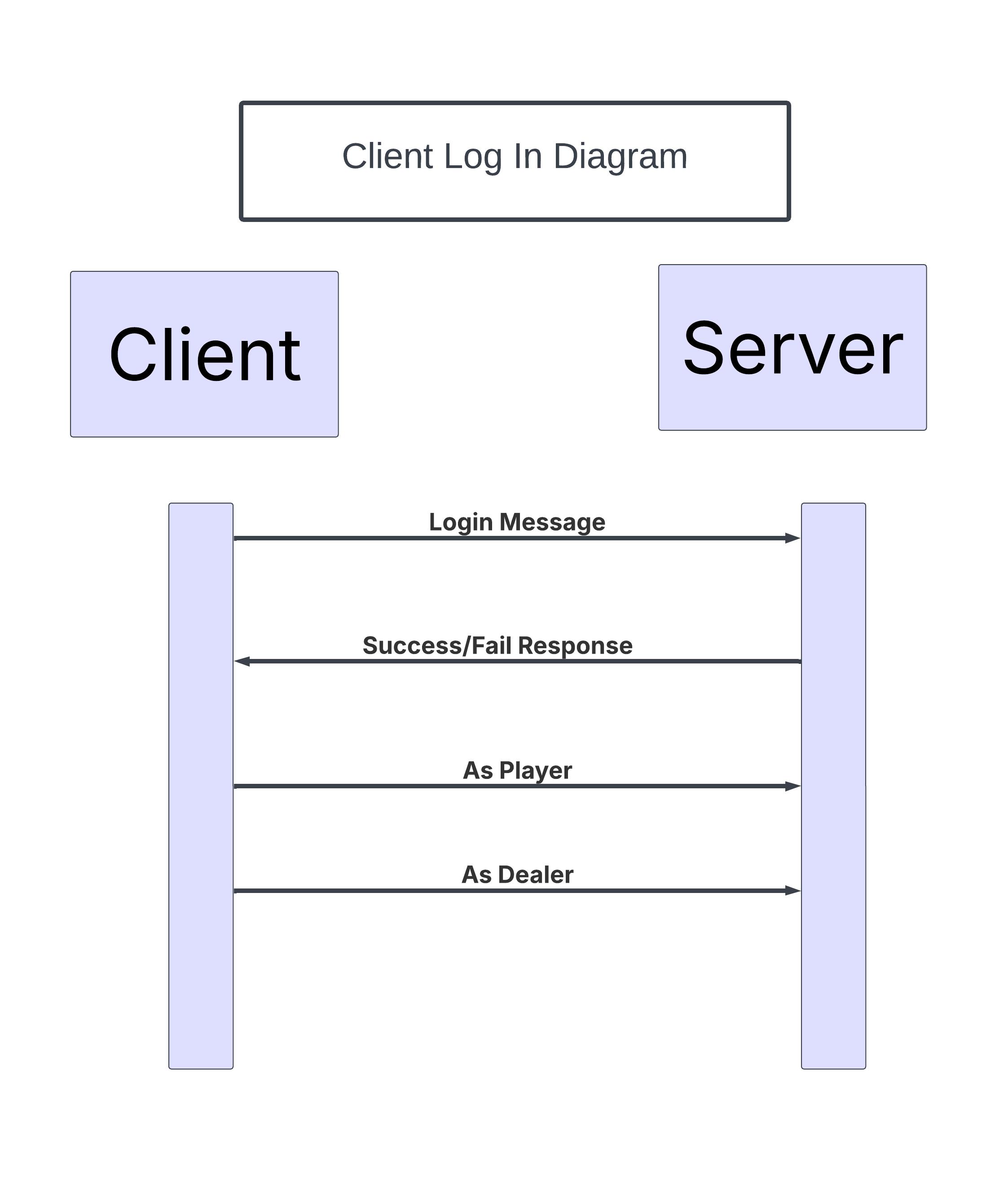
# UML Class Diagram(s)

A computer screen shot of a computer

AI-generated content may be incorrect.

­­

# Sequence Diagrams



# GitHub Repo Link

## GitHub Repo Link: <https://github.com/CSUEB-Black-Jack-Group-5/CS401-BlackJack.git>