**Multi-Player BlackJack**

Software Requirements Specification

Revision History

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 09/28/2024 | 0.1 | Section 1.1 -1.2 | Santiago Ruiz Arias |
| 09/28/2024 | 0.2 | Class Diagram Base Layout | Nikolas Alvarado |
| 9/30/2024 | 0.3 | Section 3.1.1, 4.1, 4.2 | Santiago Ruiz Arias |
| 9/30/2024 | 0.4 | Use Case - UML Case Diagram | Carolina Barbecho |
| 9/30/24 | 0.5 | Updated Classes for Players | Nikolas Alvarado |
| 9/30/24 | 0.6 | 1.4 , updated 1.2 Definitions | Carolina Barbecho |
| 9/30/24 | 0.7 | Updated Sections 2.1, 2.2, 2.3, 2.4, 2.5 | Matthew Vincent Carreon |
| 9/30/24 | 0.8 | 3.1.2 Player, 3.1.3 hand, 3.1.4 deck requirement | Carolina Barbecho |
| 10/1/24 | 0.9 | Created Class Diagram and Sequence Diagrams for UC01 - UC06 | Oscar Avendano |
| 10/1/24 | 0.10 | Finished the rest of the 3 section and added login in use case | Brandon Sandoval |
| 10/2/2024 | 0.11 | Split Player Class to encapsulate all users. Created new Hand class to call card methods. Updated Class Diagram to include Hand class and organized basic connections | Nikolas Alvarado |
| 10/02/2024 | 0.12 | Update: Use case, UML case Diagram, User Module Requirement. | Carolina Barbecho |
| 10/02/24 | 0.13 | Updated 1.3.3 Diagram for better clarity and readability | Matthew Vincent Carreon |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**1.** **Purpose 4**

1.1. Scope 4

1.2. Definitions, Acronyms, Abbreviations 4

1.3. References 4

1.4. Overview 4

**2.** **Overall Description 5**

2.1. Product Perspective 5

2.2. Product Architecture 5

2.3. Product Functionality/Features 5

2.4. Constraints 5

2.5. Assumptions and Dependencies 5

**3.** **Specific Requirements 6**

3.1. Functional Requirements 6

3.2. External Interface Requirements 6

3.3. Internal Interface Requirements 7

**4.** **Non-Functional Requirements 8**

4.1. Security and Privacy Requirements 8

4.2. Environmental Requirements 8

4.3. Performance Requirements 8

# Purpose

This document outlines the requirements for the Multi-Player BlackJack (MPBJ) game.

## Scope

This document will catalog the user, system, and hardware requirements for the MPBJ system. It will not, however, document how these requirements will be implemented.

## Definitions, Acronyms, Abbreviations

List any acronyms, terms etc. that need to be defined.

* + 1. Black Jack Rules: <https://bicyclecards.com/how-to-play/blackjack>
    2. Table: a place for 6 players and dealers can play Blackjack. Per table can seat 1-5 players and 1 dealer.
    3. Deck: Set of cards in a specific order
    4. Dealer: Role to deal cards and play the game with other players
    5. Player: A non-dealer to join lobbies, bet on their winnings, and make active decisions in Black-Jack
    6. Hit: For the person to request another card
    7. Stand: For the person to stop requesting cards and stay with their hand
    8. Hand: The set of cards a player has for themselves
    9. Bust: Occurs when a player’s total hand value exceeds 21. This results in an automatic loss for that player, regardless of the dealer’s hand or the outcome of the round.
    10. Shoe: Hold multiple decks of cards.

## References

1.3.1 Use Case Specification Document

**Use Case ID:** UC00

**Use Case Name:** Login

**Relevant Requirements:** join game, withdraw funds, deposit funds,

**Primary Actor:**

User: players/dealer

**Pre-conditions:**

The app is functional and can be used, and there must be credentials already for the player/dealer.

**Post-conditions:**

Users access the system and can have access to withdraw and deposit their funds and join a game.

**Basic Flow or Main Scenario:**

The user opens the system interface.

The user enters their username, password and selects the role as (player/dealer) .

A new menu shows options: withdraw funds, deposit funds , or join a game.

**Extensions or Alternate Flows:**

If the login failed then it tells them to retry.

**Exceptions:**

If the system is unavailable, it will display a message informing the user of the inconvenience.

**Related Use Cases:**

Add funds(UC02)

Deal cards(UC06)

Withdraw Funds(UC03)

**Use Case ID:** UC01

**Use Case Name:** Join Game

**Relevant Requirements:** Users join the game.

**Primary Actor:**

User: players/dealer

**Pre-conditions:**

The app is functional and can be used and the user must have been logged in.

**Post-conditions:**

Users access the system and can use features based on the player/dealer role.

**Basic Flow or Main Scenario:**

The user opens the system interface.

The player/dealer will have access to the black jack game and manage hand UC04 according to their roles.

**Extensions or Alternate Flows:**

If the game is full and other tables are full it will create a new table

**Exceptions:**

If the system is unavailable/full, it will display a message informing the user of the inconvenience.

**Related Use Cases:**

Add funds(UC02)

Deal cards(UC06)

Withdraw Funds(UC03)

Manage Hand(UC04)

**Use Case ID:** UC02

**Use Case Name:** Add funds

**Relevant Requirements:** Users add funds to their account

**Primary Actor:**

User: players/dealer

**Pre-conditions:**

The app is functional and accessible.

**Post-conditions:**

The user’s account balance is updated with the added funds.

**Basic Flow or Main Scenario:**

The user navigates to the “Add Funds” in the system/app.

The user enters the amount they wish to add.

The systems check the input.

If the input is a valid number:

The system updates the user’s account balance.

If the input is invalid:

The system prompts the user to enter a valid numerical amount.

**Extensions or Alternate Flows:**

**Exceptions:**

If the system is unavailable, it will display a message informing the user of the inconvenience.

**Related Use Cases:**

Join Game (UC01)

Withdraw Funds(UC03)

**Use Case ID:** UC03

**Use Case Name:** Withdraw Funds

**Relevant Requirements:**

**Primary Actor:**

User: players/dealer

**Pre-conditions:**

The app is functional and accessible.

**Post-conditions:**

The user’s account balance is updated after the withdrawal.

A confirmation message is displayed to the user of the withdrawal.

**Basic Flow or Main Scenario:**

The user navigates to the “Withdraw Funds: section of the app.

The user enters the amount they wish to withdraw.

The system checks if the amount is valid and if the user has sufficient funds.

The system processes the withdrawal.

The system updates the user’s account balance.

**Extensions or Alternate Flows:**

If the input us invalid or funds are insufficient, the system prompts the user to enter a valid amount.

If the user cancels the transaction, they are returned to the previous screen.

**Exceptions:**

**Related Use Cases:**

Join Game (UC01)

Add Funds (UC02)

**Use Case ID:** UC04

**Use Case Name:** Manage Hand

**Relevant Requirements:** Players can interact with their hand during gameplay.

**Primary Actor:**

Players/dealer

**Pre-conditions:**

The app is functional and accessible.

**Post-conditions:**

The player’s hand is updated based on their actions(hit, stand, split, double down)

**Basic Flow or Main Scenario:**

The players view their current hand, which displays the cards dealt UC006.

The player decides on an action:

**Hit**

The player selects the “Hit” option.

The system deals one additional card to the player’s hand.

If the total exceeds 21, the player is bust.

**Stand**

The player selects the "Stand" option.

The system retains the current hand and prepares for the next player’s turn.

**Split (if applicable)**

The player selects the "Split" option.

The system checks if the player can split their hand.

If valid, the system creates two hands and allows the player to play each hand separately.

**Double Down(if applicable)**

The player selects the "Double Down" option.

The player places an additional bet.

The system deals one more card to the player’s hand.

The player's turn ends after this action.

**Extensions or Alternate Flows:**

If the player attempts an action not allowed based on the current game state, the system displays an error message.

If the game round ends, the system updates player balances accordingly.

**Exceptions:**

If the system has any issue, an error message is displayed.

**Related Use Cases:**

Login (UC00)

Join Game (UC01)

Add Funds (UC02)

Withdraw Funds (UC03)

Leave Game(UC05)

Deal cards (UC06)

**Use Case ID:** UC05

**Use Case Name:** Leave Game

**Relevant Requirements:** Players can leave a game table when they are done playing.

**Primary Actor:**

Player

**Pre-conditions:**

The app is functional and accessible.

**Post-conditions:**

The player is removed from the game table.

The player’s current game state and balance are updated accordingly.

Other players and the dealer are notified of the player’s departure.

**Basic Flow or Main Scenario:**

The player decides to leave the game and selects the “Leave Game” option.

**Extensions or Alternate Flows:**

**Exceptions:**

If the system is unavailable, it will display a message informing the user of the inconvenience.

**Related Use Cases:**

Login (UC00)

Join Game (UC01)

Add Funds (UC02)

Withdraw Funds (UC03)

Manage Hand (UC04)

Leave Game(UC05)

**Use Case ID:** UC06

**Use Case Name:** Deal cards

**Relevant Requirements:** Dealers deal cards to players and themselves at the beginning of each round.

**Primary Actor:**

Dealer

**Pre-conditions:**

The app is functional and accessible.

The dealer joins the game.

The game round has started, and the players have placed their bets.

**Post-conditions:**

The player receives two cards.

**Basic Flow or Main Scenario:**

The dealer verifies that all players have placed their bets.

The dealer shuffles the deck(s).

The dealer begins dealing cards.

Players receive two cards face up.

The dealer’s first card is face up, and the second is face down.

The dealer then deals another card according to Manage hand UC004

**Extensions or Alternate Flows:**

Dealers deal cards to players based on their actions during the game.

The dealer identifies that it is the player’s turn to take action.

The dealer verifies the player’s request to hit.

The dealer draws the next card from the deck:

The dealer takes one card from the top of the shuffled deck.

The dealer places the drawn card face up in front of the player.

The system updates the player's hand total.

If the hand total exceeds 21

The system prompts a message that the player has bust.

The player’s turn ends immediately, and their bet is lost.

If the total is less than 21:

The player continues their turn and can take any action according to UC004.

The game state is updated to reflect the new total player’s hand.

**Exceptions:**

If the system is unavailable, it will display a message informing the user of the inconvenience.

If the deck runs low on cards during the dealing process, the dealer must shuffle and redeal before proceeding.

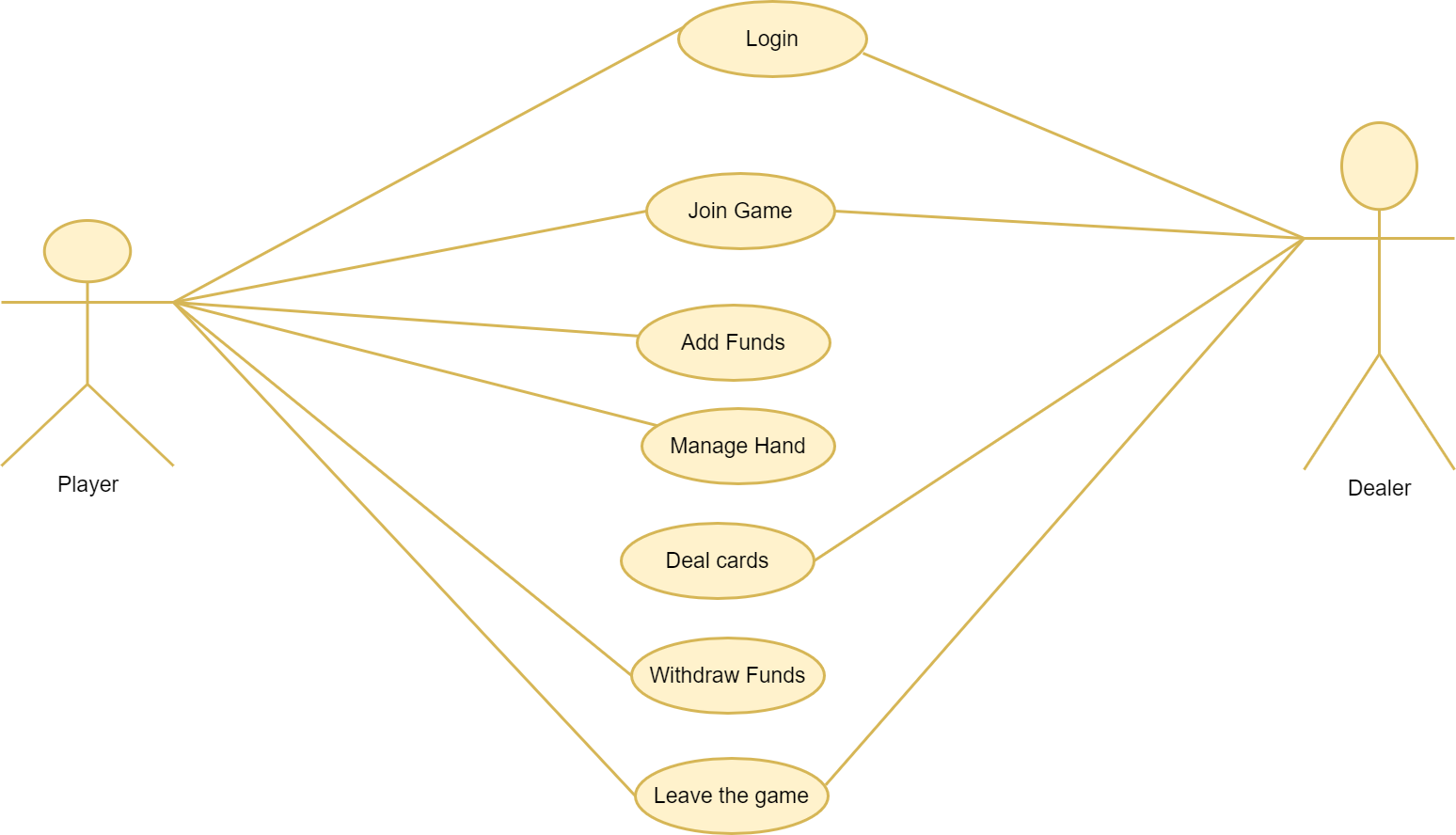
**Related Use Cases:**

Join Game (UC01)

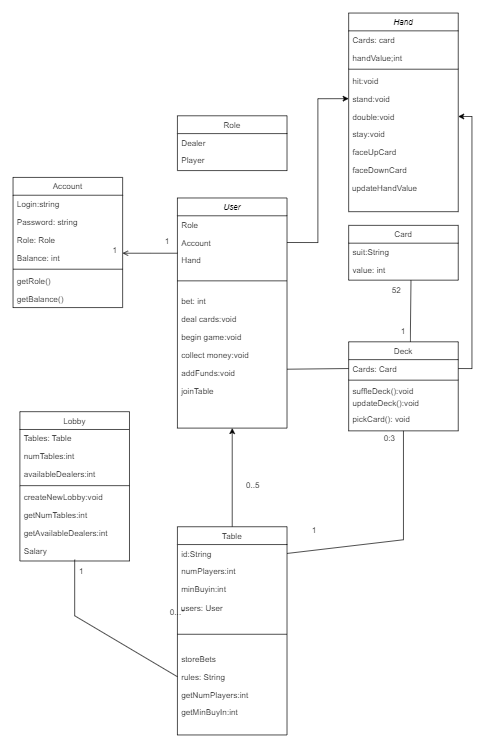
Manage Hand (UC04)

Leave Game(UC05)

1.3.2 UML Use Case Diagrams Document

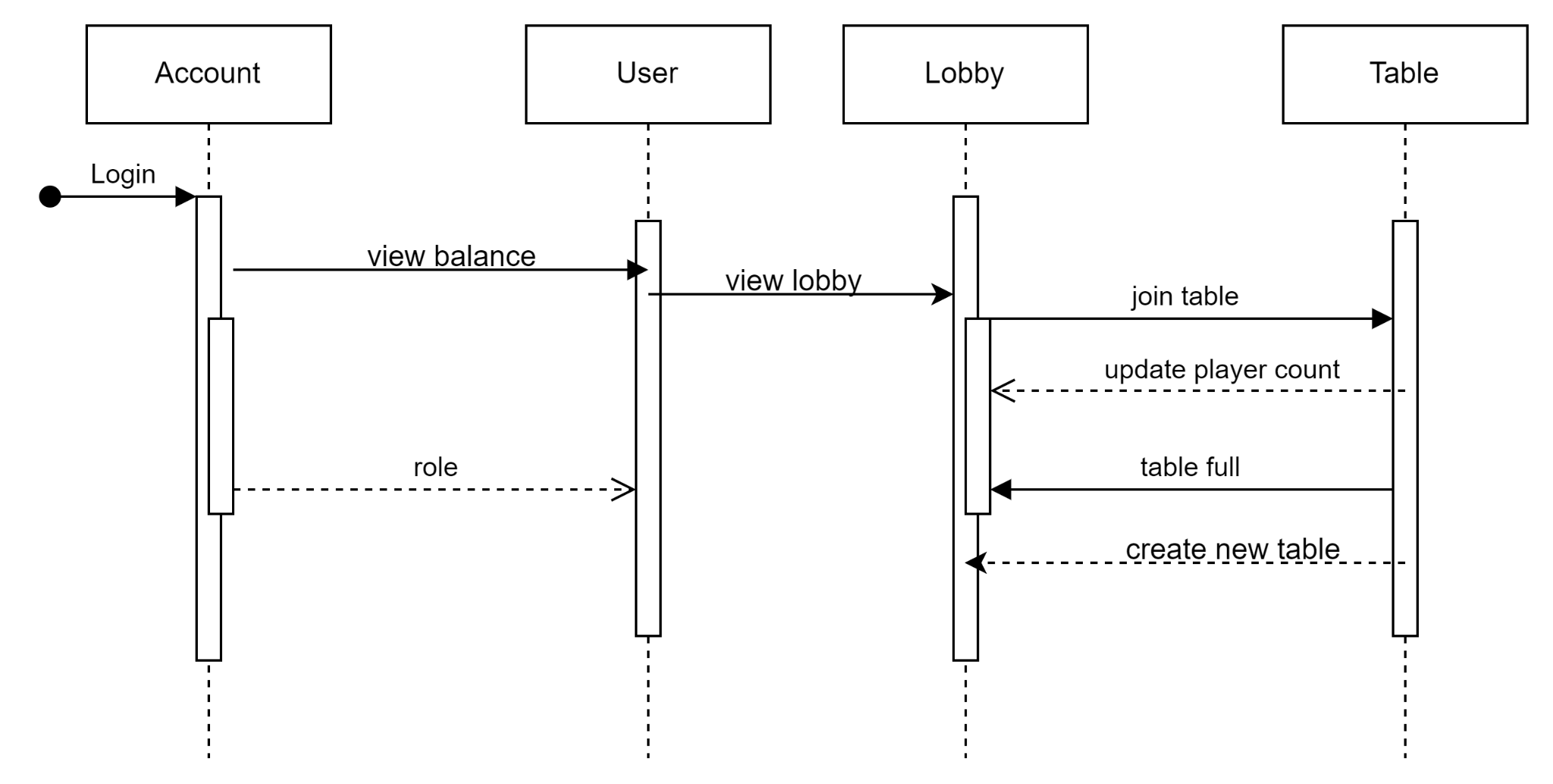


1.3.3 Class Diagram

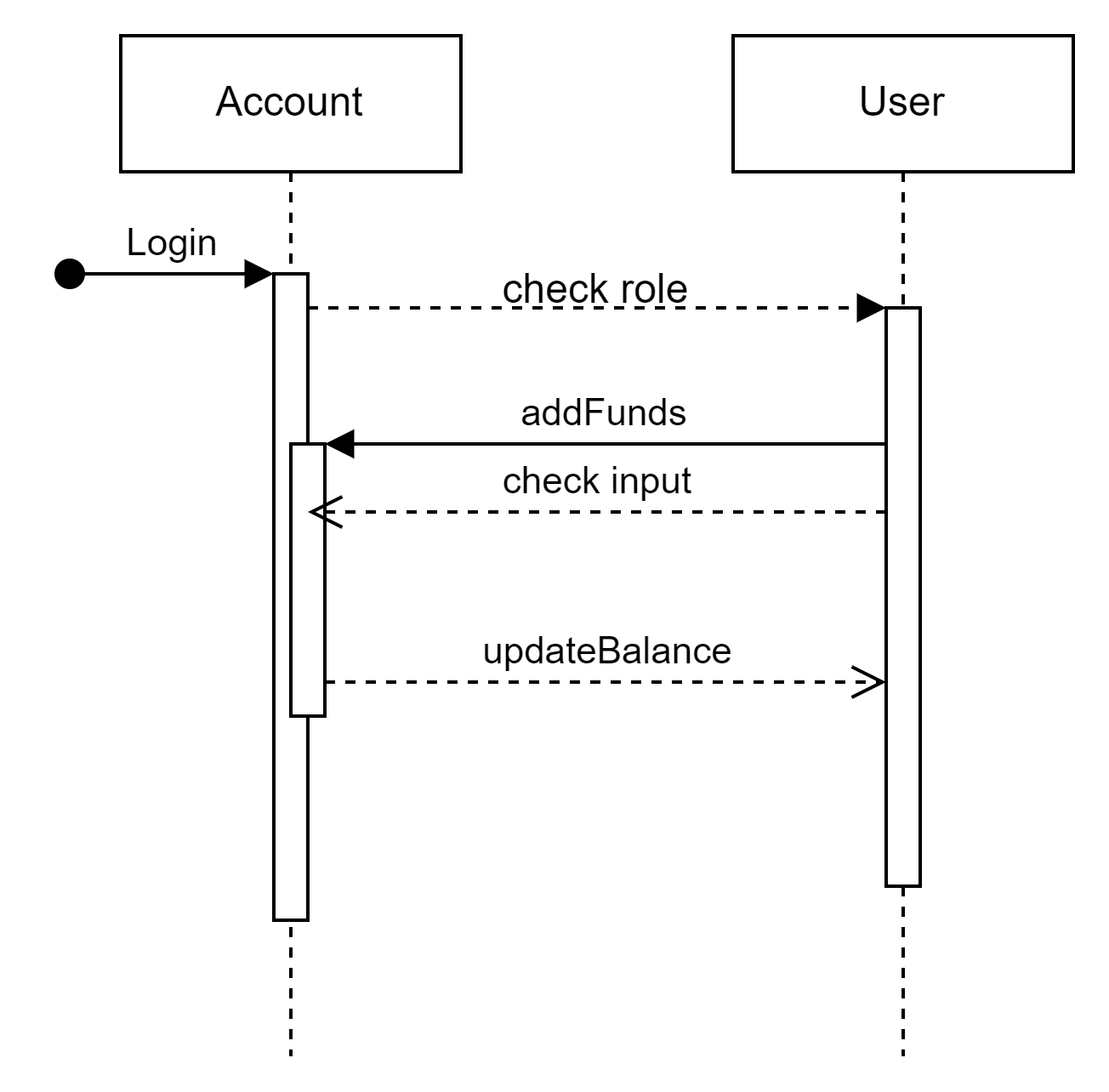


1.3.4 Sequence Diagrams

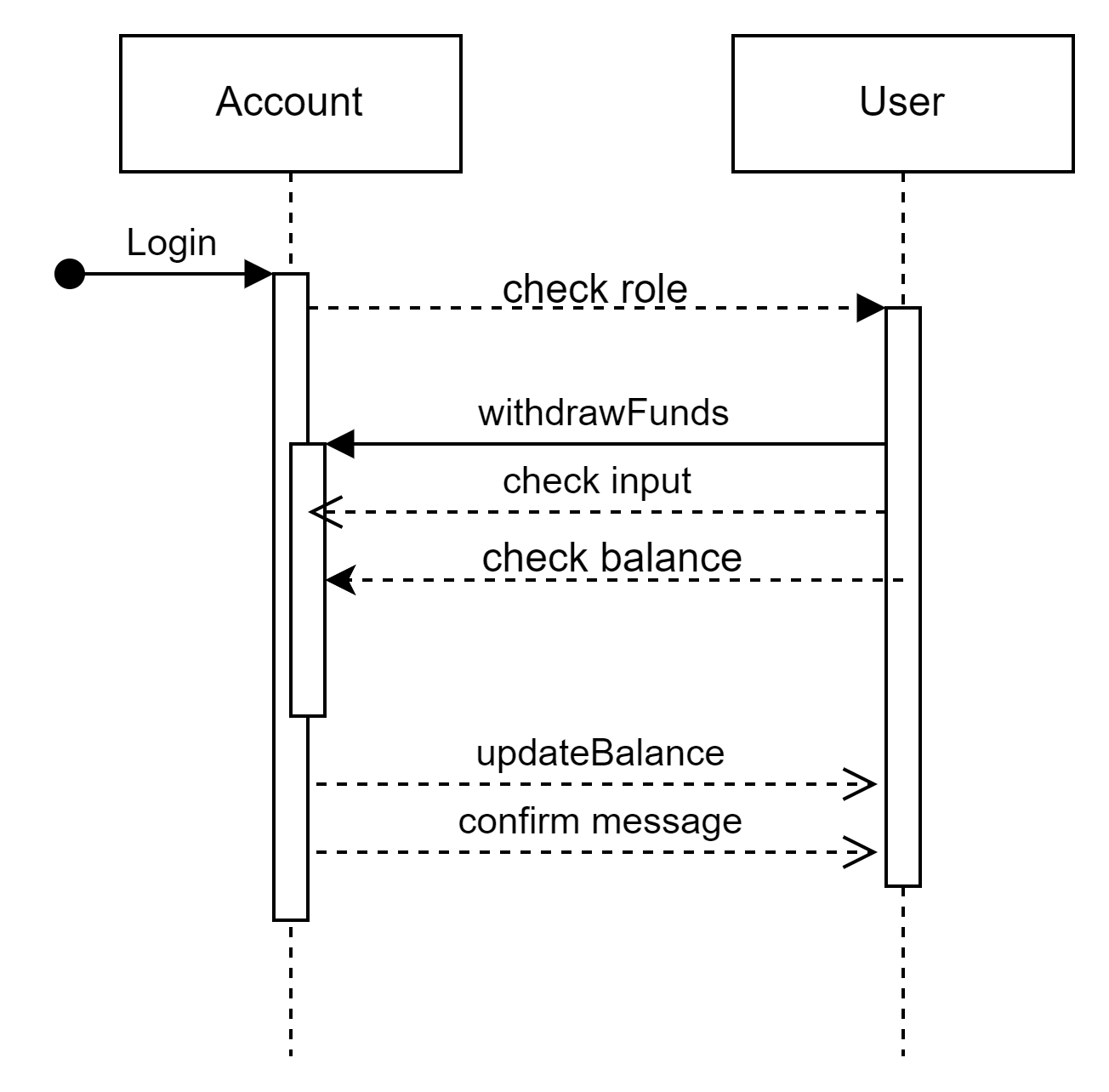
1.3.4.1 **UC01: Join Game**



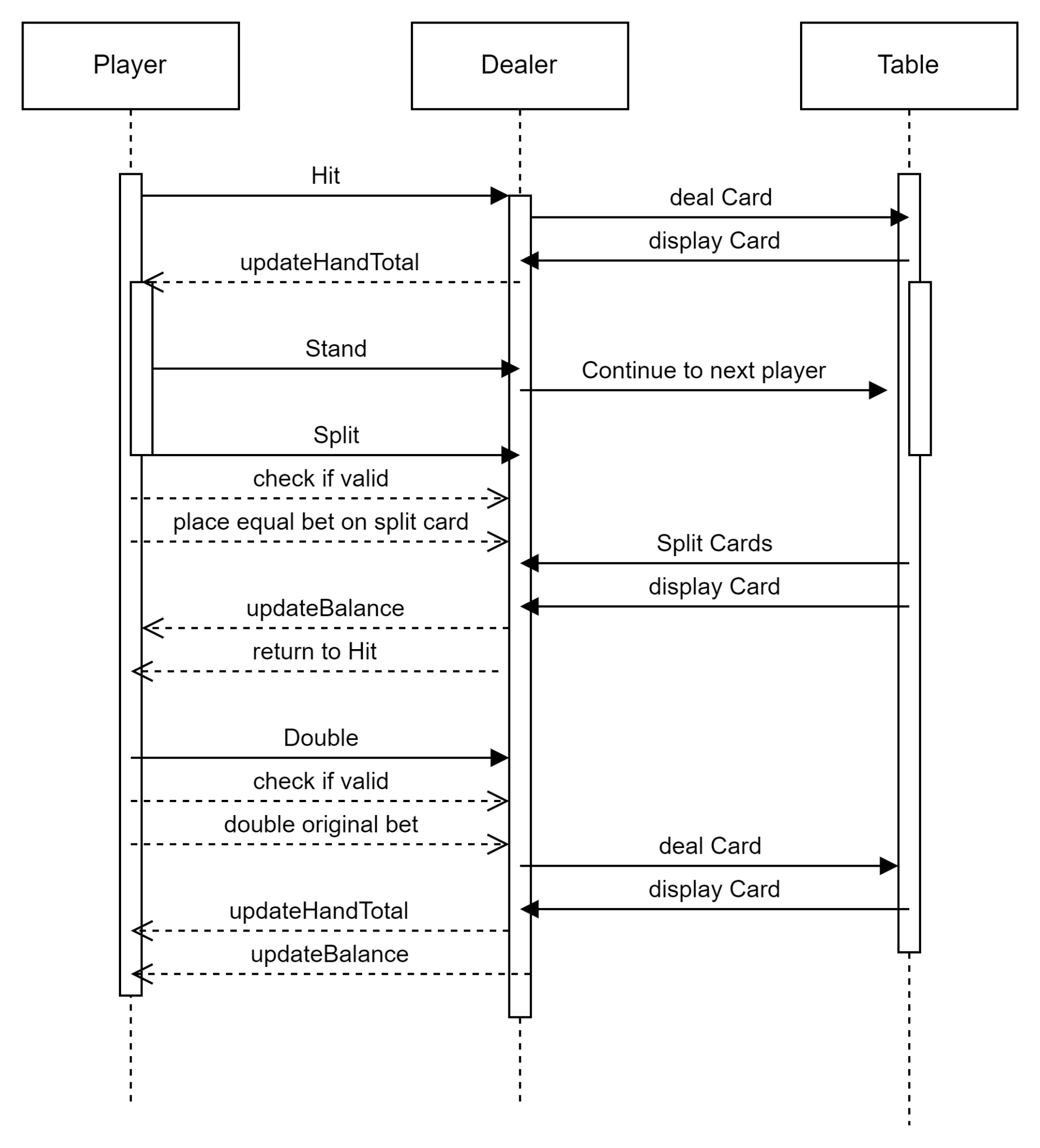
1.3.4.2 **UC02: Add funds**



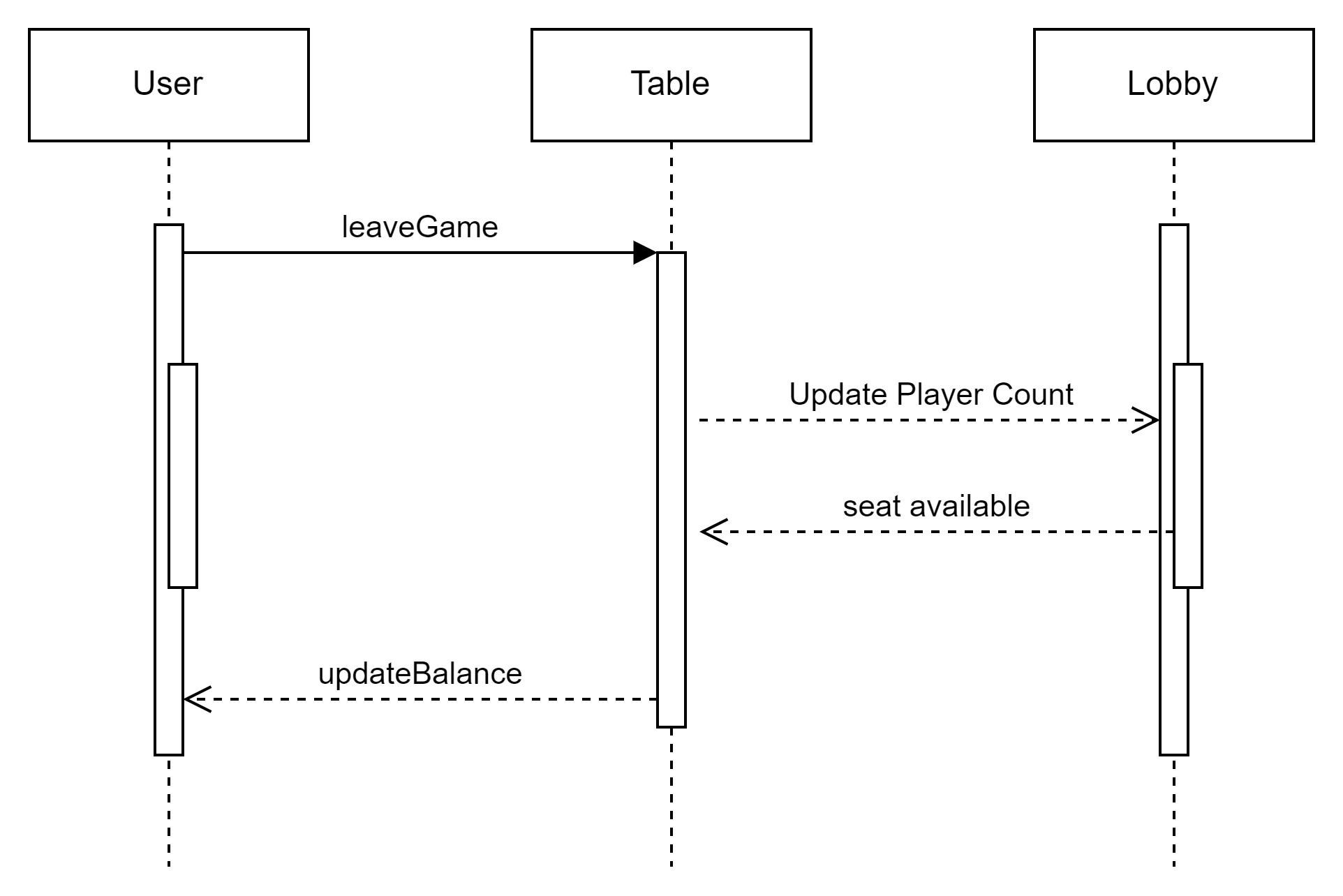
1.3.4.3 **UC03: Withdraw Funds**



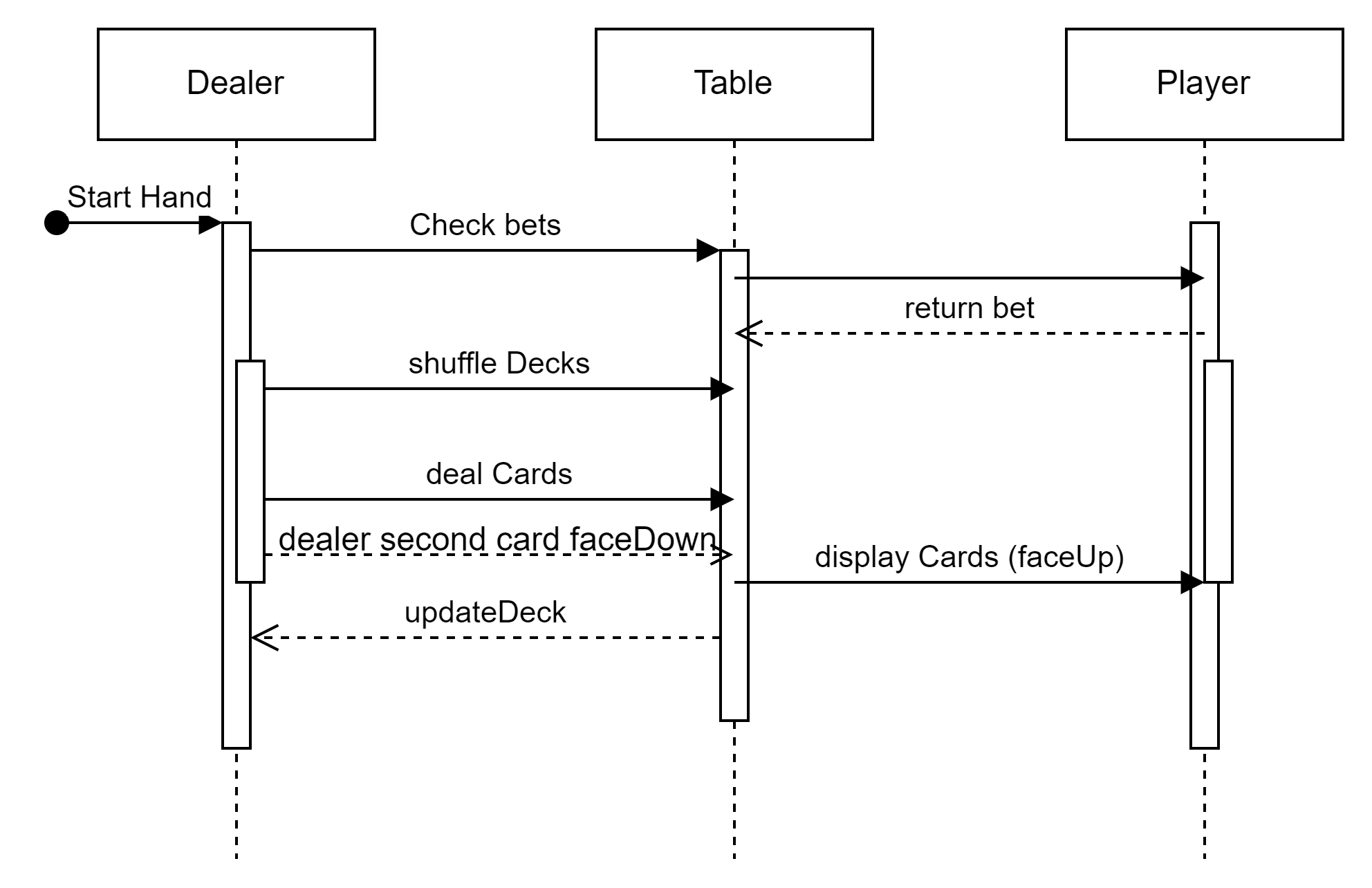
1.3.4.4 **UC04: Manage Hand**

****

**1.3.4.5 UC05: Leave Game**



1.3.4.6 **UC06: Deal cards**



## Overview

Blackjack developed in Java allows players and dealers to interact in a virtual environment. Blackjack is the most widely played casino game in the world. Each player, in turn, competes against the dealer, but players do not play against each other. In Blackjack, all players and the dealer try to build a hand that totals 21 points without going over. The hand closest to 21 wins.

# Overall Description

## Product Perspective

Multi-Player Jack Black is designed to provide an experience to play blackjack with other online real players. This game is designed to support multiple people playing as either the Dealer or the Player. This is based in Java, in order to support the vast majority of systems and users.

## Product Architecture

The system will be organized into 5 different modules: The User module, The Hand Module, the Deck Module, The Card Module, The Lobby Module

Note: System architecture should follow standard OO design practices.

## 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):

## Constraints

* + 1. Each blackjack game will be limited to 6 players, a dealer and 5 other players. Multiple games will be running at the same time.
    2. Players may be able to leave or disconnect from the game at any point of time, but they will forfeit any uncompleted bets to the house.

## Assumptions and Dependencies

* + 1. It is assumed that each blackjack game will be using a set deck of hards which are shuffled between games
    2. It is assumed that the game will be played with standard blackjack rules.
    3. It is assumed that each user will only be allowed to login once to a certain account
    4. It is assumed that users will be playing one game at a time.
    5. It is assumed that users will properly fund their accounts before joining the game.

# Specific Requirements

## Functional Requirements

### Common Requirements:

* + - 1. Users will be able to log in and out using a username and password
      2. User’s account will be either a dealer account or a player account
      3. Depending on a user’s account’s role they will use the dealer or player interface
      4. Players can add funds to their account
      5. Players can perform all the standard blackjack moves (hit, stand, split double down, etc)
      6. Dealers will be able to deal cards to players
      7. Users can NOT create their own accounts. (accounts are provided to users by admin and stored in a text file).
      8. Players and dealers are paid out according to who wins.
      9. Players can cash out their balances
      10. Players can join open tables and leave when they are done.
      11. If all tables are full, players will have to wait until a dealer opens another one or someone else leaves another table.
      12. Tables are created by a dealer.
      13. Users should not be able to call methods out of turn
    1. **Account Module Requirements:**

3.1.2.1. Users sign in with their username, password, and role to access the app.

3.1.2.2. Players can view their account balance.

### User Module Requirements:

3.1.3.1. Users should be able to access their account balance at any time through a user

interface. The balance should be updated in real-time after any transaction(add/withdrawn funds).

3.1.3.2. Players/dealers can perform standar blackjack actions according to their roles.

For players:

**Hit**: Players can request an additional card to be added to their hand.

**Stand**: Players can choose to keep their current hand and end their turn.

**Split**: Players can split their hand into two separate hands when they have two cards of the same value.

**Double Down**: Players can double their initial bet in exchange for committing to stand after receiving one more card.

For Dealers:

**Deal cards**: Dealers can deal cards to players and themselves according to the game rules.

3.1.3.3. Users should have the ability to add funds to their accounts, the system will update the account balance accordingly.

3.1.3.4. Players should be able to withdraw funds from their accounts, The system should update the balance.

3.1.3.5. Users can join open game tables when the table does not exceed its capacity limit

.

### Hand Module Requirements:

3.1.3.1 Players can view their hand, which includes all cards dealt to them.

3.1.3.2 Players can select the “Hit” option to receive an additional card.

The system deals one additional card to the player's hand.

Update the total value of the hand.

Check if the new total exceeds 21, if so, declare the player as bust.

3.1.3.3 Players can select the “Stand” option to keep their current hand.

Dealer goes to the next player turn.

3.1.3.4 Players can select the “Split” option if they have two cards of the same rank .

3.1.3.5 Players can select the “Double Down” option to double their initial bet before the

dealers give the player another card.

### Deck Module Requirements:

3.1.5.1 The game must initialize a standard 52 card-deck, utilizing a shoe containing four decks, at the start of each round.

3.1.5.2 A shuffle function must be implemented to randomize the deck before dealing.

### 3.1.5 Card Module Requirements:

3.1.5.1 The individual card must have some interface showing the dealer or player what

the card represents

3.1.5.2 The cards must also be logically correct so that if there is a 9 of spades showing

as the interface then the logic or game should not use it as something else like a

ace of hearts

3.1.5.3 The cards must follow the standard deck of playing cards there can not be

something like a 9 of stars

### 3.1.6 Lobby Module Requirements:

3.1.6.1 When the dealer identifies that all existing tables are full, they can creates a new table.

3.1.6.2 Players must be able to leave and join Tables appropriately without disrupting a current game

### 3.2 External Interface Requirements

3.2.1 The system when it starts must provide an interface that allows the user to login and

if the login is wrong it will be forced to retry.

3.2.2 After logging in there will be buttons where they are able to withdraw, deposit and

join a game that the user is able to join.

3.2.3 Choosing withdraw or deposit would be very similar interface where they can put in

or take out money.

3.2.4 Choosing join game would be sent to a lobby where they can join a table

3.2.5 After joining a table they will be able to play the game corresponding to their role

3.2.6 There is a interface to show the card that is drawn and the cards on the table

3.2.7 An interface to allow the users to play the game

## 3.3 Internal Interface Requirements

3.3.1 The system must have a way to check the login information correctly and set the

information of the users accordingly. The information will be in a text file

separated by new line characters.

3.3.2 Once logged in the logic of the flow for the program must correctly lead to the

option that is chosen: withdraw, deposit, play game.

3.3.3 For adding funds the system must take an input from the user and check if the input

is valid. Then if it is valid it update the user’s account balance and if it not valid

it will prompt the user to enter a valid input.

3.3.4 For withdrawing the system will check if the amount inputted is enough to not

make the user be in an invalid amount. Then it will update the user’s balance.

3.3.5 With joining the game the system will have the number of tables and include the

population of the table. If the population of of the table is full with all the tables

and there is a user in the main game section then a dealer will have to create a

new table so that the user is able to join a table.

3.3.6 Once in the table there will be some system to store all the cards that will be dealt

for the player and the dealer and the cards will be all randomized. After that the

once the cards are pulled from the deck it will be transferred to the dealer first

then to the players hands and remove the card from the deck.

3.3.7 The system will properly allow the player to get more cards from the deck and go

into the player's and dealer hand which is called hitting. The “hitting” aspect will

also go at the start for everyone so everyone gets 2 cards. The system will allow

users to stand which is the same as skipping a turn. Splitting is only applicable

to the players and instead of having one hand it is split into two hands and is

being able to be played twice. Double down is also for the players which allows

the player to place the same bet they put at the beginning of play.

3.3.8 The system will check the hands of the players each time they hit and if it is above

21 it is losing the play. Then at the end it will compare the players score

compared to the dealers hand to see who wins and the points go accordingly to

the winners and losers.

# Non-Functional Requirements

## Security and Privacy Requirements

* + 1. The System must not encrypt any data
    2. All data is stored in plain text

## Environmental Requirements

* + 1. Application is a Server-Client application
    2. Both Server and Client is written in Java
    3. Client application must have a GUI

## Performance Requirements

4.3.1 The system should handle real-times updates for player balances and game states with minimal delay.

4.3.2 The system should efficiently handle multiple simultaneous user requests without significant latency, ensuring a smooth gaming experience.