

# *Software Requirements Specification*

## Revision History

Date	Revision	Description	Author
09/22/2025	1.0	Purpose	Rahul Suthar
<u>9/23/2025</u>	2.0	Started on Overall Description	Blake Messer
<u>9/23/2025</u>	2.1	Started Specific requirements	Rahul Suthar
<u>9/23/2025</u>	3.0	External requirements	Samuel Millet
<u>9/24/2025</u>	3.1	Internal requirements	Samuel Millet
<u>9/24/2025</u>	4.0	Started Non-Functional Requirements (security)	Blake Messer
<u>9/24/2025</u>	4.1	Non-Functional Requirements (Environmental)	Blake Messer
<u>9.25/2025</u>	4.2	Non-Functional Requirements (Performance)	Rahul Suthar
<u>9/29/2025</u>	5.0	Started Use Case Specifications (1-3)	Rahul Suthar
<u>9/29/2025</u>	5.1	Use Case Specifications (3-4)	Blake Messer
<u>9/29/2025</u>	5.2	Use Case Specifications(5)	Samuel Millet
<u>9/29/2025</u>	5.3	Use Case Specifications(6)	Rahul Suthar
<u>9/29/2025</u>	6.0	UML Case Diagrams (1-2)	Rahul Suthar
<u>9/30/2025</u>	6.1	UML Case Diagrams (3-5)	Blake Messer
<u>9/30/2025</u>	6.2	UML Case Diagrams (6)	Samuel Millet
<u>10/1/2025</u>	7.0	Class Diagram (1-2)	Samuel Millet
<u>10/1/2025</u>	7.1	Class Diagram (3-4)	Rahul Suthar
<u>10/1/2025</u>	7.2	Class Diagram (5-6)	Blake Messer
<u>10/1/2025</u>	8.0	Sequence Diagram (1-3)	Samuel Millet
<u>10/1/2025</u>	8.1	Sequence Diagram (4)	Rahul Suthar
<u>10/1/2025</u>	8.2	Sequence Diagram (5-6)	Blake Messer

# Table of Contents

<b>1. Purpose</b>	<b>4</b>
1.1. Scope	4
1.2. Definitions, Acronyms, Abbreviations	4
1.3. References	5
1.4. Overview	5
<b>2. Overall Description</b>	<b>6</b>
2.1. Product Perspective	6
2.2. Product Architecture	6
2.3. Product Functionality/Features	6
2.4. Constraints	6
2.5. Assumptions and Dependencies	6
<b>3. Specific Requirements</b>	<b>7</b>
3.1. Functional Requirements	7
3.1.1. Common Requirements:	7
3.1.2. Payouts and Betting Module Requirements:	7
3.1.4. Network Module Requirements:	8
3.2. External Interface Requirements	9
3.3. Internal Interface Requirements	9
<b>4. Non-Functional Requirements</b>	<b>11</b>
4.1. Security and Privacy Requirements	11
4.2. Environmental Requirements	11
4.3. Performance Requirements	11
<b>5. Use Case Specification</b>	<b>12</b>
<b>6. UML Use Case Diagram</b>	<b>18</b>
<b>7. Class Diagrams</b>	<b>21</b>
<b>8. Sequence Diagrams</b>	<b>24</b>

# **1. Purpose**

This document outlines the requirements for a Blackjack Application

## **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 cards a player has during a round
- 1.2.2 Table: A game room consisting of 1-7 players
- 1.2.3 Dealer: The person who deals cards to active players
- 1.2.4 BlackJack: A hand consisting of 2 cards that add up to 21
- 1.2.5 Table Limit: The minimum and maximum number of people at a single table
- 1.2.6 Hit: When a player requests an additional card from the dealer.
- 1.2.7 Stand: When a player decides not to take any more cards and ends their turn.
- 1.2.8 Bust: When a player's hand value exceeds 21, resulting in an automatic loss.
- 1.2.9 Card: A single playing card used in the game, with a rank and suit.
- 1.2.10 Suit: One of the four categories of cards (Hearts, Diamonds, Clubs, Spades).
- 1.2.11 Numbered Card: Of a card, numbered 2-10
- 1.2.12 Face Card: Of a card, of rank Jack, Queen, King
- 1.2.13 Ace card: A special card valued at either 1 or 11, depending on the hand.
- 1.2.14 Deck: A standard set of 52 playing cards, consisting of 13 ranks in 4 suits.
- 1.2.15 Push: A tie between the player and dealer where the bet is returned
- 1.2.16 Split: When a player has two cards of the same rank and separates them into two hands.
- 1.2.17 Double Down: When a player doubles their bet and receives only one additional card.
- 1.2.18 TCP/IP: Transmission Control Protocol/Internet Protocol, a set of networking rules enabling communication over the internet.
- 1.2.19 GUI: Graphical User Interface, a visual way for users to interact with the system.
- 1.2.20 RNG: Random Number Generator

### **1.3. References**

- 1.3.1 Use Case Specification Document: Refer to Use Case Specification.docx
- 1.3.2 Github Repository: <https://github.com/Rahul4301/Blackjack/>
- 1.3.3 UML Use Case Diagrams Document
- 1.3.4 Class Diagrams Document
- 1.3.5 Sequence Diagrams Document

### **1.4. Overview**

The multiplayer blackjack system is a Java-based application featuring a graphical user interface (GUI).

It allows players to join virtual tables, place bets, and play according to standard blackjack rules.

The system runs over TCP/IP to provide real-time gameplay, secure fund transactions, fair play, and protection against cheating.

## 2. Overall Description

### 2.1. Product Perspective

The multiplayer blackjack system is a client-server application. The server is the main authority, managing all game state, player accounts, and logic.

### 2.2. Product Architecture

**2.2.1 Client Communication Module:** Manages network connections and messaging with all client applications

**2.2.2 Game Logic Module:** Enforces the rules of blackjack, manages game state for all active tables, and handles dealer's actions.

**2.2.3 Player and Session Management Module:** Handles user authentication, account balances, and active player sessions.

### 2.3. Product Functionality/Features

2.3.1 User account login and management of current session.

2.3.2 Player account management; like viewing account balances, adding or removing funds.

2.3.3 Concurrent multiplayer tables.

2.3.4 A game lobby that shows current game tables and if they are open/available.

2.3.5 Game rules are enforced.

2.3.6 There are real-time updates to all players at a table.

### 2.4. Constraints

2.4.1 Minimum JDK Version

2.4.2 No databases, frameworks, or external libraries unless explicitly approved

2.4.3 Client must provide a graphical user interface (GUI)

### 2.5. Assumptions and Dependencies

2.5.1 Assume that players understand the rules and how to play blackjack.

2.5.2 Assume that system depends on a stable network connection.

2.5.3 Assume that players have a stable internet connection.

2.5.4 Players contain enough funds to engage in the game.

2.5.5 Assume that each game table works within the proper player limit.

2.5.6 Assume that each player has a unique username and unique password for their account.

2.5.7 Each player should make a decision in a certain amount of time or else they will be kicked or booted from the game.

2.5.8 The game should start once at least one player minimum joins the game table.

2.5.9 Once confirmed any bet made is final and cannot be rescinded.

## **3. Specific Requirements**

### **3.1. Functional Requirements**

#### **3.1.1. Common Requirements:**

- 3.1.1.1 Users should be allowed to create an account with a unique username and password, between 5-20 characters
- 3.1.1.2 Users should be able to log out of the system at any time.
- 3.1.1.3 Users should be allowed to register for a new account with a unique username and password.
- 3.1.1.4 Users should be able to recover or reset their password in case of loss.
- 3.1.1.5 The system shall enforce account states (New, Active, Blocked, Banned) for user access control.

#### **3.1.2. Payouts and Betting Module Requirements:**

- 3.1.2.1 The system must allow players to wager money from their account funds when they are in an active game.
- 3.1.2.2 The system must add game winnings from each game to the winning player's funds.
- 3.1.2.3 The system must deduct funds from player accounts when placing a bet if sufficient funds are available
- 3.1.2.4 Players must place a minimum bet of \$1 per hand.
- 3.1.2.5 Players must place bets using integer values only; decimal values are not allowed.
- 3.1.2.6 The system must prevent bets exceeding the player's available balance.
- 3.1.2.7 The system shall log all bets and payouts for audit and reporting.
- 3.1.2.8 1:1 Payout odds, 3:2 payout for a blackjack (21)

#### **3.1.3. Game Logic Module:**

- 3.1.3.1 The system will ensure fair play by utilizing true shuffle algorithms for the decks of cards
- 3.1.3.2 The system will check win and lose conditions after every turn, recognizing if either is met.
- 3.1.3.3 The system will be able to handle checking game states for multiple tables simultaneously
- 3.1.3.4 The system must track and update the game's current state during transitions between betting, card dealing, player turns, and round end.

- 3.1.3.5 A time limit of 30 seconds will be implemented for each player's turn to ensure smooth gameplay.
- 3.1.3.6 If a player fails to act within the time limit, the system will automatically stand for them.
- 3.1.3.7 The system must compare the dealer's final hand against each player's hand to determine winners.
- 3.1.3.8 The system will remove a table from the list of available tables once there is no dealer present.
- 3.1.3.9 The system must handle game restarts and prepare the table for a new round.

#### **3.1.4. Network Module Requirements:**

- 3.1.1.1 The system shall allow users to connect to the game server securely.
- 3.1.1.2 The system shall update game data in real time for all players at a table.
- 3.1.1.3 The system shall allow players to disconnect at any time; dealers must wait until the end of the current game before disconnecting.
- 3.1.1.4 If a player loses connection, they will be dropped from the game and all their current bets will be forfeited.
- 3.1.1.5 The system shall handle reconnection attempts and restore sessions if possible.

#### **3.1.5. Player Module Requirements**

- 3.1.5.1 Players are allowed to deposit and withdraw money from their account.
- 3.1.5.2 Player winnings shall be credited automatically, and withdrawals may be requested at any time.
- 3.1.5.3 The player account will include session management and current balance.
- 3.1.5.4 Players can only join a table if it does not exceed the maximum of 6 players and if they have sufficient funds.
- 3.1.5.5 Players must wait in the lobby until invited to join a table by the dealer.
- 3.1.5.6 Players must be able to view their game history, including past bets and results.
- 3.1.5.7 The system must enforce one active session per player account.

#### **3.1.6. Dealer Module Requirements**

- 3.1.6.1 The dealer must play only after all players have completed their actions.
- 3.1.6.2 The dealer must hand out two cards to each player and themselves initially.
- 3.1.6.3 Dealers may not leave a table while there is an active game in progress.
- 3.1.6.4 Dealers must manage player seating and invitations from the lobby.



### **3.1.7. Table Module Requirements**

- 3.1.7.1 In order for a game to start, there must be at least one player and exactly one dealer, with a maximum of seven players.
- 3.1.7.2 A table will remain active as long as there is exactly one dealer present.
- 3.1.7.3 The system shall ensure that no player is seated at more than one table simultaneously.
- 3.1.7.4 The system must provide unique identifiers for each table to distinguish active sessions.

## **3.2. External Interface Requirements**

- 3.2.1. The system must provide a user-friendly graphical interface for both players and dealers.
- 3.2.2. The GUI must display real-time game updates, including player actions, card draws, bets, payouts, results, and actions like standing, hitting, busting, splitting, and doubling down.
- 3.2.3. Separate dashboards must be provided for players and dealers.
- 3.2.4. The GUI must be visually engaging and immersive to enhance the player's experience.
- 3.2.5. Animations for shuffling, dealing cards, and moving chips must be included to improve immersion.
- 3.2.6. Players must be able to place bets, hit, stand, double down, split, or leave the game.
- 3.2.7. Before joining a game, the system must present lobbies showing available tables.

## **3.3. Internal Interface Requirements**

- 1.3.1. The system must authenticate users by verifying their username and password during login.
- 1.3.2. If a user times out, quits, or disconnects, all in-game interactions must end immediately.
- 1.3.3. The system must track and display all game status changes in real time, including hand results for both players and dealers.
- 1.3.4. The internal interface must support all valid player actions: hit, stand, double down, split, and leave the game.
- 1.3.5. A Random Number Generator must be used to ensure fair shuffling of the deck.
- 1.3.6. If a player lacks sufficient funds, the system must reject their bet before the game begins.
- 1.3.7. The deck must simulate a "shoe" of three decks combined, with reshuffling triggered halfway through the game.

1.3.8. The GUI must receive and display data from the game engine in real time.

## 4. Non-Functional Requirements

### 4.1. Security and Privacy Requirements

4.1.1 **Anti Cheating Measures:** The system will include anti-cheating measures to prevent malicious intent by using two decks of cards for card counters and reshuffle the deck halfway.

4.1.2 **Secure Authentication:** The player has to log into the system using a valid and unique username and password.

4.1.3 **Controls based on Role:** Players of certain roles will have access to limited functions.

### 4.2. Environmental Requirements

4.2.1 System must be developed using the Java programming language that must be compatible with all platforms including Windows, Linux, and MacOS.

4.2.2 System must have consistent UI design from all screens from the blackjack lobby, game table, and balance inquiry screens.

4.2.3 System must operate over TCP/IP, to ensure that real-time gameplay is achieved with more than one player on multiple different devices over the same network.

4.2.4 System needs to have a stable network connection to ensure real-time gameplay engagement.

### 4.3. Performance Requirements

4.3.1 **Table Players:** System must be able to support up to 6 players at the same time for each game table while also maintaining real-time gameplay to ensure the engagement is continued.

4.3.2 **Lobby Players:** System must be able to support a lobby that consists of multiple tables of 6 coexisting players happening at the same time without any performance complications.

4.3.3 **Disconnecting/Inactive Players:** In an event where a player loses connection to the system, quits the game, or their connection is timed out due to being inactive, then the game will remove them from the table to ensure a consistent experience without any interruption.

## 5. Use Case Specification

### Use Case 1: Account Management

Use Case ID: UC01

Use Case Name: Account Management

Relevant Requirements: 3.1.1.1–3.1.1.5, 4.1.2

Primary Actor: Player

Preconditions:

None

Postconditions:

- Player account is created, secured, or session is activated.
- Account state updated (New, Active, etc.).

Basic Flow:

- Player selects “Create Account,” “Login,” or “Reset Password.”
- If creating account: player enters username & password.
- System validates uniqueness, format, and password strength.
- If logging in: system verifies credentials and activates session.
- If resetting password: system enforces new secure password.

Extensions / Alternate Flow:

- Username already exists → system requests another.
- Invalid credentials → deny login.

Exceptions:

- If system crash occurs, account update not applied.

Related Use Cases: UC02: Fund Management

## Use Case 2: Fund Management

Use Case ID:

Use Case Name: Fund Management (Deposit/Withdraw/Bet)

Relevant Requirements: 3.1.2.1–3.1.2.7, 3.1.5.1–3.1.5.3

Primary Actor: Player

Preconditions:

- Player must be logged in.

Postconditions:

- Account balance updated.
- Bets locked for the current round.

Basic Flow:

- Player selects “Deposit,” “Withdraw,” or “Place Bet.”
- If deposit: player enters amount → system validates and updates balance.
- If withdraw: system checks sufficient balance → processes withdrawal.
- If bet: player enters amount  $\geq \$1$ , integer only.
- System validates funds and table rules, deducts balance, and records bet.

Extensions / Alternate Flow:

- Insufficient funds → reject withdrawal or bet.
- Invalid amount (decimal or  $< \$1$ ) → reject request.

Exceptions:

- If connection is lost mid-transaction → cancel operation.

Related Use Cases: UC03: Join Game Table, UC04: Play Round

### Use Case 3: Join Game Table

Use Case ID: UC03

Use Case Name: Join Game Table

Relevant Requirements: 3.1.5.4–3.1.5.5, 3.1.7.1–3.1.7.4

Primary Actor: Player, Dealer

Preconditions:

- Player is logged in and has sufficient funds.
- Dealer is present at table.

Postconditions:

- Player is seated at selected table.
- Table state updated to include new participant.

Basic Flow:

- Player selects a table from the lobby.
- System validates seat availability (max 6 players, one dealer).
- Dealer approves or invites player.
- Player joins table and waits for round start.

Extensions / Alternate Flow:

- Table is full → reject request.
- No dealer present → table unavailable.

Exceptions:

- If player disconnects mid-join → request canceled.

Related Use Cases: UC04: Play Round

## Use Case 4: Play Round

Use Case ID: UC04

Use Case Name: Play Round (Player + Dealer Actions)

Relevant Requirements: 3.1.3.1–3.1.3.9, 3.1.6.1–3.1.6.4

Primary Actor: Player, Dealer (System-controlled)

Preconditions:

- Player has placed a valid bet.
- Cards are dealt.

Postconditions:

- Dealer hand finalized.
- Round outcome ready for calculation.

Basic Flow:

- Dealer deals two cards to each player and self.
- Each player takes turn choosing actions: Hit, Stand, Double Down, Split.
- System updates hand after each action.
- If  $> 21$  → player busts immediately.
- Once all players finish → dealer reveals hand and plays (hit  $\geq 17$  or bust).

Extensions / Alternate Flow:

- Player timeout (30s) → system auto-stands.
- Player disconnects → bet forfeited.

Exceptions:

- If dealer disconnects → table closes automatically.

Related Use Cases: UC05: Determine Outcome and Payout

## Use Case 5: Determine Outcome and Payout

Use Case ID: UC05

Use Case Name: Determine Outcome and Payout

Relevant Requirements: 3.1.2.7–3.1.2.8, 3.1.3.7

Primary Actor: System

Preconditions:

- Dealer and players have final hands.

Postconditions:

- Player balances updated.
- Game outcome recorded in history log.

Basic Flow:

- System compares dealer vs. each player's hand.
- If dealer busts → all active players win.
- If player > dealer → 1:1 payout.
- If player has Blackjack → payout 3:2.
- If tie → push, return bet.
- Otherwise, dealer wins → player loses bet.

Extensions / Alternate Flow:

- None.

Exceptions:

- If transaction logging fails → rollback payout changes.

Related Use Cases: UC04: Play Round, UC02: Fund Management



## **Use Case 6: Shuffle and Manage Deck**

Use Case ID: UC06

Use Case Name: Shuffle and Manage Deck

Relevant Requirements: 3.1.3.1, 3.1.3.4, 3.1.3.9, 1.3.5

Primary Actor: System (Dealer role)

Preconditions:

- New round is about to begin OR shuffle threshold met.

Postconditions:

- Fairly shuffled deck prepared for next round.

Basic Flow:

- System initializes multi-deck shoe.
- System applies RNG shuffle.
- System updates deck state.
- If halfway exhausted → reshuffle automatically.

Extensions / Alternate Flow:

- Deck empty → reinitialize and reshuffle.

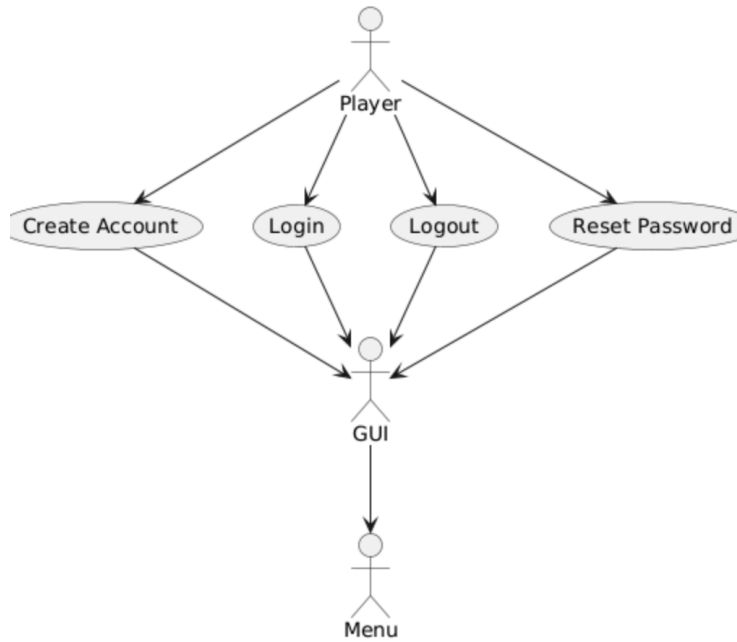
Exceptions:

- If RNG fails, reshuffle not applied (game paused).

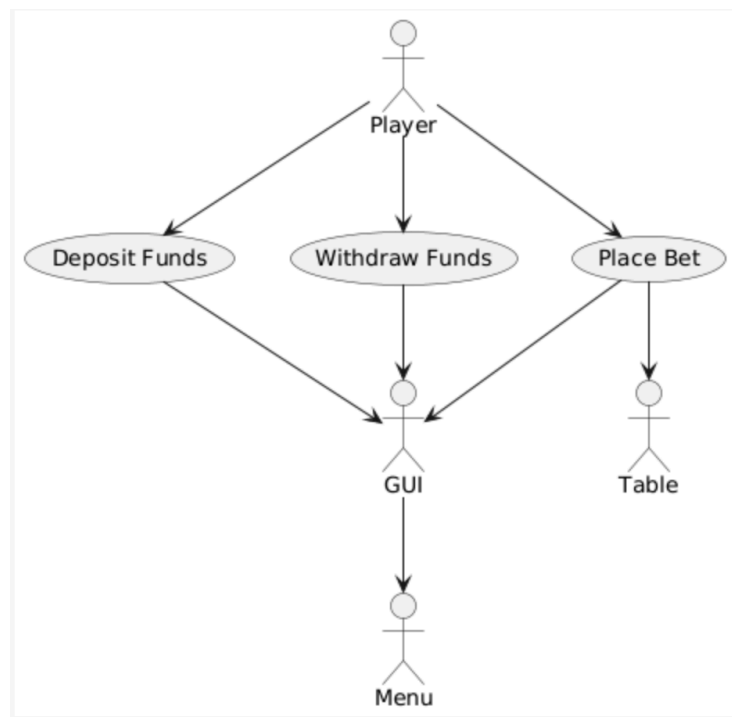
Related Use Cases: UC04: Play Round

## 6. UML Use Case Diagram

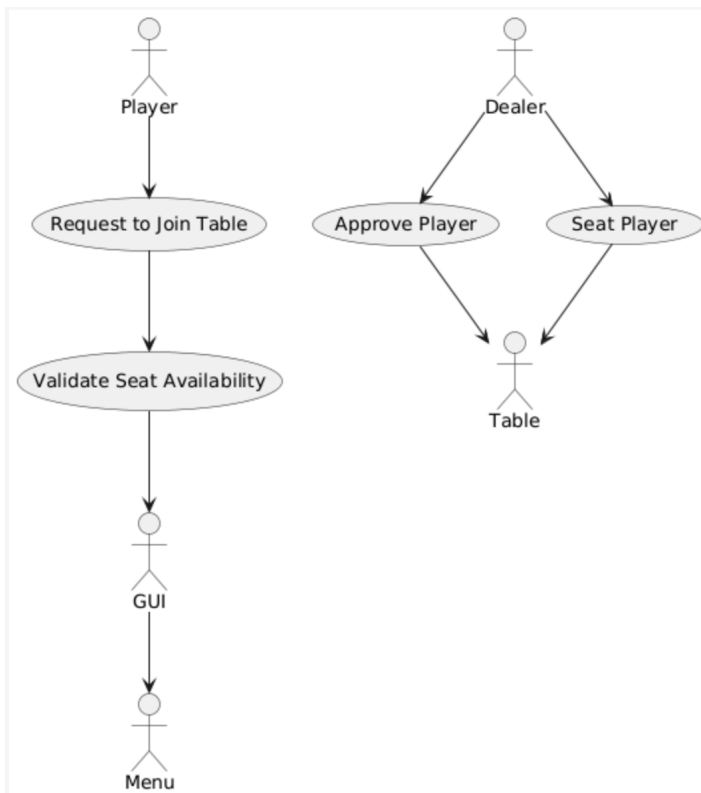
### Use Case 1: Account Management



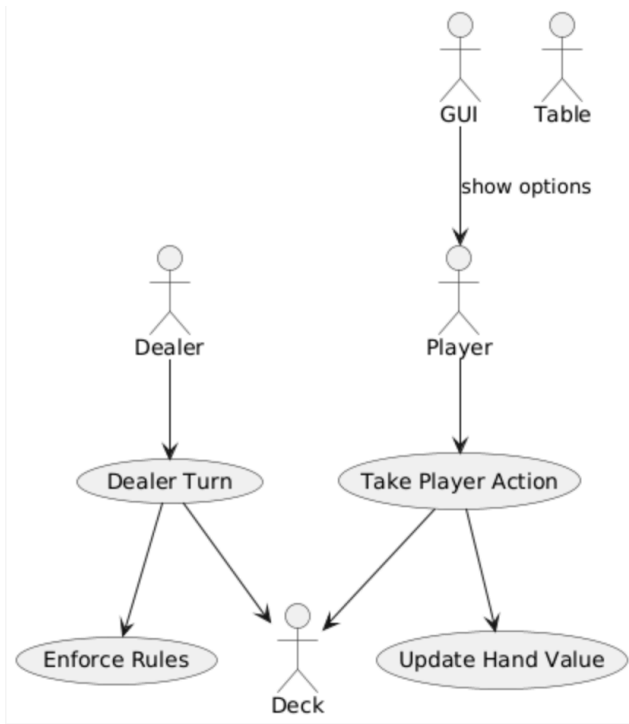
### Use Case 2: Fund Management



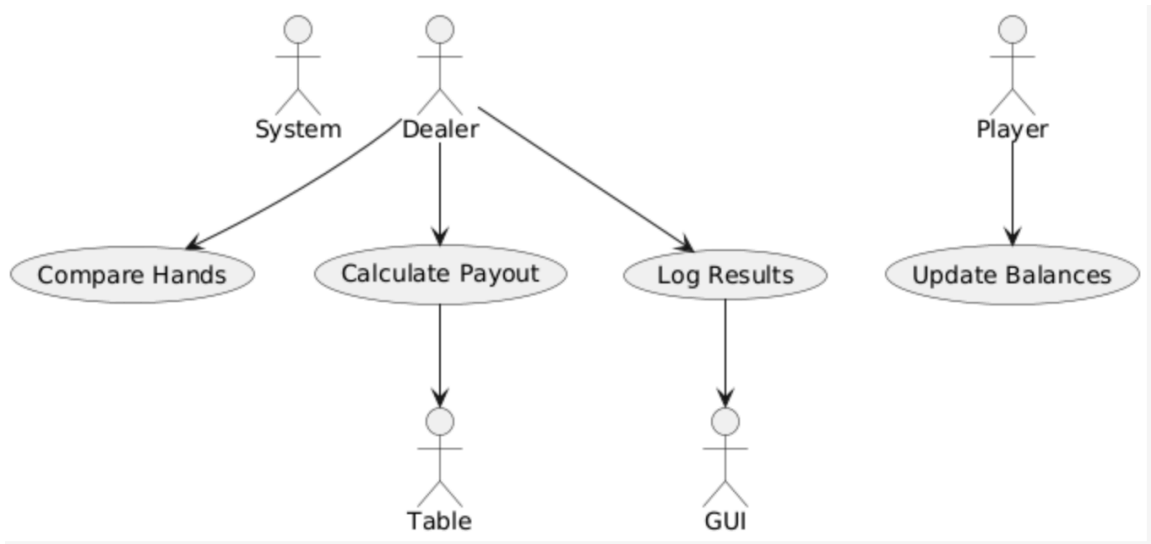
### Use Case 3: Join Game Table



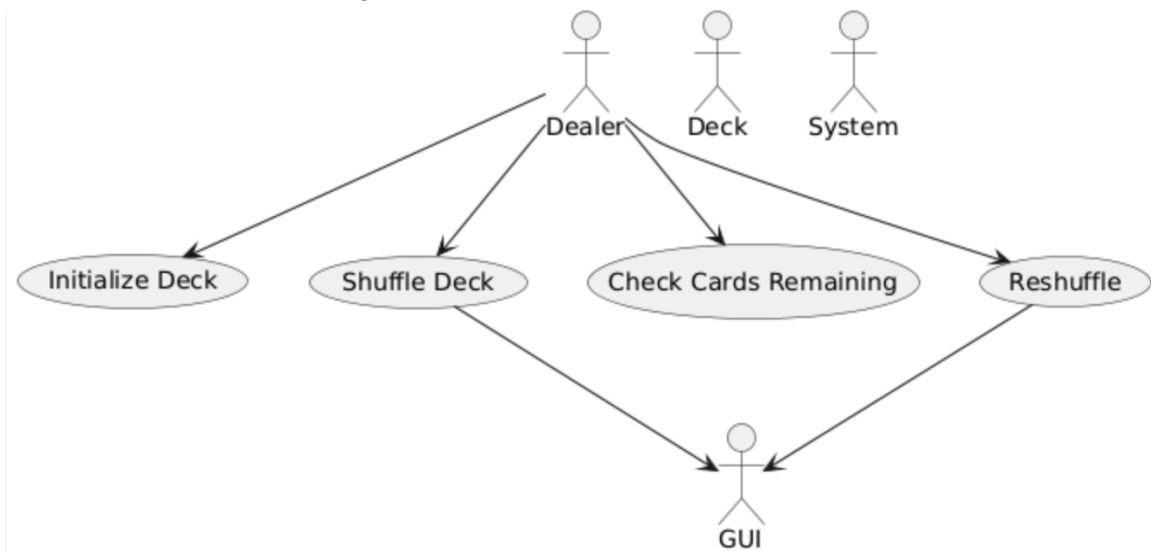
### Use Case 4: Play Round



### Use Case 5: Determine Outcome and Payout

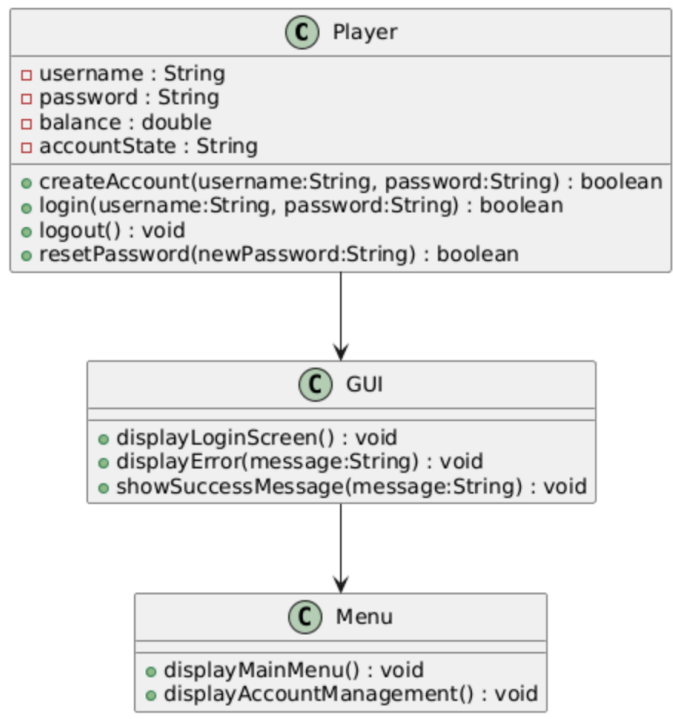


### Use Case 6: Shuffle and Manage Deck

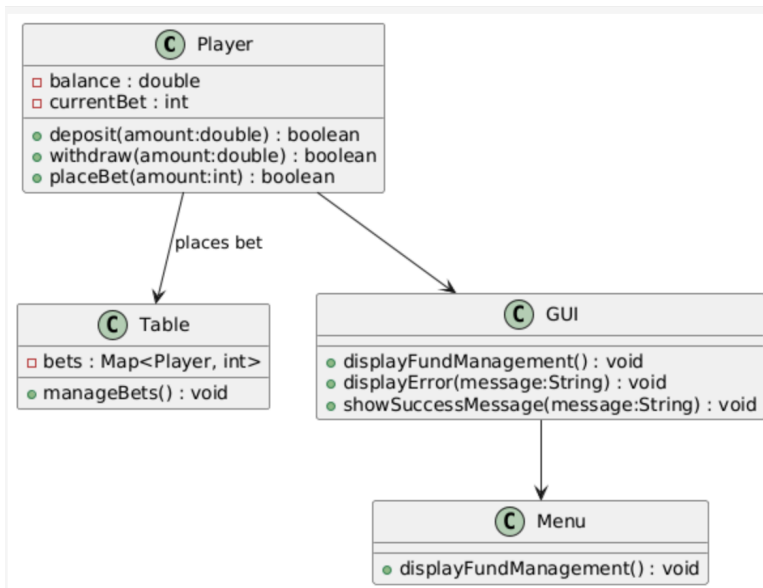


## 7. Class Diagrams

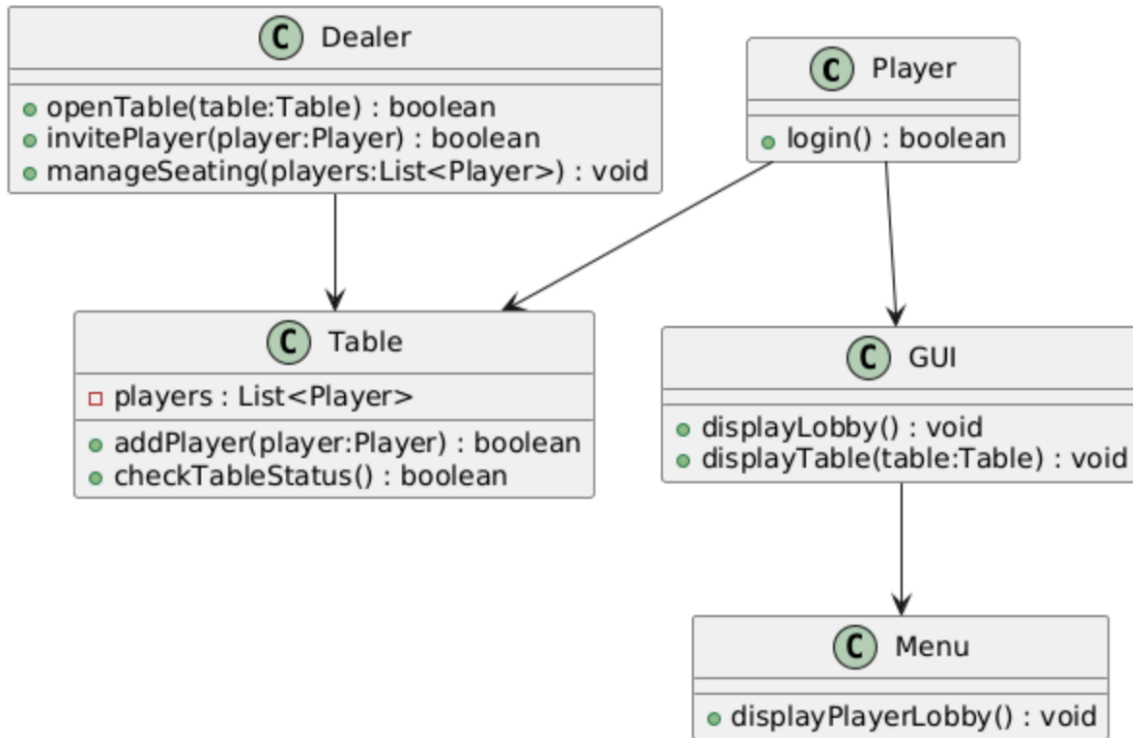
### Use Case 1: Account Management



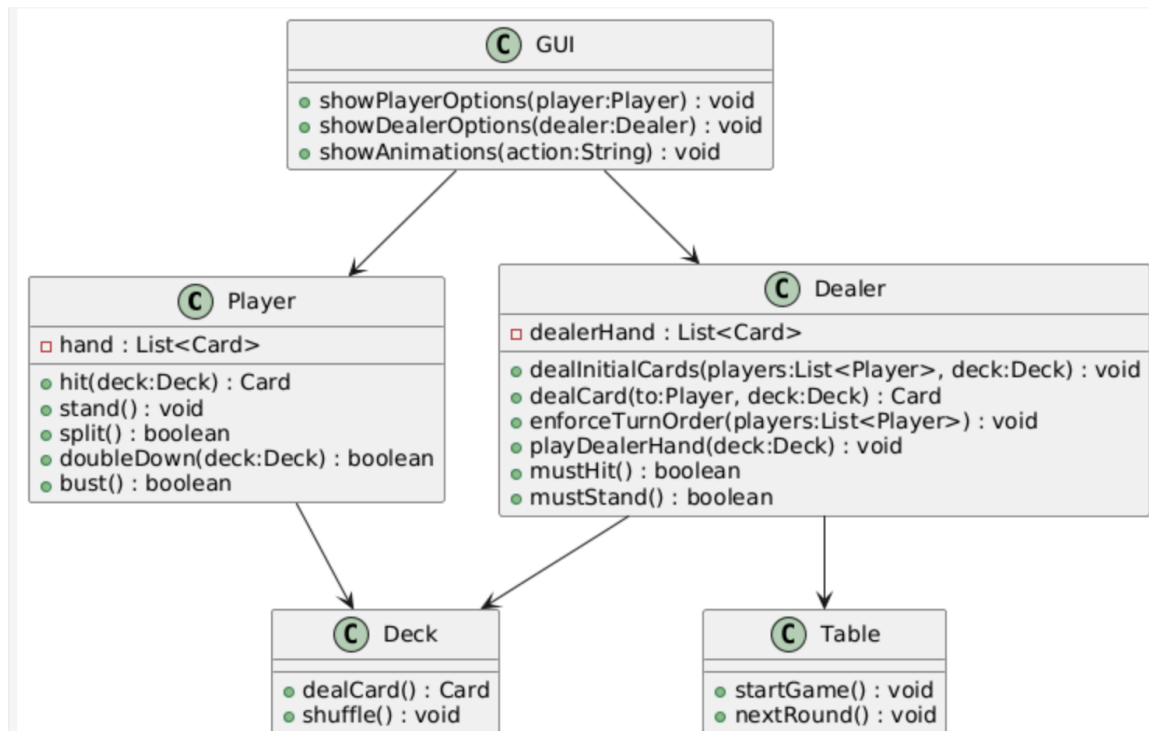
### Use Case 2: Fund Management



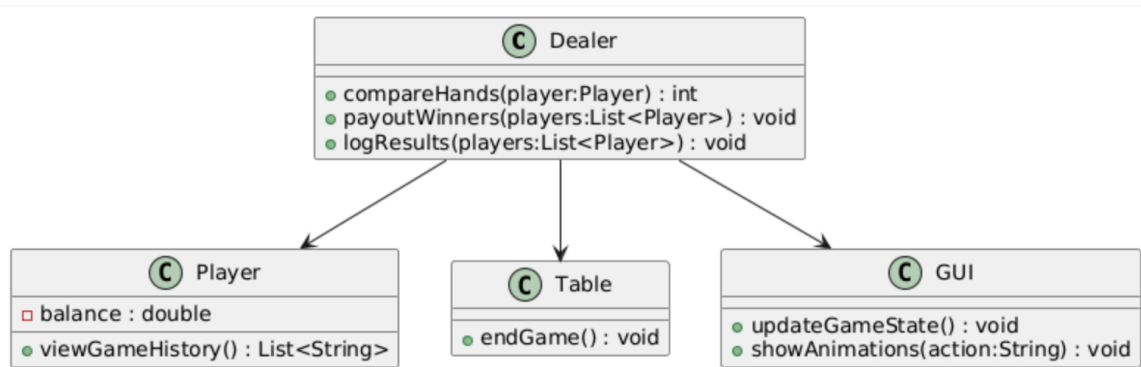
### Use Case 3: Join Game Table



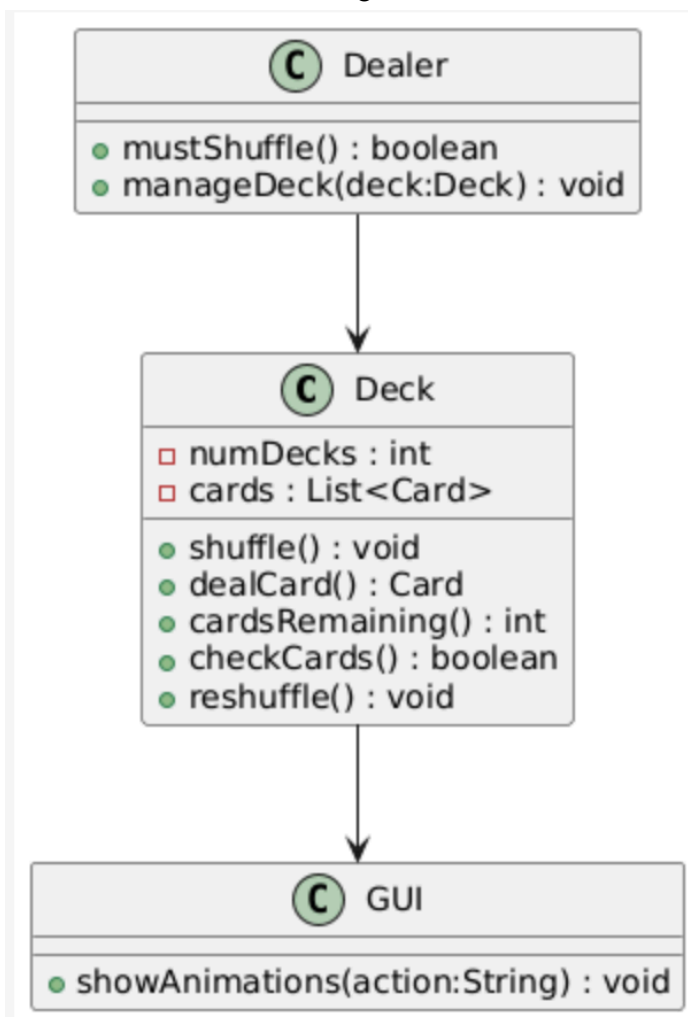
### Use Case 4: Play Round



#### Use Case 5: Determine Outcome and Payout

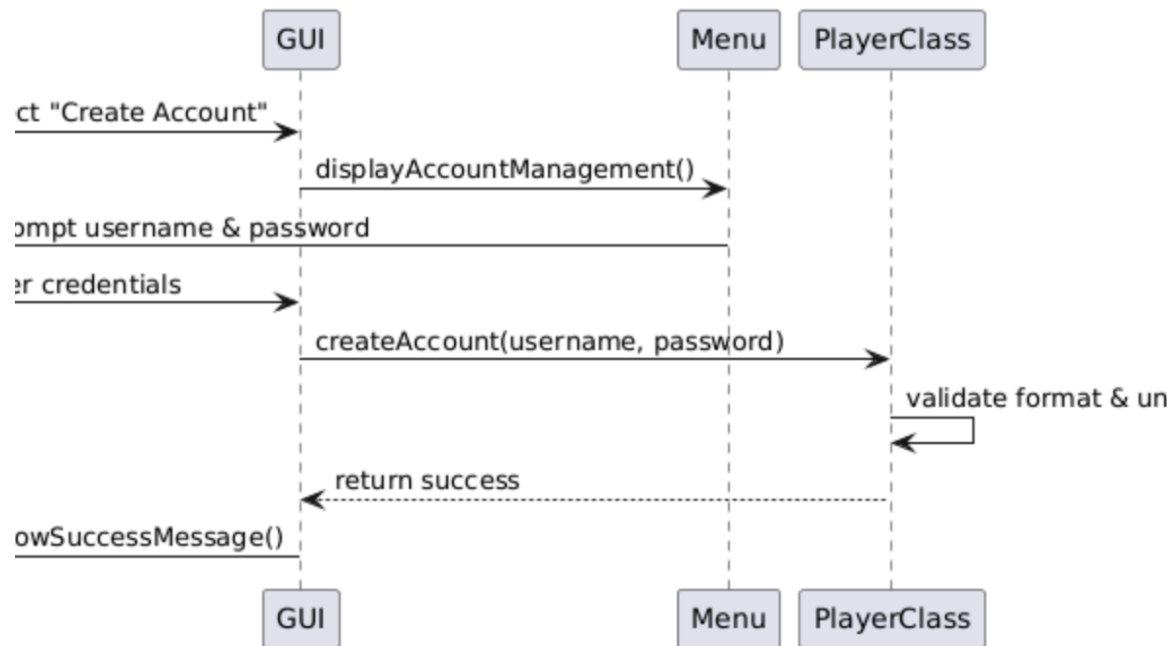


#### Use Case 6: Shuffle and Manage Deck



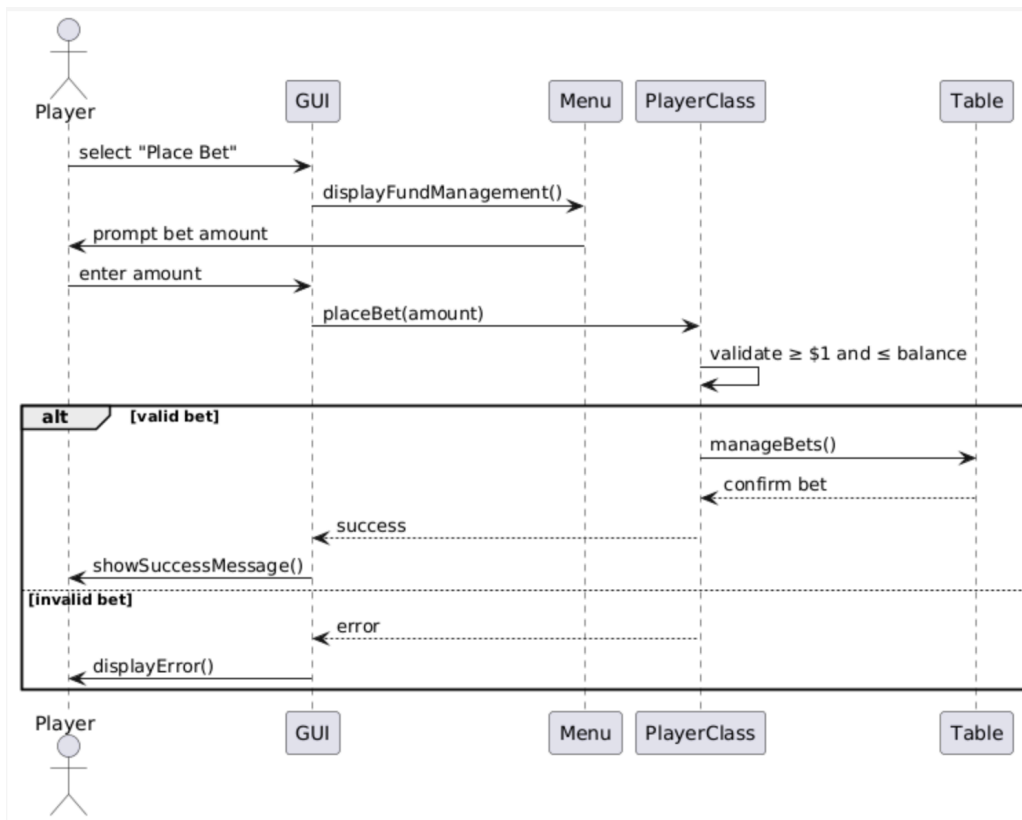
## 8. Sequence Diagrams

Use Case 1: Account Management

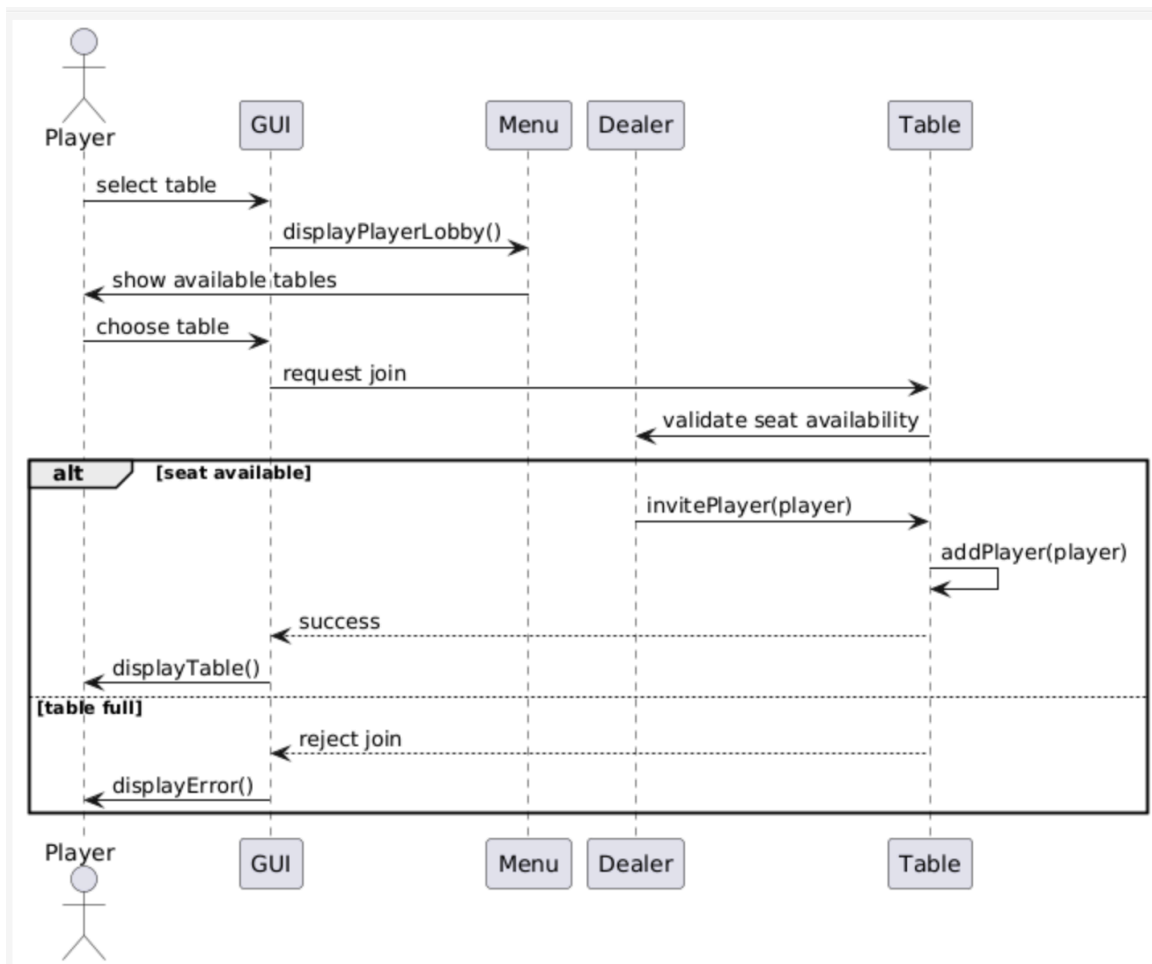




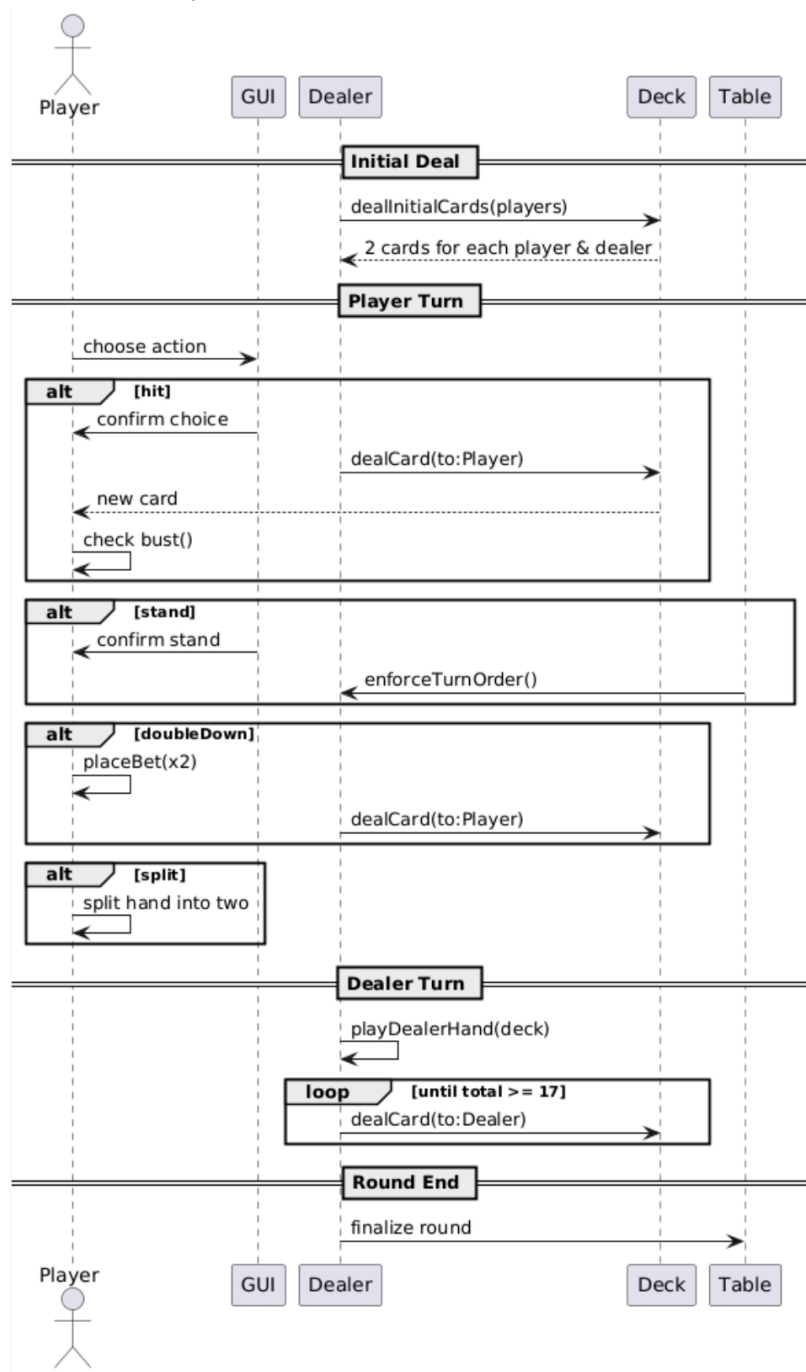
## Use Case 2: Fund Management



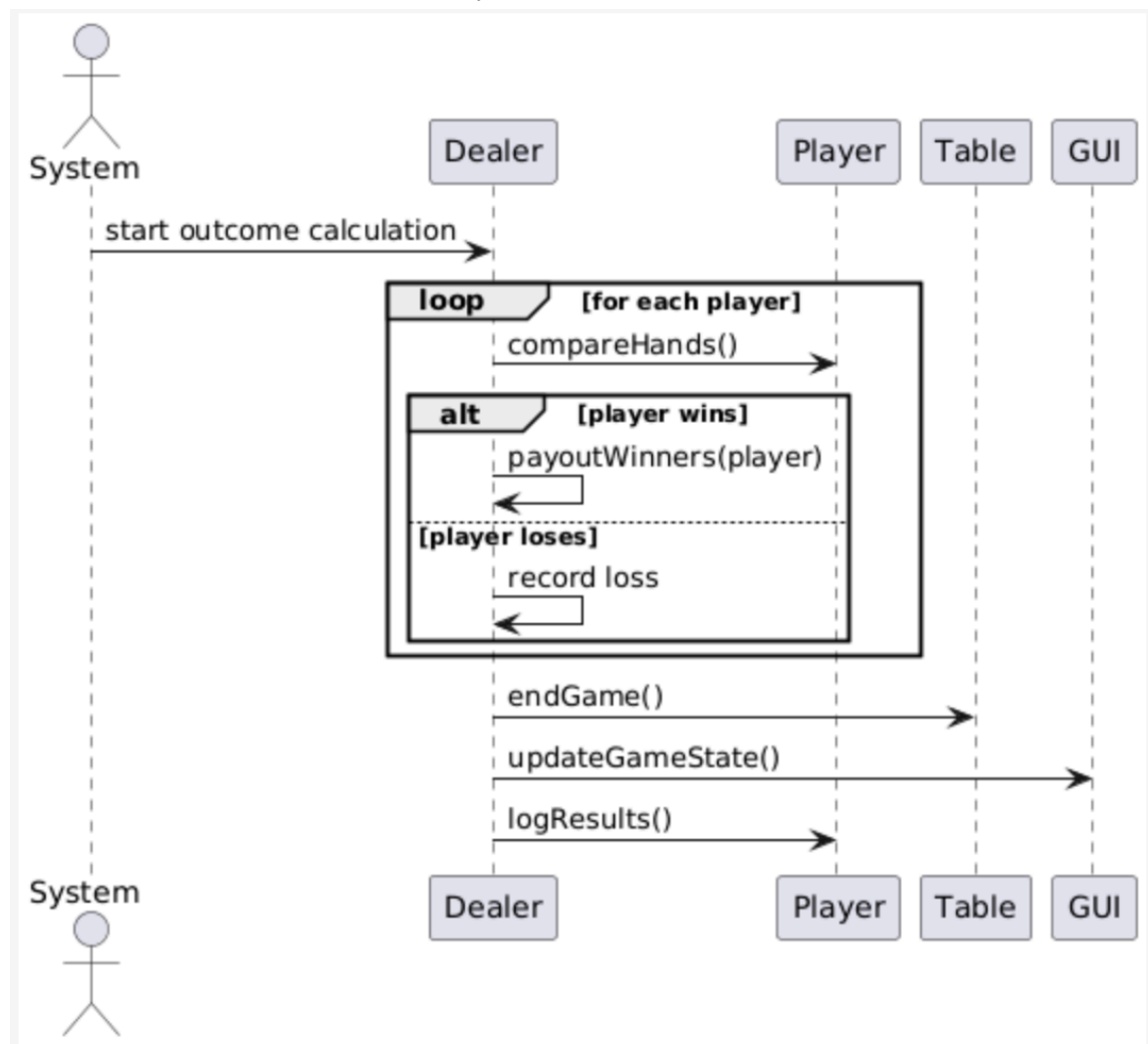
### Use Case 3: Join Game Table



## Use Case 4: Play Round



Use Case 5: Determine Outcome and Payout



# Use Case 6: Shuffle and Manage Deck

