Black Jack

Software Requirements Specification

Revision History

Date	Revision	Description	Author
2/16/2024	1.0.1.1	Module Description: Server, Client multithreaded operation	Sandeep Deoja
2/21/2024	1.0.2.1	Spec Requirements: Login, Player, Dealer, & Game Logic Module	Andrew Nguyen
2/21/2024	1.0.3.1	Descriptions: Product Perspective, Architecture, Func/Features, Constraints & Assumptions/Deps	Ayoub Mekkaoui
2/21/2024	1.0.3.2	Spec Requirements: Common, GUI, External/Internal Interface	Ayoub Mekkaoui
2/21/2024	1.0.3.3	Use Case Document: added ID 001, ID 002	Ayoub Mekkaoui
2/21/2024	1.0.3.4	Spec Requirements: Game Logic and Dealer module	Andrew Nguyen
2/21/2024	1.0.3.5	Use Case Document: added ID 003, ID 004	Andrew Nguyen
2/24/2024	1.0.3.6	Use Case Document: added ID 005	Ishwdeep Singh
2/25/2024	1.0.3.7	Purpose: added Definitions, Acronyms, Abbreviations	Ishwdeep Singh
2/26/2024	1.0.3.8	Definitions:Module, Interface,Flagged, Verified Connection,etc. Use Case:006. Non-Functional Requirement: Security, Environmental and Performance requirements	Sandeep Deoja
2/26/2024	1.0.4.1	Created Class Diagram	Ayoub Mekkaoui
2/26/2024	1.0.4.2	Spec Requirements: Game Logic and Dealer module	Andrew Nguyen
2/27/2024	1.0.4.3	Changes to the Server Client Requirement	Sandeep Deoja
2/27/2024	1.0.4.4	Use Case Diagram: Client Server Use Case	Sandeep Deoja
2/27/2024	1.0.4.5	Use Case Diagram: Login and Player action	Andrew Nguyen
2/27/2024	1.0.4.6	Sequence Diagram 1 - User Login	Ayoub Mekkaoui
2/27/2024	1.0.4.7	Sequence Diagram 2 - Player Action	Ayoub Mekkaoui
2/28/2024	1.0.5.1	Use Case Diagram: Dealer action	Andrew Nguyen
3/15/2024	1.1.0	Use Case Document: added ID 007, 008, 009, 010, 011	Ayoub Mekkaoui
3/18/2024	1.1.0.1	Use Case Document: added ID 012, 013, 014, 015, 016	Andrew Nguyen
3/18/2024	1.1.0.2	Spec Requirements: Client and Server module	Andrew Nguyen
3/24/2024	1.2.0	Added 4 new Sequence Diagrams (Player Hit/Stand, Dealer Start/End Game)	Ayoub Mekkaoui
3/25/2024	1.2.1	Added 3 more Sequence Diagrams (Player Add/Withdraw funds, Player requests game history)	Ayoub Mekkaoui
3/26/2024	1.2.1	Revised Class Diagram	Andrew Nguyen
3/26/2024	1.2.1.1	Use Case Document: added ID 017, 018, 019, 020, 021,022	Andrew Nguyen
3/28/2024	1.2.1.2	UML Revision for LogIn and Client Server Design	Sandeep Deoja
3/28/2024	1.2.1.3	UML for Client Message to Server	Sandeep Deoja
3/28/2024	1.2.1.4	UML for Error Handling for Multithreaded and Connection Operations	Sandeep Deoja
3/28/2024	1.2.1.5	UML for BroadCasting Current Status of the Round to all Players and Dealer, added Use Case:019	Sandeep Deoja

3/28/2024	1.2.1.6	Use Case added: 023,024,025,026	Sandeep Deoja
3/28/2024	1.2.1.7	Use Case Document: added ID 027,028,029,030,031	Andrew Nguyen
3/28/2024	1.2.1.8	Updated Client and Server requirements	Andrew Nguyen

Table of Content

1. Purpose

This document outlines the requirements for the Black Jack Game.

1.1 Scope

This document outlines the requirements for a Black Jack Game based on Client Server Architecture over TCP/IP.

1.2 Definitions, Acronyms, Abbreviations

Player Log In - This is where the player logs in.
Dealer Log In - This is where the dealer logs in.
Hit - Player chooses to draw a card.
Stand - Player chooses to end their turn.
Module: - A common theme that umbrellas one or many classes
Multiple Connections: Multiplayer system
Port: An Abstraction for ingress and egress
Interface: A pre-existing defined class which enforces certain behavior
Messages: Data
Verified Connection: An established channel of communication between two mediums
Flagged: Define a certain characteristic or behavior
Recorded: data stored in a text file
Ingress: incoming
Egress: outgoing
System Crash: shutting down of the software without any notification to the user

1.3 References

Use Case Specification Document
UML Use Case Diagrams Document
Class Diagrams
Sequence Diagrams

2. Overall Description

2.1 Product Perspective

The Black Jack Gaming System is an application that allows multiple users to play the classic card game of Black Jack over the internet. It incorporates a basic graphical user interface (GUI) to facilitate user interaction, providing essential functions like logging in, account management, and gameplay. By using standard server/client protocols and sound game logic, we can promote a safe, fair and simple platform for players to play the game of Black Jack.

2.2 Product Architecture

The system will be organized into 8 major modules: the Sensor Module, the Server Module, the Client Module, the Log-In Module, the Player Game Module, the Dealer Module, the Game Logic Module, and the GUI Module.

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):

-Enables multiple players to join and play in the same game session.
-Players can make decisions in real-time.
-Users can create and manage their accounts, including tracking their wins, losses, and funds.
-The system includes an automated dealer to manage the deck, deal cards, and enforce game rules.

2.4 Constraints

-The game requires a stable internet connection for all players.
-Due to the simplistic nature of the project, basic security features for user authentication and data protection.
-The system is designed for web browsers and cannot be used for mobile devices.

2.5 Assumptions and Dependencies

-Assumes users have access to an internet connection.
-Assumes server has ability to handle multiple concurrent connections without significant latency.
-Assumes the Game Logic Module ensures fairness and enforces rules without error.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Common Requirements:

3.1.1.1 Implement basic error handling to catch and alert for common errors such as invalid inputs or connection issues.
3.1.1.2 Implement a simple login system without encryption, using plain text file for storing valid username and password storage.
3.1.1.3 Ensures GUI maintains a consistent layout and design across different modules to enhance user experience.

3.1.2 Server Module Requirements:

3.1.2.1 The Server will need to have an open communication channel.
3.1.2.2 The Server will need to service multiple or one Client at a time or simultaneously.
3.1.2.3 Errors should not cause the program to crash.
3.1.2.4 The Server should save Player data every time win/loss or bank status changes.
3.1.2.5 The server sends login confirmation message to the client.
3.1.2.6 The server will update the game state for each client after every player and dealer action.
3.1.2.7 The server sends place bet success/fail messages to client.
3.1.2.8 The server sends withdraw success/fail messages to client.

3.1.3 Client Module Requirements:

3.1.3.1 A Client can connect to the Server through a communication channel designated by the Server.
3.1.3.2 There can be one,many Clients connected to the Server simultaneously.
3.1.3.3 Any error in the communication between Client and the Server should not crash the system.
3.1.3.4 The client sends login requests to the server.
3.1.3.5 The client sends hit requests to the server.
3.1.3.6 The client sends stand requests to the server.
3.1.3.7 The client sends double down requests to the server.
3.1.3.8 The client sends start round request to the server.
3.1.3.9 The client sends end round request to the server.
3.1.3.10 The client sends deposit requests to the server.
3.1.3.11 The client sends withdraw requests to the server.
3.1.3.12 The client sends place bet requests to the server.
3.1.3.13 The client sends logout requests to the server.

3.1.4 Log-in Module Requirements:

3.1.4.1 The client will request a username and password so the user may log in
3.1.4.2 The log-in data for each user will be stored in a separate text file
3.1.4.3 The log-in module will compare and validate username and password to the contents of the file
3.1.4.4 Each account will be flagged as either dealer or player
3.1.4.5 Within the account management section, the system shall provide a feature for players to view their total wins and losses. This aims to offer players insight into their performance over time.

3.1.5 Player Module Requirements:

3.1.5.1 The player has the option to hit, stand, and double down.
3.1.5.2 The player is only able to bet before the round starts.
3.1.5.3 The player can only double down if they have enough funds to double their bet.
3.1.5.4 The player can not change their bet after the cards are dealt.
3.1.5.5 The player hand value will be stored and read to ensure fair play
3.1.5.6 The player will be allowed to deposit money to play
3.1.5.7 The player’s bankroll will be unique to them.

3.1.5.8 The player will not be able to bet more than what they have in their bankroll.
3.1.5.9 The player can not bet a negative amount.
3.1.5.10 Track and Store Player Win/Loss Records: The system shall track the outcome of each game for every player and update their win/loss record accordingly. This record will be stored within the player's profile.
3.1.5.11 The player will not be in the round if they did not make a bet.

3.1.6 Dealer Module Requirements:

3.1.6.1 The Dealer will initiate the start of the game once everyone has placed their bets.
3.1.6.2 The dealer's turn will start once all players stand or bust.
3.1.6.3 The dealer's hand value will be stored and read to ensure fair play
3.1.6.4 The dealer will be able to draw after the players are done with their turns.
3.1.6.5 The dealer will be able to press a button to finish the round which will payout or take from players depending on if they won or lost the round.

3.1.7 Game Logic Module Requirements:

3.1.7.1 If the dealer's hand value after the initial draw is under 17, the dealer will automatically draw a card
3.1.7.2 The round is complete when all players stand or bust
3.1.7.3 The dealer starts their turn after the players stand or bust.
3.1.7.3 If the dealer's hand value is over 21 it is an automatic win for any player still in the round.
3.1.7.4 If the player's hand value is over 21 it is an automatic loss for the player.
3.1.7.5 If the dealer and player have the same hand value at the end of the round, there will be a draw and the player does not lose any money.
3.1.7.6 On the very first action,the system allows the player to double down. This doubles their bet at the cost of being able to draw only one card. The system will then move to the next player to request their action.
3.1.7.7 To ensure fairness, the game will be played with 2 decks of 52 cards.
3.1.7.8 The decks will be randomized before the round starts.
3.1.7.9 Only the dealer's first card will be known by the players until all players stand or bust.
3.1.7.10 Since aces are worth 1 or 11, the Ace will be counted as 1 if the players hand would be over 21 and Aces will be counted as 11 if the players hand would be under 21.
3.1.7.11 Jack, Queens, and Kings will count as 10 towards the players and dealers hand.
3.1.7.12 Numbered cards will count as the number towards the players and dealers hand.
3.1.7.13 If the player has 21 with the first 2 cards it is considered a BlackJack which is an automatic win for the player if the dealer does not have 21.
3.1.7.14 If both the player and dealer have 21 then it will be considered a tie and the player will not lose their bet.
3.1.7.15 During the round, only one player is allowed to take action at a time to ensure fair play.
3.1.7.16 When a player stands(finishes their turn) the system will request the next player for their action.
3.1.7.17 The dealer's turn will start once all players stand.
3.1.7.18 At the start of the round, the players and dealer will be dealt 2 cards.

3.1.8 GUI Requirements:

3.1.8.1 Designed interface to be intuitive, allowing for easy navigation and interaction with the application.
3.1.8.2 Display error messages for invalid operations or inputs to guide users towards correct actions.
3.1.8.3 Display game rules throughout gameplay to help navigate players through each game action.
3.1.8.4 Dealer accounts will have different button options than player accounts
3.1.8.5 Dealer accounts will have access to buttons that start and end the rounds.
3.1.8.6 Player buttons will be disabled if it is not their turn.

3.2 External Interface Requirements

3.2.1 Use TCP/IP sockets for establishing and managing connections between clients and the server, detailing protocols for message formatting and handling.
3.2.2 Define the methods for serializing data for transmission over sockets, specifying formats for encoding game actions, and state information.
3.2.3 Implement a message queue or a similar mechanism to handle asynchronous communications and callbacks between modules, ensuring non-blocking operations.

3.2.4 Define a standardized access layer for interacting with the file system or local databases, supporting CRUD operations (Create, Read, Update, Delete) for game state, user profiles, and transaction logs.

3.3 Internal Interface Requirements

3.3.1 Simple Communication Between Modules - Modules communicate using straightforward method calls and return basic data types or simple objects.
3.3.2 Use simple file operations (open, read, write, close) for handling game data and user information with plain text files.
3.3.3 Minimal Logging - Implement a basic logging function that records errors and key events to a text file, without detailed configuration.

4. Non-Functional Requirements

4.1 Security and Privacy Requirements:

4.1.1 For certain features such as gamer progress and editing player profile, user data is recorded.

4.2 Environmental Requirements:

4.2.1 The System should be compatible for Java development.
4.2.2 A connection between server and client has to be established and communication to take place within the network.
4.2.3 The flow of data between the server and client is handled by inbuilt Java class such as ServerSocket and Socket Class.

4.3 Performance Requirements

4.3.1 The connection between the client and the server will be through an ingress and egress port designated by the server.
4.3.2 The file sharing mechanism within the system will be handled by inbuilt Java I/O class.
4.3.3 If there are errors that will prevent the system from running or errors that cause the system to crash, those errors need to be caught using try/catch method and notify the user of the error. Specifics of the error need not be shown to the players.

Black Jack

Design Document

Table of Content

Use Case Description

Actors

1. ***Player:*** A user who interacts with the game to play Black Jack. This includes making bets, deciding to hit or stand, and managing their account.
2. ***Dealer:*** The system component that automates the role of the card dealer in Black Jack, dealing cards and managing the game rules.
3. ***System:*** Represents the backend logic and databases that handle game sessions, user authentication, and data persistence.

Use Case ID: 001
Use Case Name: Player Login
Relevant Requirements: 3.1.4 Log-in Module Requirements
Primary Actor: Player
Pre-conditions: Player has an account registered with the system.
Post-conditions: Player is logged into their account and can access the game lobby.
Basic Flow or Main Scenario: Player opens the application and selects the login option. → Player enters their username and password. → System verifies the credentials against the stored data. → Player is granted access to the game lobby.
Extensions or Alternate Flows: If the credentials are incorrect, the player is informed and prompted to try again.
Related Use Cases: Account Management, Play Game

Use Case ID: 002
Use Case Name: Place Bet
Relevant Requirements: 3.1.5 Player Module Requirements
Primary Actor: Player
Pre-conditions: Player is logged in and has sufficient funds in their bankroll.
Post-conditions: Player has placed a bet for the upcoming round.
Basic Flow or Main Scenario:Player selects the option to place a bet for the next round. → Player inputs the amount they wish to bet. → System verifies that the player has enough funds. → Bet is registered, and funds are reserved for the round.
Extensions or Alternate Flows: If the player does not have enough funds, they are informed and asked to input a different amount.
Related Use Cases: Add Funds, Player Wins Round, Player Loses Round

Use Case ID: 003
Use Case Name: Add funds
Relevant Requirements: 3.1.5.5, 3.1.5.6
Primary Actor: Player
Pre-conditions: Player is logged in.
Post-Conditions: Player allocates funds to their account.
Basic Flow or Main Scenario: Player selects the option to add funds to their account. The system responds by adding their funds to the players profile.
Extensions or Alternate Flows: If the player enters a non-positive number, they will be notified to input a positive number.
Related Use Cases: Place Bet, Player Log in.

Use Case ID: 004
Use Case Name: Player Action (Hit/Stand)
Relevant Requirements: 3.1.5.1, 3.1.7.6
Primary Actor: Player
Pre-conditions: Player is logged in and playing the round.
Post-conditions: Player is able to hit, stand, or double down.
Basic Flow or Main Scenario: Player selects the option to either hit or stand. The system responds by requesting the correct action for the player and updating their hand or user state.
Extensions or Alternate flows: If the player chooses to double down, the system will respond by drawing them one card, doubling their bet, and moving to the next player.
Related Use Cases: Place bet, Player Log in

Use Case ID: 005
Use Case Name: Dealer Action
Relevant Requirements: 3.1.6, 3.1.7
Primary Actor: Dealer
Pre-conditions: The dealer is logged in and hosting the round with a set number of players.
Post-conditions: The dealer will have dealt cards and auto-draw its own card, with 1 card value being shown to players and 1 card hidden. If the round is over, the dealer will finish the round for the game logic module to determine its outcome.
Extensions or Alternate flows: —
Related Use Cases: Dealer Place bet, Dealer Log in.

Use Case ID:006
Use Case Name: Server Client Interaction
Relevant Requirements: 3.1.2,3.1.3
Primary Actor: Systems Administrator
Pre-Conditions: The Computer System is on
Post-Conditions: A network connection is established with one port designated as a server. There can be multiple or zero clients connected to the designated server port. Once the connection is established and verified, the clients and servers can communicate.
Basic Flow: 1)A designated port will listen for any incoming connection.2) The client module will start the connection to the server. 3) Once the server verifies the connection, data can then be passed within the network.
Alternate Flow: The login module will be the first module to run on the network. The Player Account Management section will help the player to edit their profile. The Gaming Module will move the game forward.
Related Use Cases: Other related use cases will be the player module, the dealer module.
Exceptions: Connection is not verified. Error in the InputStream and OutputStream.

Use Case ID: 007
Use Case Name: View Wins and Losses
Relevant Requirements: 3.1.4.5, 3.1.5.10
Primary Actor: Player
Pre-conditions: Player is logged in.
Post-conditions: Player has viewed their win/loss record.
Basic Flow: 1) Player navigates to the account management section. 2) Player selects the option to view win/loss records. 3) System retrieves and displays the player's win/loss record from their profile.
Extensions or Alternate Flows: If the player's win/loss record is not available or there's an error, display an error message and suggest trying again later.
Related Use Cases: Player Login

Use Case ID: 008
Use Case Name: Withdraw Funds
Relevant Requirements: 3.1.5.6
Primary Actor: Player
Pre-conditions: Player is logged in and has funds in their account.
Post-conditions: Player has withdrawn funds from their account.
Basic Flow or Main Scenario:
Player accesses the account management section.
Player selects the option to withdraw funds.
Player inputs the amount they wish to withdraw and confirms.
System processes the withdrawal, updates the account balance, and confirms the transaction to the player.
Extensions or Alternate Flows:
If insufficient funds, notify the player and cancel the transaction.
Related Use Cases: Player Login, Add Funds

Use Case ID: 009
Use Case Name: Dealer Initiates Game
Relevant Requirements: 3.1.6.1
Primary Actor: Dealer
Pre-conditions: All players have placed their bets.
Post-conditions: The game round starts.
Basic Flow or Main Scenario: 1) Dealer verifies all players have placed their bets. 2) Dealer presses the Start Round button. 3) System deals two cards to each player and the dealer.
Extensions or Alternate Flows: If not all players have placed bets, the Start Round button remains disabled.
Related Use Cases: Place Bet

Use Case ID: 010
Use Case Name: Update Player Account After Game
Relevant Requirements: 3.1.2.4, 3.1.5.10
Primary Actor: System
Pre-conditions: A game round has concluded.
Post-conditions: Player accounts are updated with the result.
Basic Flow or Main Scenario: 1) The system calculates the result of the game. 2) The system updates each player's win/loss record and account balance based on the game's outcome. 3) Players are notified of the update.
Extensions or Alternate Flows: If an error occurs during the update, log the error and attempt to retry the update process.
Related Use Cases: Dealer Action, Player Action

Use Case ID: 011
Use Case Name: Player Requests Game History
Relevant Requirements: 3.1.5.10
Primary Actor: Player
Pre-conditions: Player is logged in.
Post-conditions: Player has viewed their past game history.
Basic Flow or Main Scenario: 1) Player navigates to the account management section. 2) Player selects the option to view past games. 3) System retrieves and displays a list of past games and outcomes.
Extensions or Alternate Flows: If the system cannot retrieve the history, display an error message.
Related Use Cases: Player Login

Use Case ID: 012
Use Case Name: Client Sends Login Request to Server
Relevant Requirements: 3.1.3.4
Primary Actor: Client
Pre-Conditions: GUI is started up and the User enters login information
Post-Conditions: The user is logged in
Basic Flow or Main Scenario: 1) User enters their username and password. 2) The client creates a login object with the username and password as attributes. 3) The client sends the serialized login object to the server. 4) The server receives the login request. 5) The server responds to the login request by validating the username and password against the stored user database.
Extensions or Alternate Flows: If the server cannot validate the login information, a prompt for login information is displayed again on the GUI.
Related Use Cases: Player Login, Dealer Login

Use Case ID: 013
Use Case Name: Server sends login confirmation message to the client
Relevant Requirements: 3.1.2.5
Primary Actor: Server
Pre-Conditions: Client sent login object with username and password as attributes to the server
Post-Condition: Server sends confirmation message to the client
Basic Flow or Main Scenario: 1)Server receives login message with username and password as attributes. 2) Server validates username and password against stored user database. 3) Server confirms the username and password matches with one in the user data base.4) Server sends a user information object to the client with username, player id, and bankroll as attributes.
Extensions or Alternate Flows: If the username and password are matched to a dealer account, the Server will send a dealer information object to the client with access level as attributes to provide the dealer with the appropriate GUI.
Related Use Cases: Client Sends Login Request to Server

Use Case ID: 014
Use Case Name: Server broadcasts game state to all clients
Relevant Requirements: 3.1.2.6
Primary Actor: Server
Pre-Conditions: The client sends a player or dealer action object to the server
Post-Conditions: Server broadcasts and updates the current game state to all clients
Basic Flow or Main Scenario: 1)Server receives player or dealer request. 2) Server processes the request. 3) Once the server processes the request, the game state i.e player hands, dealer hands, are updated on every client
Extensions or Alternate Flows: If there are no requests that warrant updating the gamestate, the server will not broadcast the game state to every client.
Related Use Cases: Client sends player or dealer hit request to server, Client sends player's stand request to server, Client sends player's double down request to server,

Use Case ID: 015
Use Case Name: Client sends player or dealer hit request to server
Relevant Requirements: 3.1.3.5
Primary Actor: Client
Pre-Conditions: Player or dealer requested for another card (hit)
Post-Conditions: Server updates corresponding player or dealers hand value and broadcasts game state
Basic Flow or Main Scenario: 1) Player or dealer requests a hit. 2) Client sends hit message object with player ID and username or dealer boolean as attributes. 3) Server processes this message. 4)The server will give the player or dealer a card through the GameLogic class. 5) The server will then update the game state for all connected clients.
Extensions or Alternate Flows: If there are no hit requests, the server will not broadcast the game state to every client.

Use Case ID: 016
Use Case Name: Client sends player's stand request to server
Relevant Requirements: 3.1.3.6
Primary Actor: Client
Pre-Conditions: Player requests to stand
Post-Conditions: Server updates player's status to stand and broadcasts game state
Basic Flow or Main Scenario: 1) Player requests to stand. 2) Client sends message to the server with type stand. 3) Client handler processes the request. 4) The player's stand value will be set to true. 5) The server will update the game state for all connected clients. 6) The server will move onto the next player and request for their action.
Extensions or Alternate Flows: If there are no stand requests from the player, the server will not broadcast the game state to every client
Related Use Cases: Place bet, client sends player or dealer hit request to server

Use Case ID: 017
Use Case Name: Client sends player's double down request to server
Relevant Requirements: 3.1.3.7
Primary Actor: Client
Pre-Conditions: Player requests to double down
Post-Conditions: Server updates player's hand, status to stand, and broadcasts game state
Basic Flow or Main Scenario: 1) Player requests to double down. 3) Client sends message to the server with type double down. 3) Client Handler processes the request. 4) GameLogic adds a card to player's hand and does the appropriate checks 5) Player stand value will be set to true. 6) The server will update the game state for all connected clients. 7) The server will move onto the next player and request for their action.
Extensions or Alternate Flows: If the player double's down and has a hand value over 21, their status will be set to bust
Related Use Cases: Place bet, Client sends dealer's start round request to server

Use Case ID: 018
Use Case Name: Client sends dealer's start round request to server
Relevant Requirements: 3.1.3.8
Primary Actor: Client
Pre-Conditions: Dealer requests to start the game
Post-Conditions: Server adds 2 cards to all players that bet and first player can request an action.
Basic Flow or Main Scenario: 1)All players that want to play the round have made a bet. 2) Dealer requests to start the round. 3) GameLogic class adds cards to the dealer and every players hand. 4) Once all initial cards have been dealt, the first player will have their buttons enabled to make an action.
Extensions or Alternate Flows: If no players have made a bet the Client will not be able to send a start game request.
Related Use Cases:

Use Case ID: 019
Use Case Name: Client sends dealer's end round request to server
Relevant Requirements: 3.1.3.9
Primary Actor: Client
Pre-Conditions: Dealer requests to end the round
Post-Conditions: Server ends the round and does the appropriate checks, pays out players that won and takes money from players that lost
Basic Flow or Main Scenario: 1)Dealer requests to end the round. 2) The client sends a message of type end_round to server. 3) The client handler processes this request. 4) The server checks each hand that has not bust against the dealer's hand. 5) The server pays the players who won and takes money from players who lost.6) The server sends a message to the client allowing the dealer to start the next round.
Extensions or Alternate Flows: The dealer will not be able to request to end the round unless all players are stand or bust and the dealer's turn is finished
Related Use Cases: Stand, Dealer action

Use Case ID: 020
Use Case Name: Client sends player's deposit request to server
Relevant Requirements: 3.1.3.10
Primary Actor: Client
Pre-Conditions: Player requests to deposit funds
Post-Conditions: Player bankroll is updated
Basic Flow or Main Scenario: 1) Player requests to deposit funds. 2) Player enters an integer to deposit funds to their account. 3) The client handler processes this request. 4) The server adds the allocated amount to the players bankroll. 5) The server replies with a message to update the client of the players bankroll
Extensions or Alternate Flows:
Related Use Cases: Add funds

Use Case ID: 021
Use Case Name: Client sends player's withdraw request to server
Relevant Requirements: 3.1.3.11
Primary Actor: Client
Pre-Conditions: Player requests to withdraw positive funds
Post-Conditions: Player withdraws from their bankroll
Basic Flow or Main Scenario: 1) Player requests to withdraw funds. 2) Player enters an integer to withdraw funds from their account. 3)The client handler processes this request. 4) The server subtracts the amount from the players bankroll. 5) The server replies with a message to update the client of the players bankroll
Extensions or Alternate Flows: If the player tries to withdraw more than what they have in the bankroll, the server will respond with a message of type withdraw_fail.
Related Use Cases: Add funds, Client sends player's deposit request to server

Use Case ID: 022
Use Case Name: Client sends player's place bet request to server
Relevant Requirements: 3.1.3.12
Primary Actor: Client
Pre-Conditions: Player requests to place a bet to join the round
Post-Conditions: The player's bet is updated and they are able to play the round
Basic Flow or Main Scenario: 1) Player requests to place bet. 2) The player enters a positive integer for their bet. 3) The client sends a message of type place bet to the server. 4) The server responds by placing the bet for that player
Extensions or Alternate Flows: If the player tries to bet more than they have in the bankroll, the server will deny their bet and send a message to the client that prompts the user to place another bet.
Related Use Cases: Place bet

Use Case ID: 023
Use Case Name: Player logout request
Relevant Requirements: 3.1.3.10
Primary Actor: Player
Pre-Conditions: Player has to be logged in the system
Post-Conditions: The player is logged out of the system and the corresponding thread is terminated
Basic Flow: 1) Player sends a message to the server about logging out of the system. 2) The message is received by the server. 3) The thread is then ended that corresponds to the client.
Alternate Flow: The logged out players info, the bank roll is then saved. The Client List<Clients> is updated.
Related Use Case: Player Login

Use Case ID: 024
Use Case Name: Client logged in as Player
Relevant Requirements: 3.1.4 Log-in Module Requirements
Primary Actor: Client
Pre-conditions: Player has an account registered with the system.
Post-conditions: Player is logged into their account and can access the game lobby.
Basic Flow or Main Scenario: Client opens the application and selects the login option. → Client enters their username and password. → login module verifies the credentials against the stored data. → Client is then granted access to the game lobby and assigned 1 to verify as a player. 4) The Player is then added to the List<Clients> in the server module.
Extensions or Alternate Flows: If the credentials are incorrect, the player is informed and prompted to try again.
Related Use Cases: Account Management, Play Game

Use Case ID: 025
Use Case Name: Client logged in as Dealer
Relevant Requirements: 3.1.4 Log-in Module Requirements
Primary Actor: Client
Pre-conditions: Client has an account registered with the system.
Post-conditions: Dealer is logged into their account and can access the game lobby.
Basic Flow or Main Scenario: Client opens the application and selects the login option. → Client enters their username and password. → login module verifies the credentials against the stored data. → Client is then granted access to the game lobby and assigned 0 to verify as a dealer. 4) The dealer is then added to the List<Client> in the server module.
Extensions or Alternate Flows: If the credentials are incorrect, the dealer is informed and prompted to try again.
Related Use Cases: Account Management, Play Game

Use Case ID: 026
Use Case Name: Multiple Players join the round and is added to the players list
Relevant Requirements:3.1.4
Primary Actor: ClientHandler
Pre-Conditions: The player list is empty
Post-Conditions: There are at least one player added to the list
Basic Flow: 1) The Client connects and is either assigned as a dealer or a player. 2) From the ClientList, the client is then added to the Map<Integer,Player> list.
Alternate Flow: Once a hand for a single player finishes, the result is then broadcasted to other players in the list.
Related Use Case:

Use Case ID: 027
Use Case Name: Server Sends Login Fail message to client
Relevant Requirements: 3.1.2.5
Primary Actor: Server
Pre-Conditions: The server received a login request and was not able to validate the username and password
Post-Conditions: The server sends a message of type login fail to the client
Basic Flow or Main Scenario: 1) The serve received a login request. 2) The server attempts to validate the login. 3) The login could not be validated so the server responds to the login request with a login fail message. 4) The client prompts the user to login again
Alternate Flow: If the login was successful, the user will be able to access their appropriate commands
Related Use Case: Client sends login request to server

Use Case ID: 028
Use Case Name: Server sends Place bet success message to client
Relevant Requirements: 3.1.2.7
Primary Actor: Server
Pre-Conditions: The server received a request to place a bet for the player.
Post-Conditions: The server successfully adds the bet to the player
Basic Flow or Main Scenario: 1) The server receives a place bet request. 2) The server checks if the player has enough funds and if they do to add it to their bet for the current round. 3)The server broadcasts the gamestate to all clients to update the players bet size. 4) The server sends a message of type place bet success to the client which will allow the player to make an action.
Extensions or Alternate Flows: If the bet size for the player is bigger then their bankroll, the server will send a place bet fail message to the client
Related Use Case: Place bet, Client sends place bet request to server

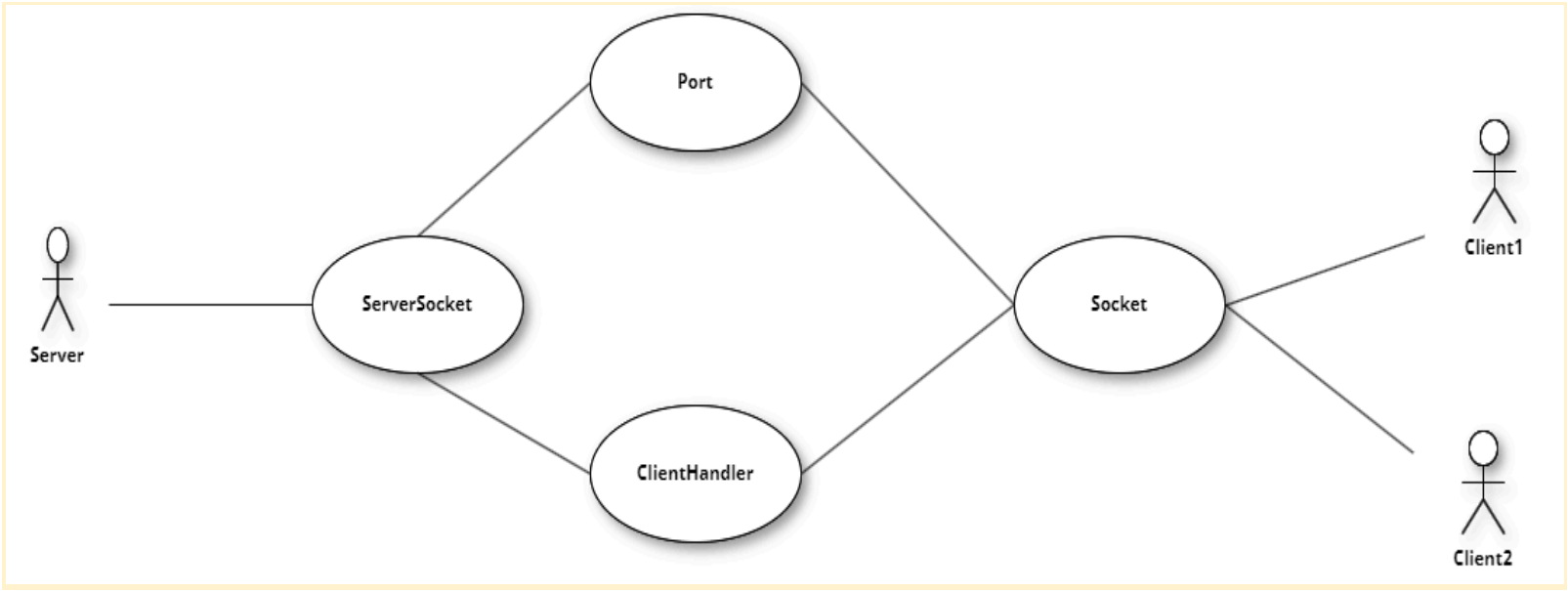
Use Case ID: 029
Use Case Name: Server sends Place bet fail message to client
Relevant Requirements: 3.1.2.7
Primary Actor: Server
Pre-Conditions: The server received a request to place a bet for the player
Post-Conditions: The server does not add the bet to the player
Basic Flow or Main Scenario: 1) The server receives a place bet request. 2) The server checks if the player has enough funds and if they do, add it to their bet for the current round. 3) The server sees that the user does not have enough funds to bet that much. 4) The server responds by sending a message of type place bet fail to the client prompting them for another bet.
Extensions or Alternate Flows: If the server receives a place bet request with a valid amount, the server will place the bet for the player
Related Use Cases: Place bet, Client sends place bet request to server.

Use Case ID: 030
Use Case Name: Server sends withdraw success message to client
Relevant Requirements: 3.1.2.8
Primary Actor: Server
Pre-Conditions: The server received a request to withdraw funds from the client
Post-Conditions: The funds are deducted from the players bankroll
Basic Flow or Main Scenario: 1) The server receives a withdraw request. 2) The server checks if the player has enough funds to withdraw the specified amount. 3) The server sees that the user has enough funds and deducts the specified amount from the user's bankroll. 4) The server responds with a message of type withdraw success, which will prompt the client to update the users bankroll after withdrawal
Extensions or Alternate Flows : If the server receives a withdraw request with an amount that is higher than the players bank roll, the server will respond by sending a withdraw fail message.
Related Use Cases: Withdraw funds, Client sends player's withdraw request to server.

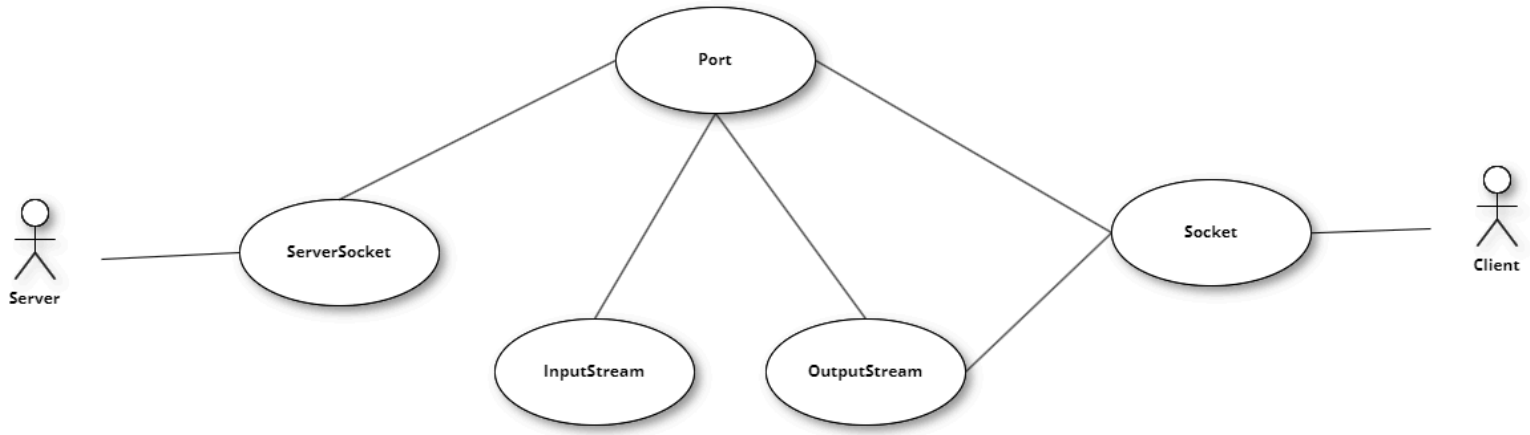
Use Case ID: 031
Use Case Name: Server sends withdraw fail message to client
Relevant Requirements: 3.1.2.8
Primary Actor: Server
Pre-Conditions: The server received a request to withdraw funds from the client
Post-Conditions: The funds are not deducted from the players bankroll
Basic Flow or Main Scenario: 1) The server receives a withdraw request. 2) The server checks if the player has enough funds to withdraw the specified amount. 3) The server sees that the user does not have enough funds. 4) The server responds with a message of type withdraw fail, which will prompt the client to ask for another withdraw amount
Extensions or Alternate Flows: If the server receives a withdraw request with an amount that is lower than the player's bank roll, the server will respond by sending a withdraw success message.
Related Use Cases: Withdraw funds, Client sends player's withdraw request to server.

Use Case Diagrams

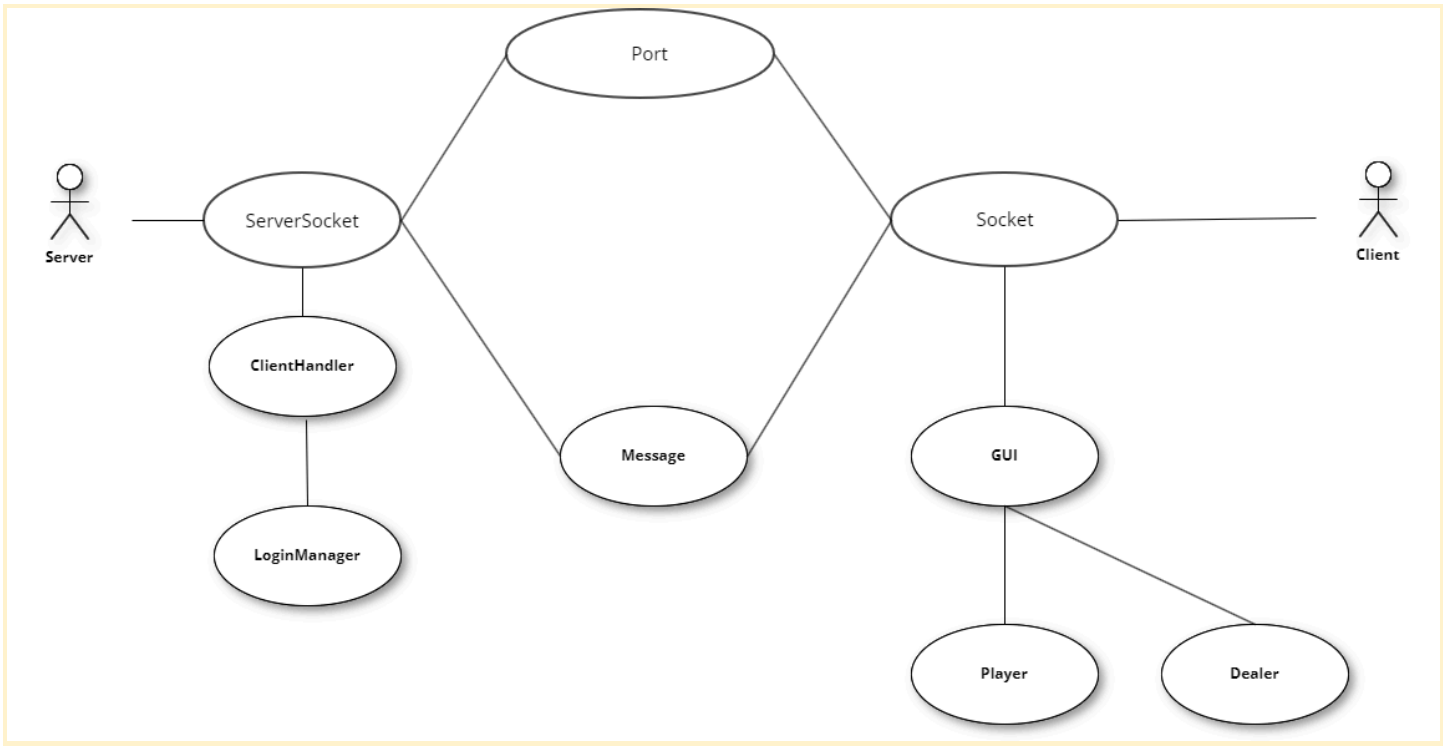
Use Case: Client Server



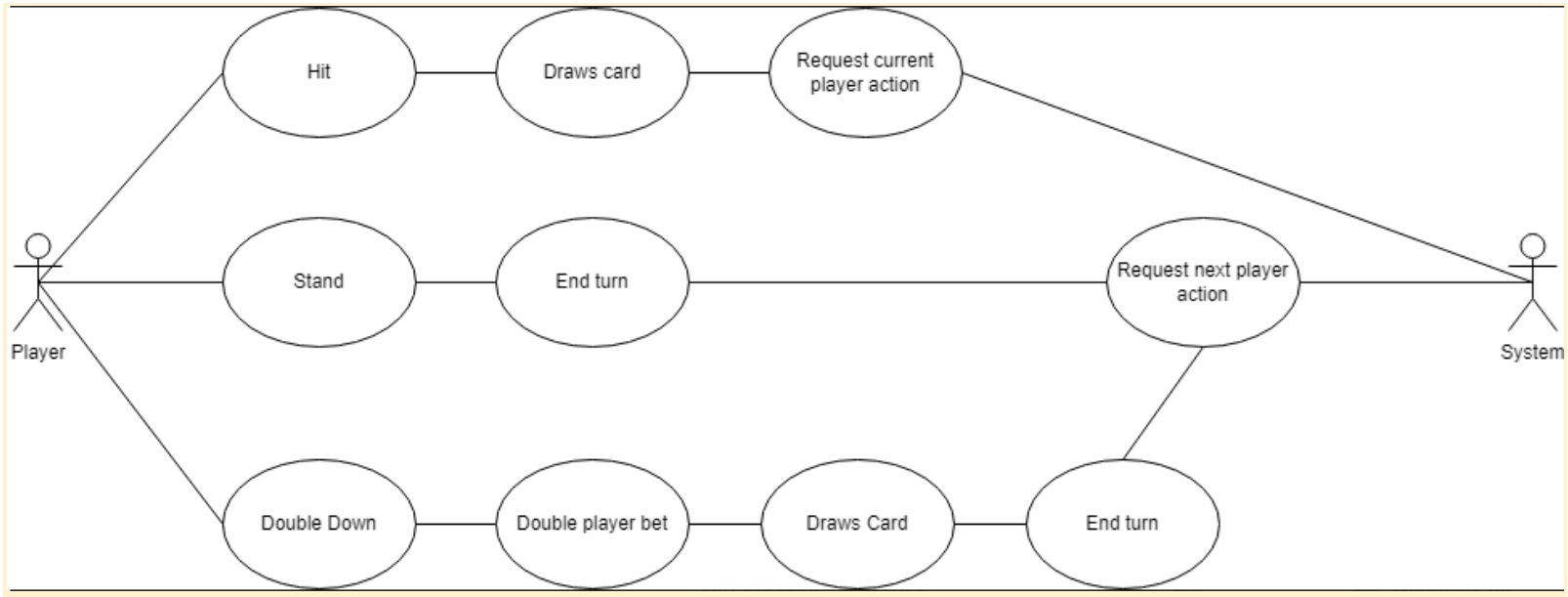
Use Case: Communication Channel for Client Server



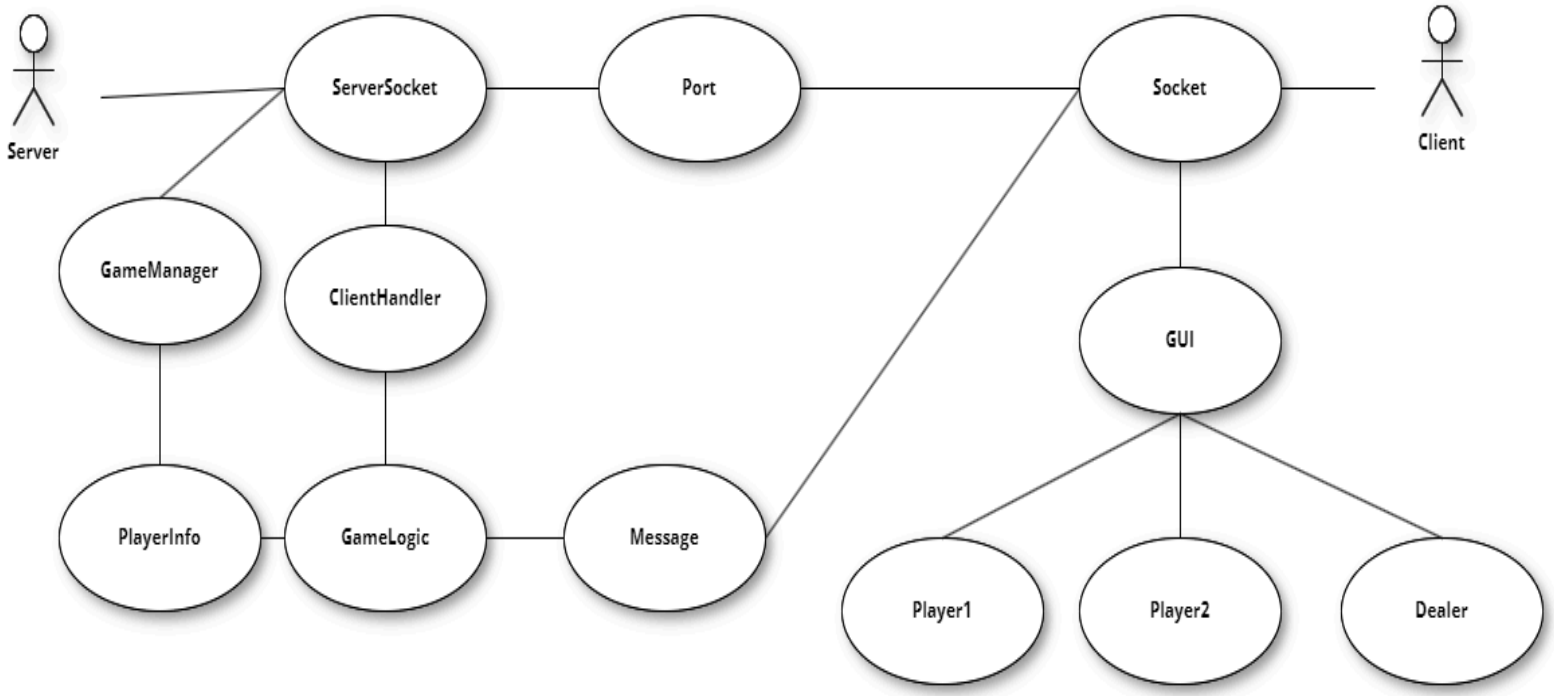
Use Case: User Login



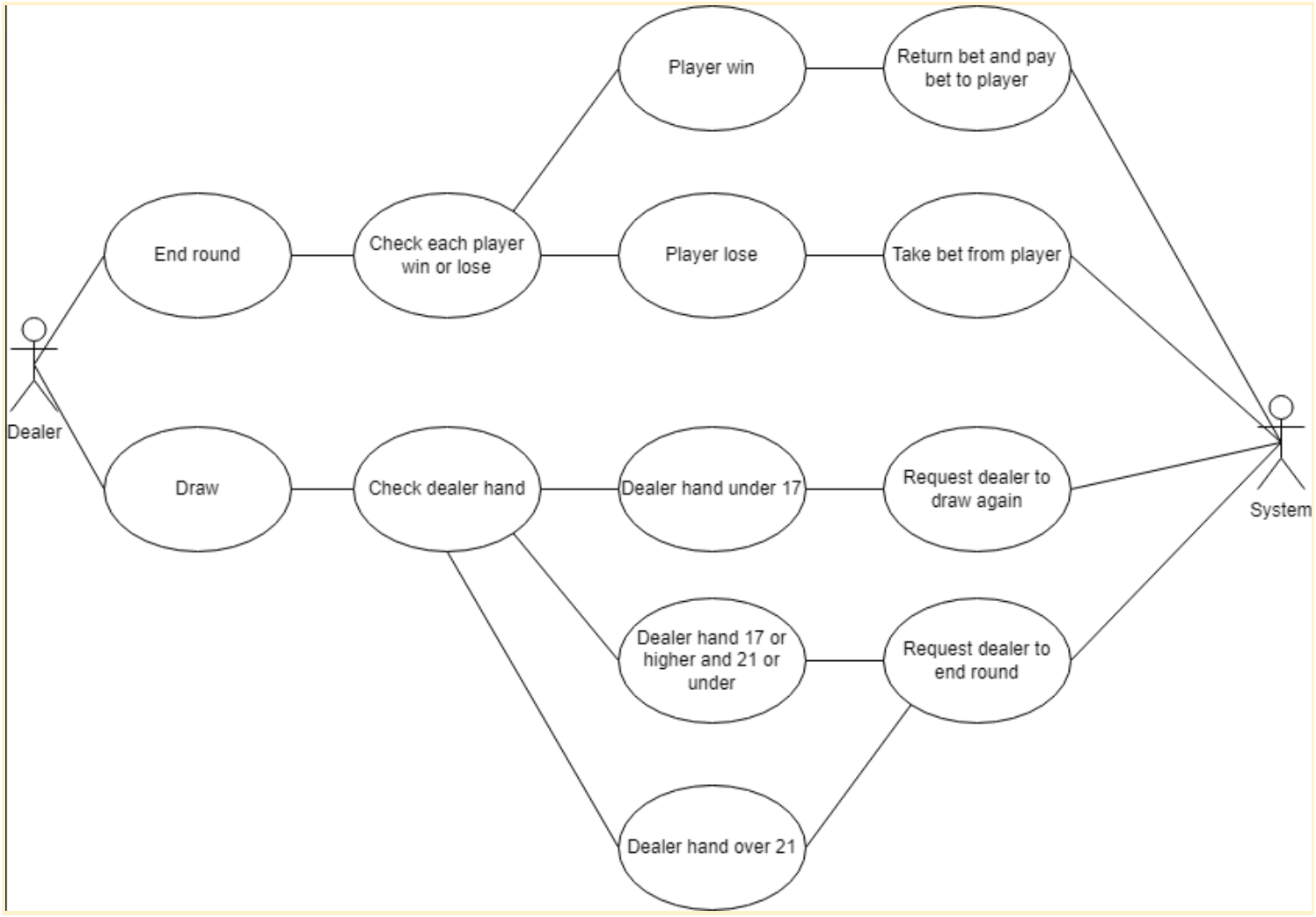
Use Case: Player Action



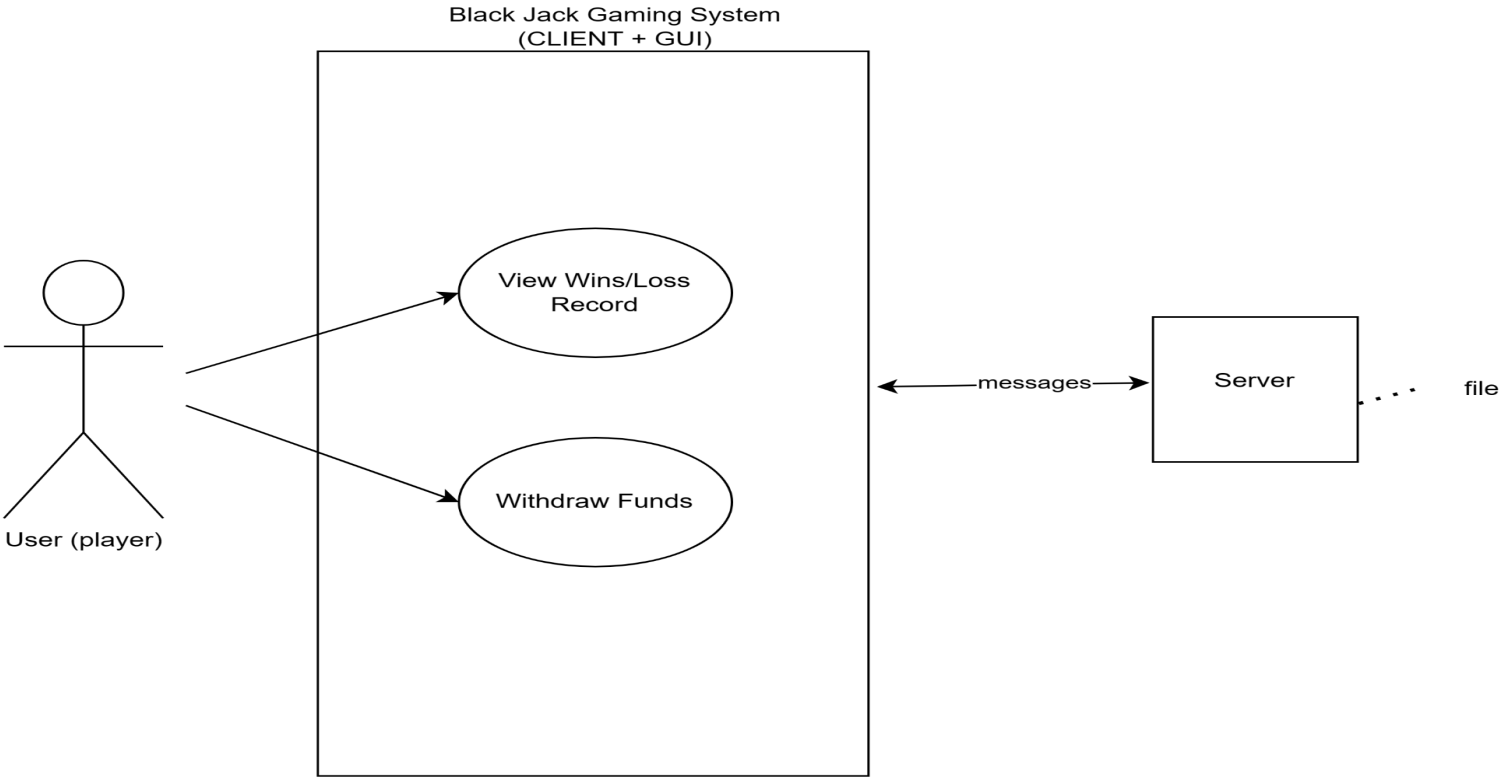
Use Case:BroadCast Status of the Current Game



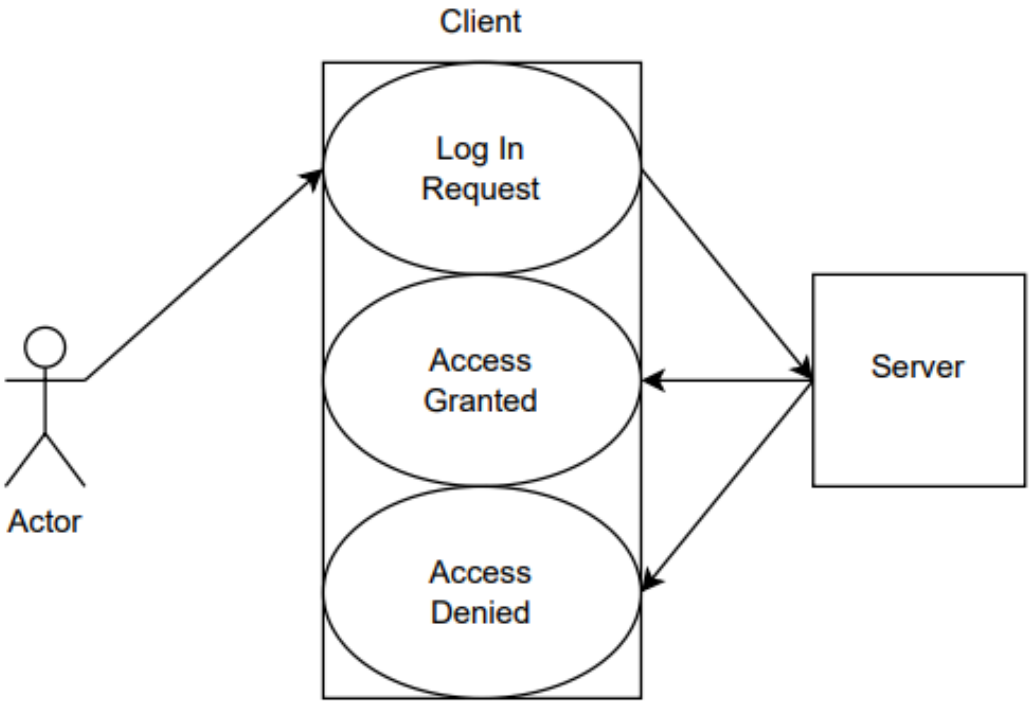
Use Case: Dealer action



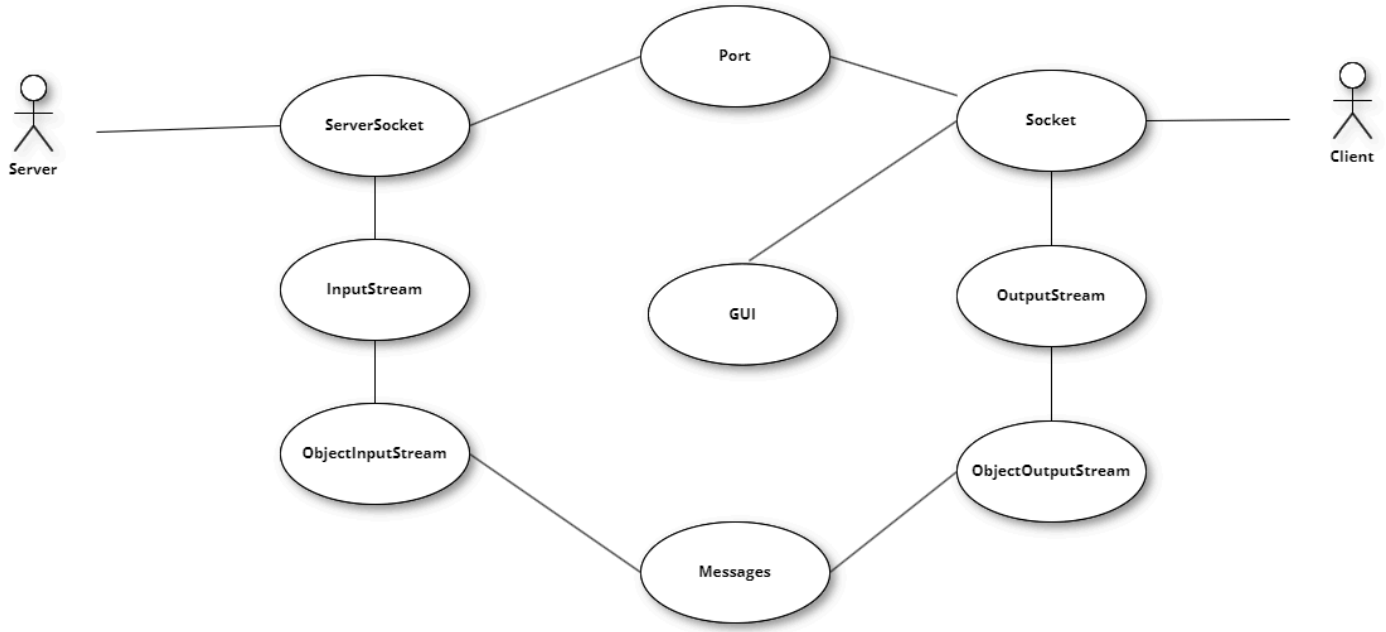
Use Case: Player Requests



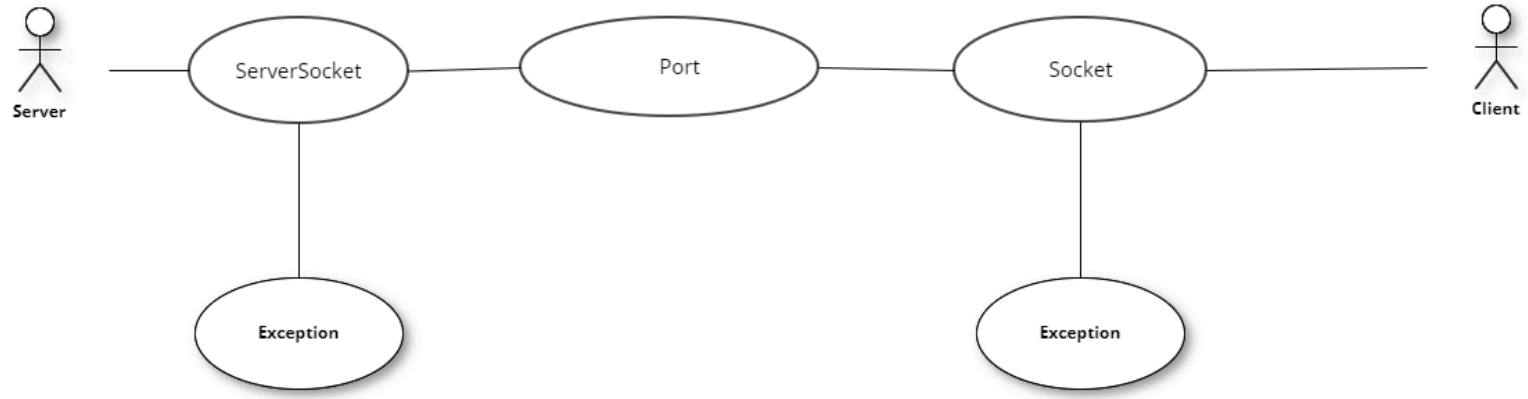
Use Case: Server Use Case



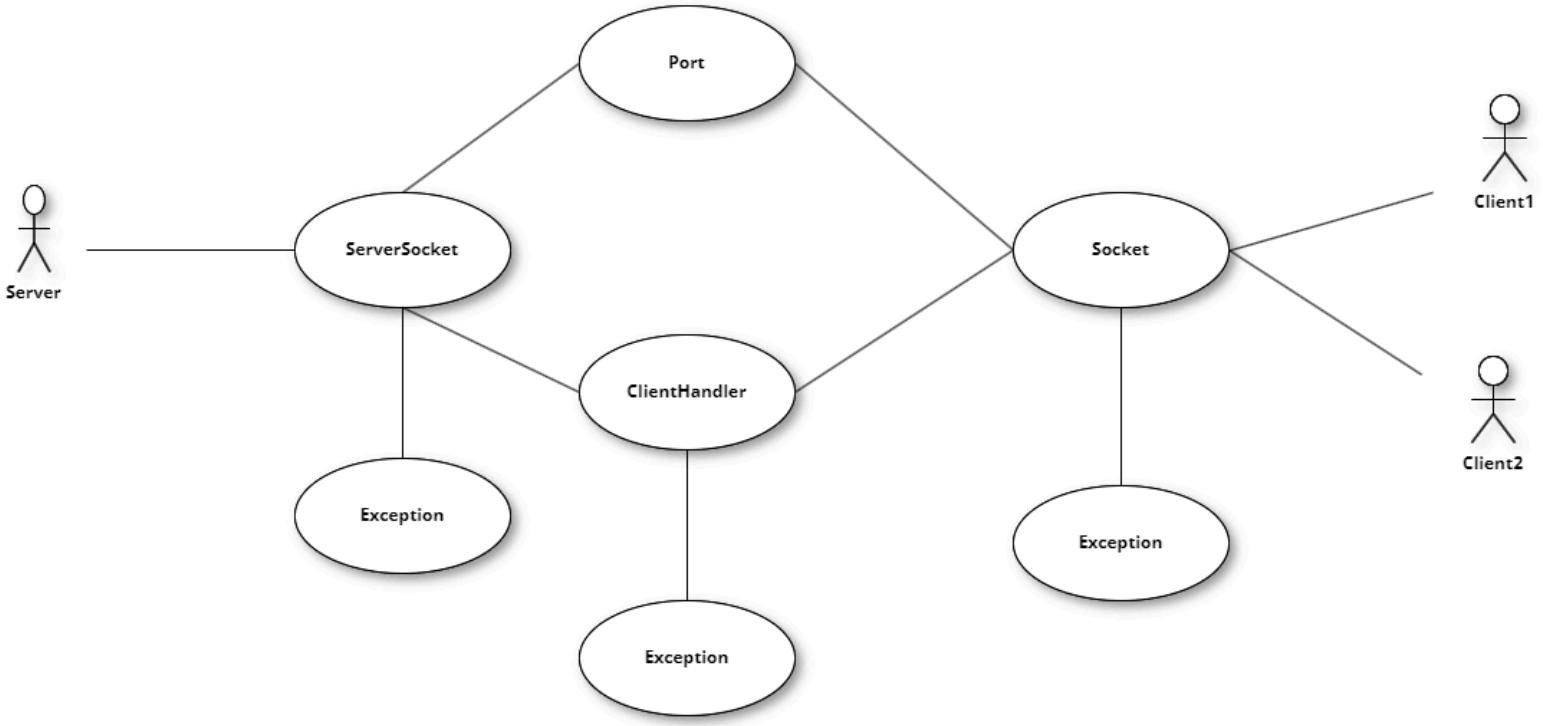
Use Case: Message from Client to Server



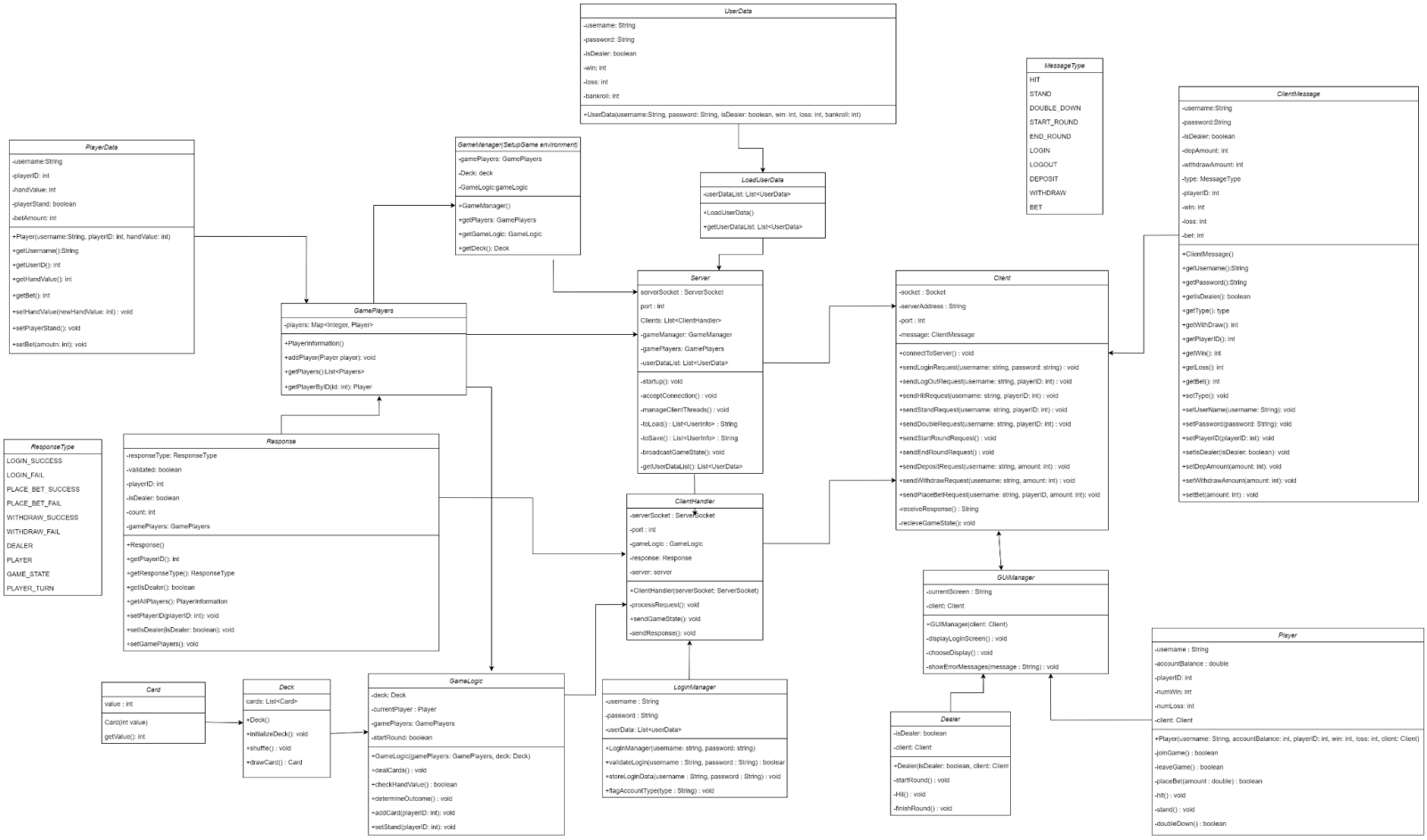
Use Case: Error Handling for Client Server Connection



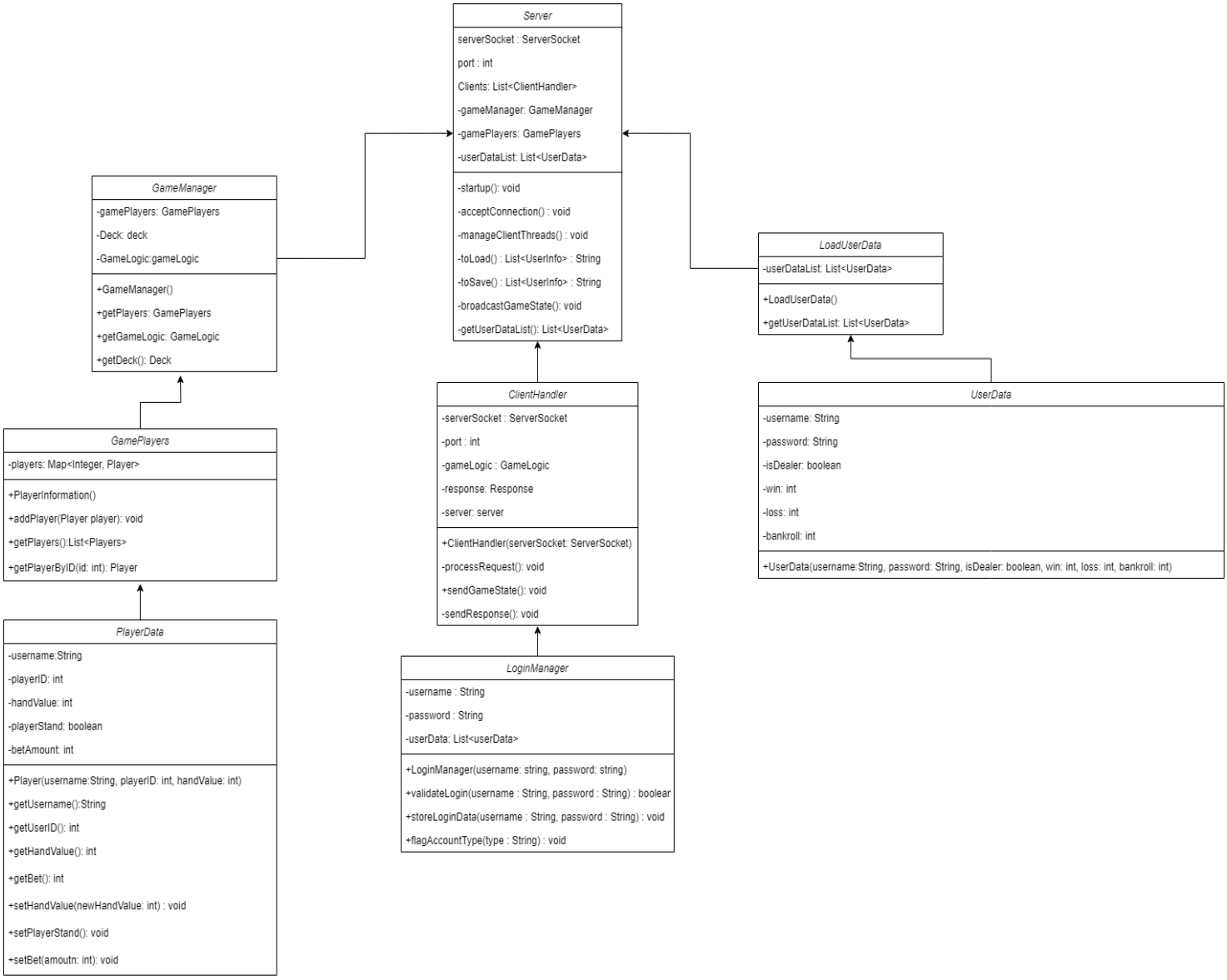
Use Case: Error Handling for Multithreaded Operation



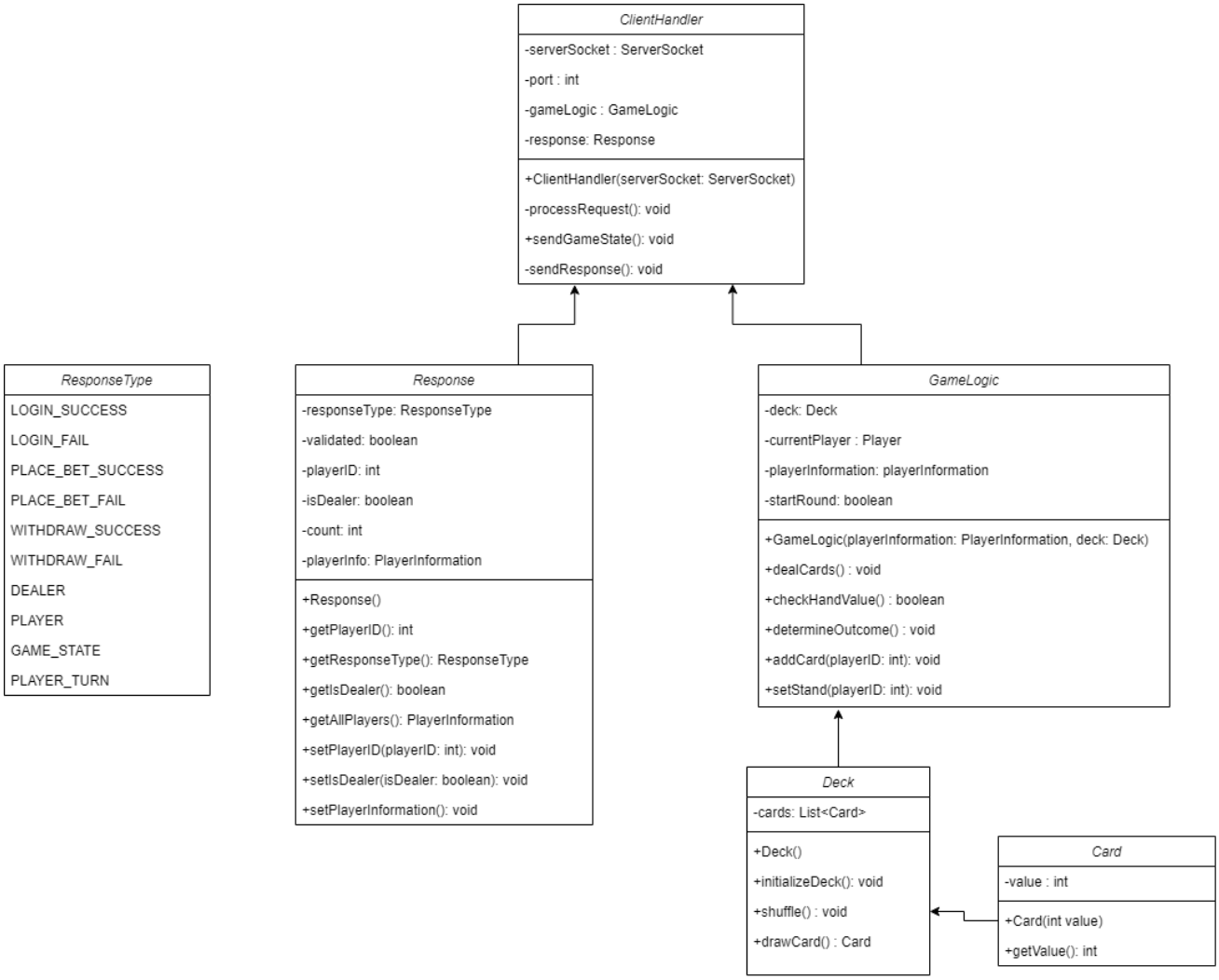
Class Diagram



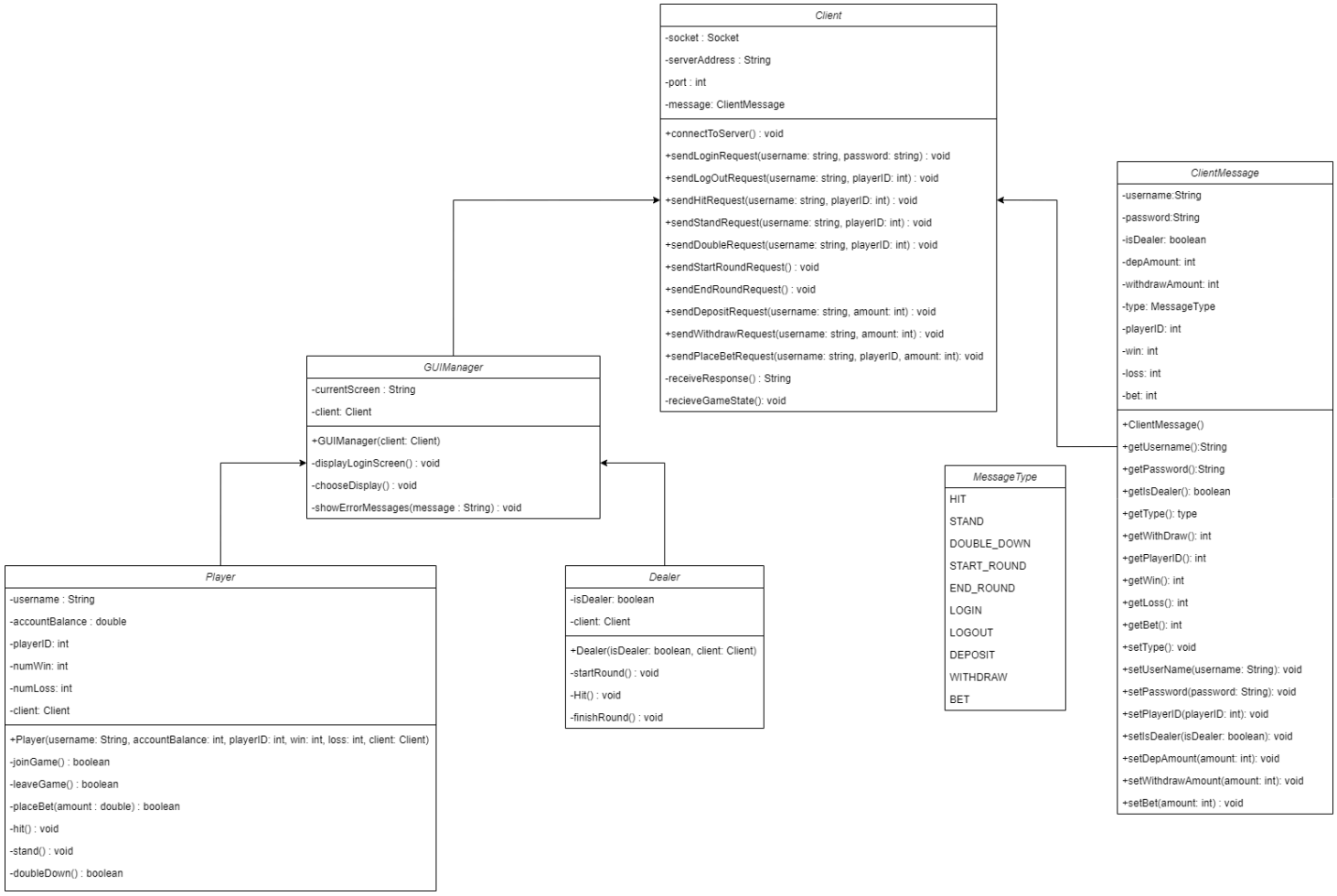
Server Side Class Diagram



Server Side Class Diagram(Client Handler)

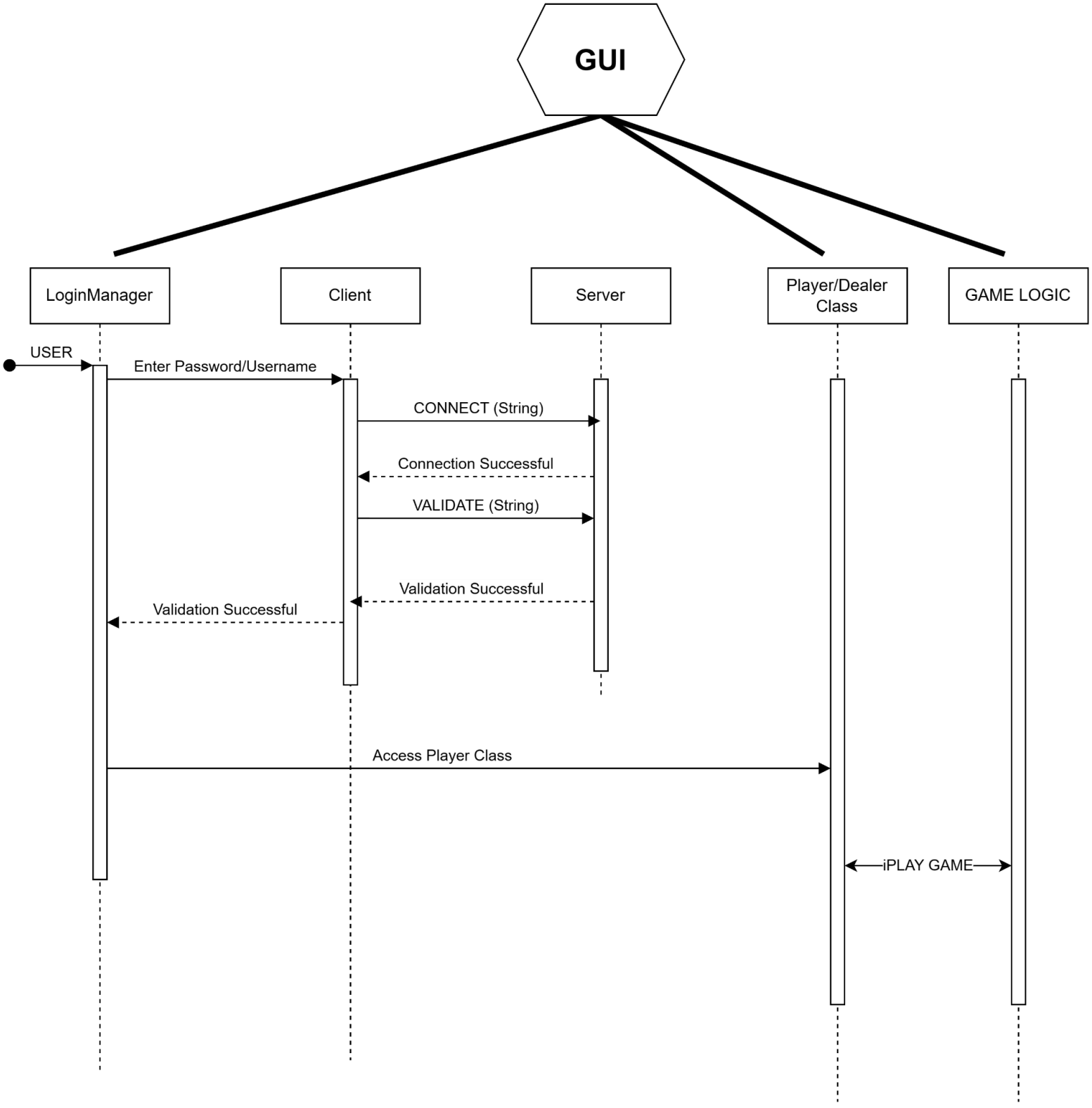


Client Side Class Diagram

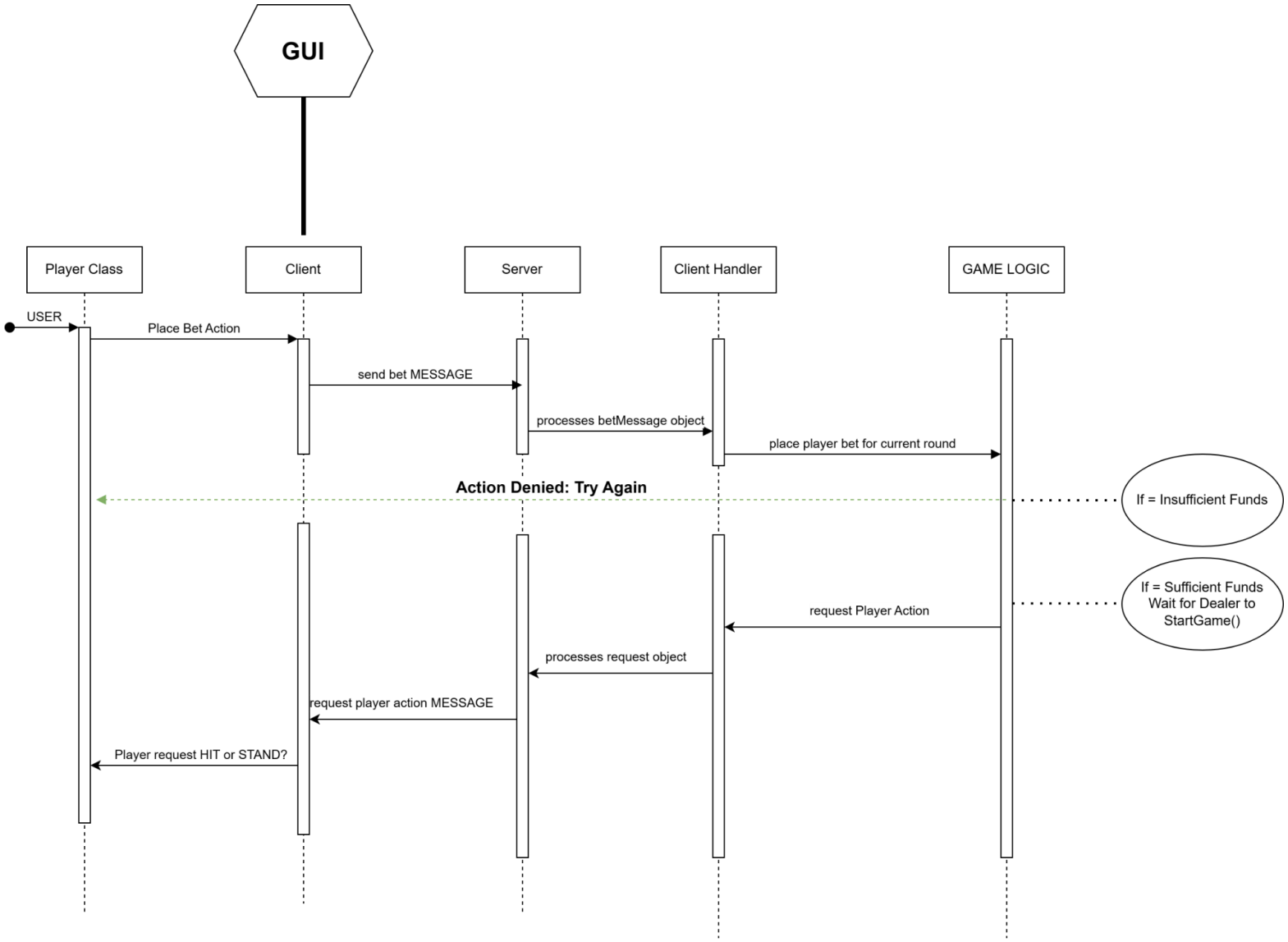


Sequence Diagrams

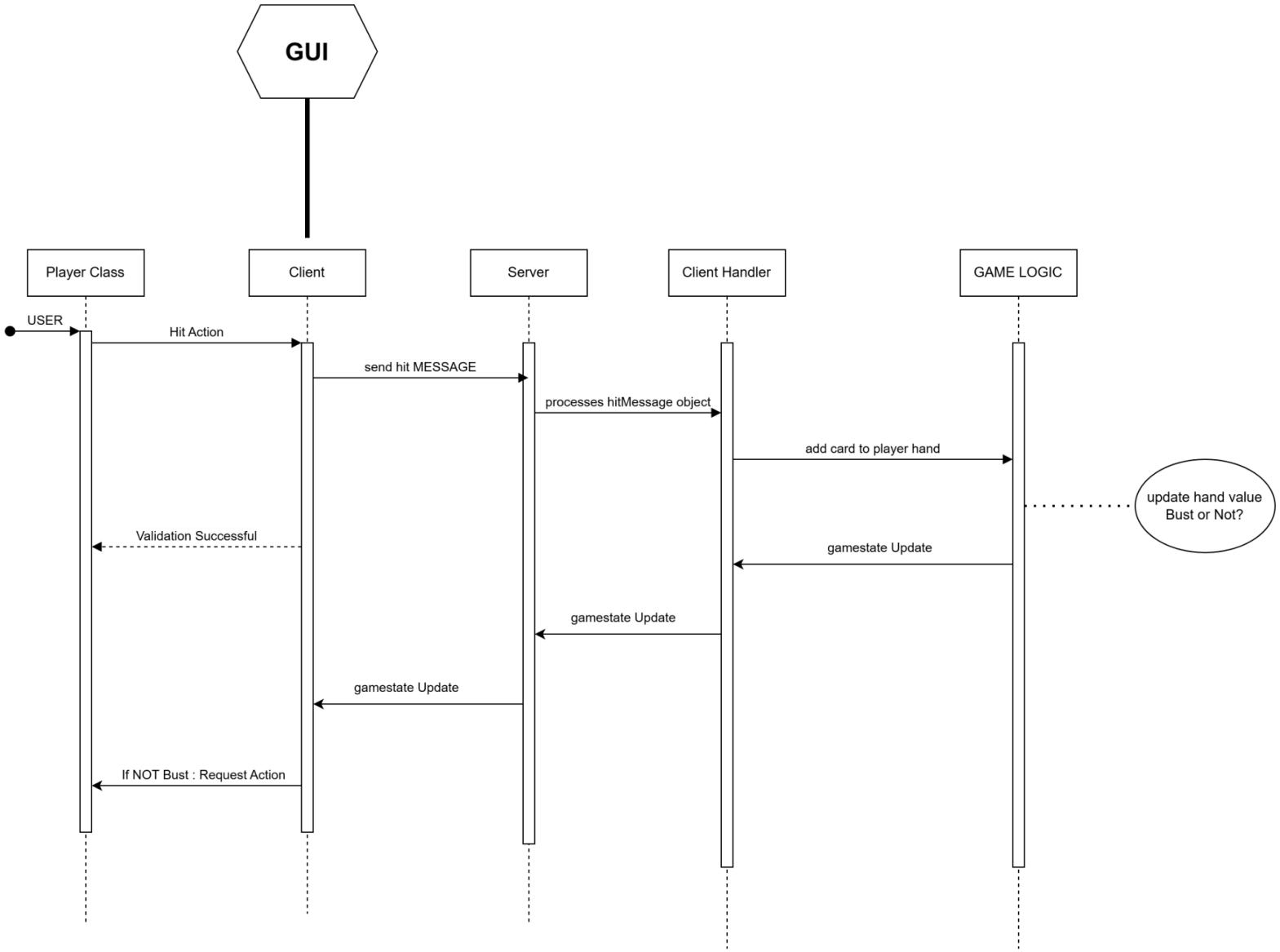
Use Case: User Login



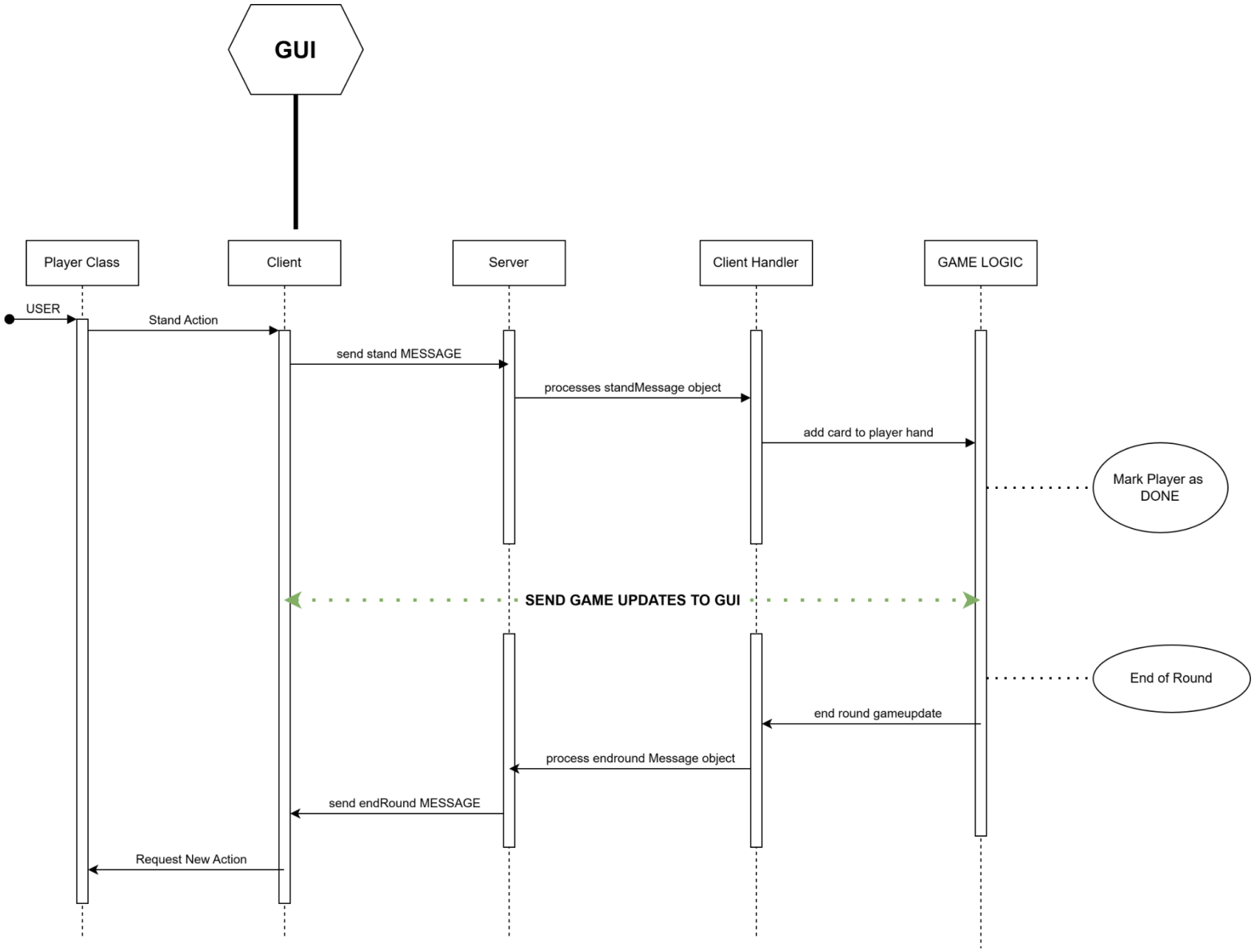
Use Case: Player Options → Place Bet



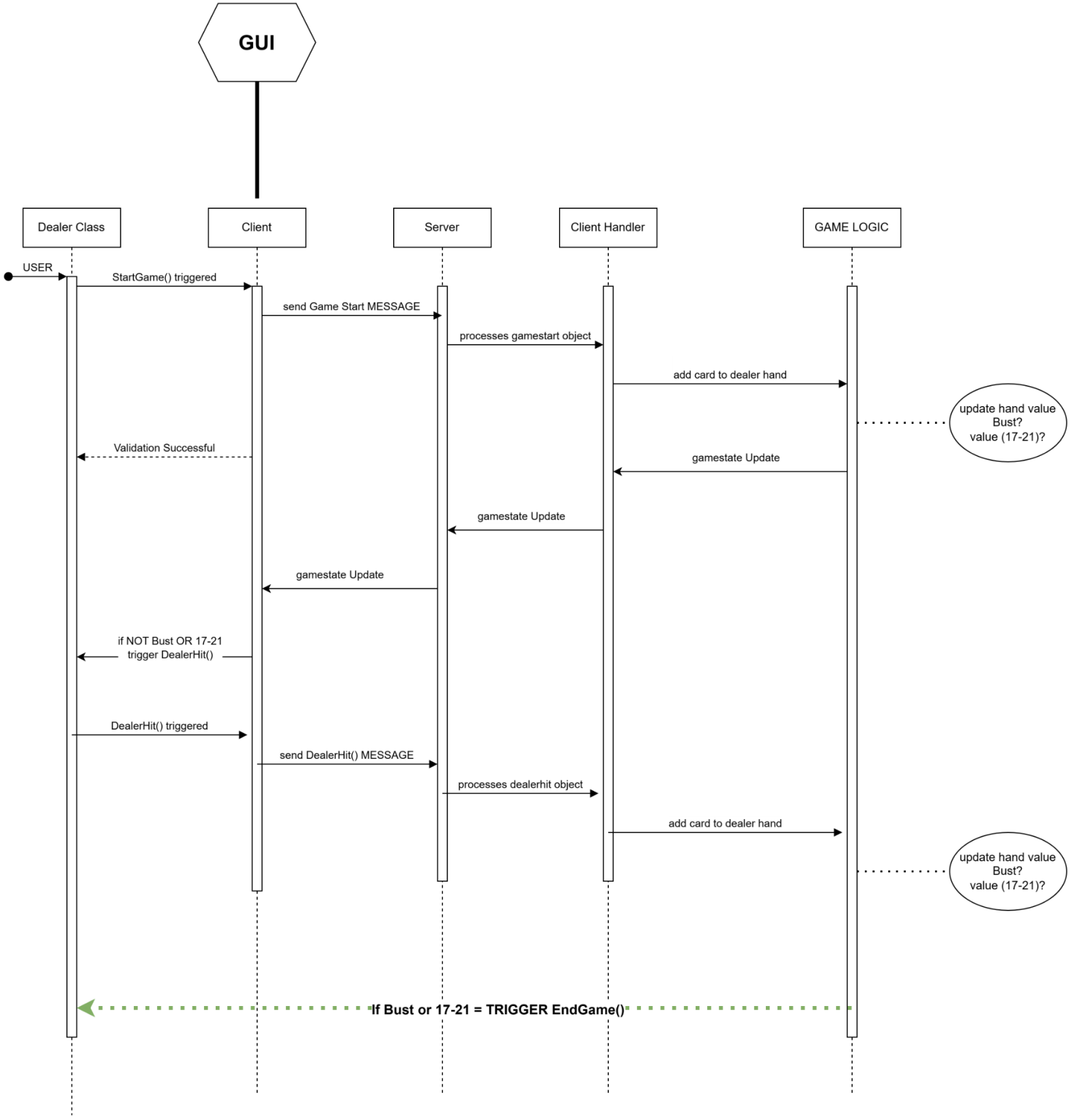
Use Case: Player Action → Hit



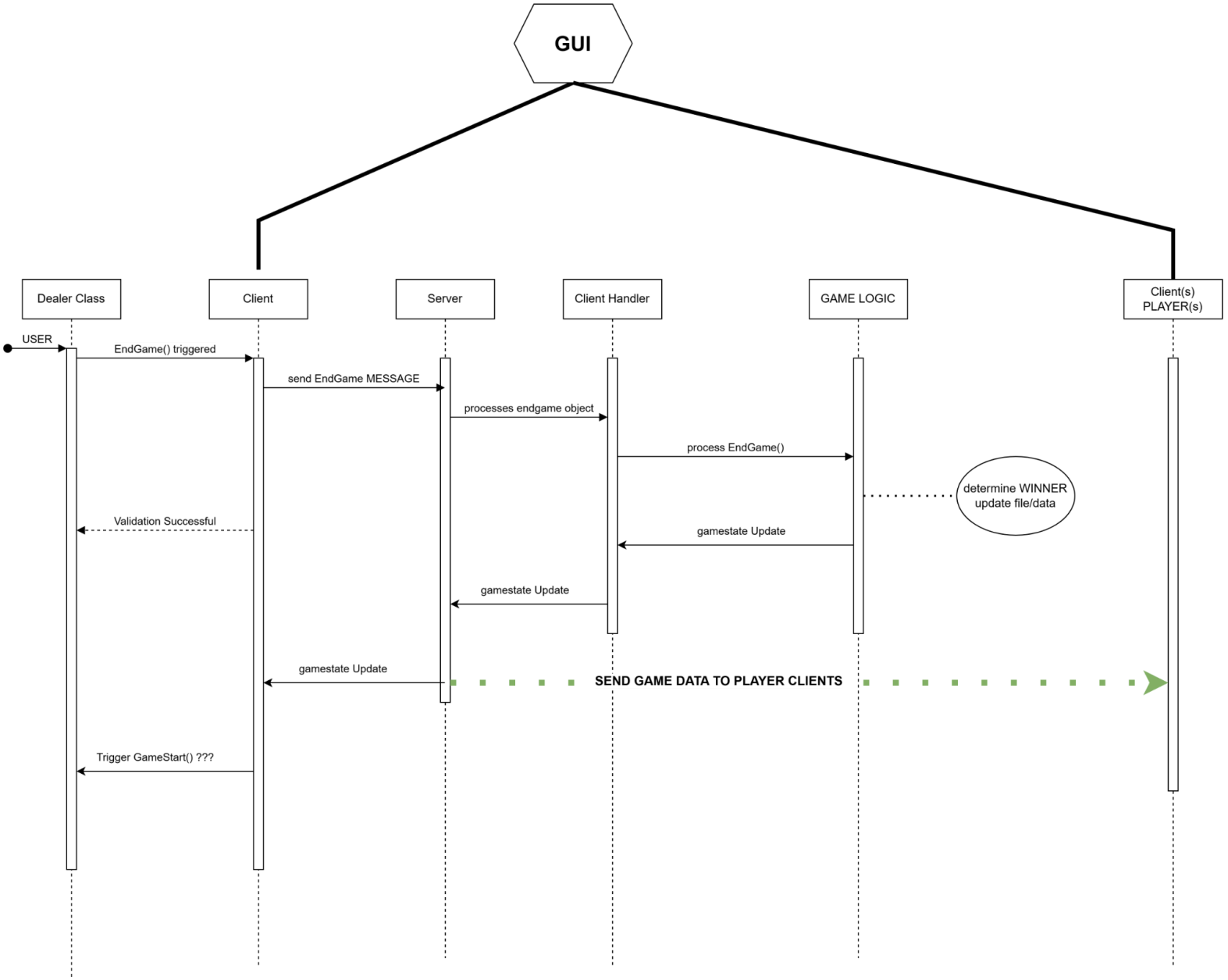
Use Case: Player Action → Options Stand



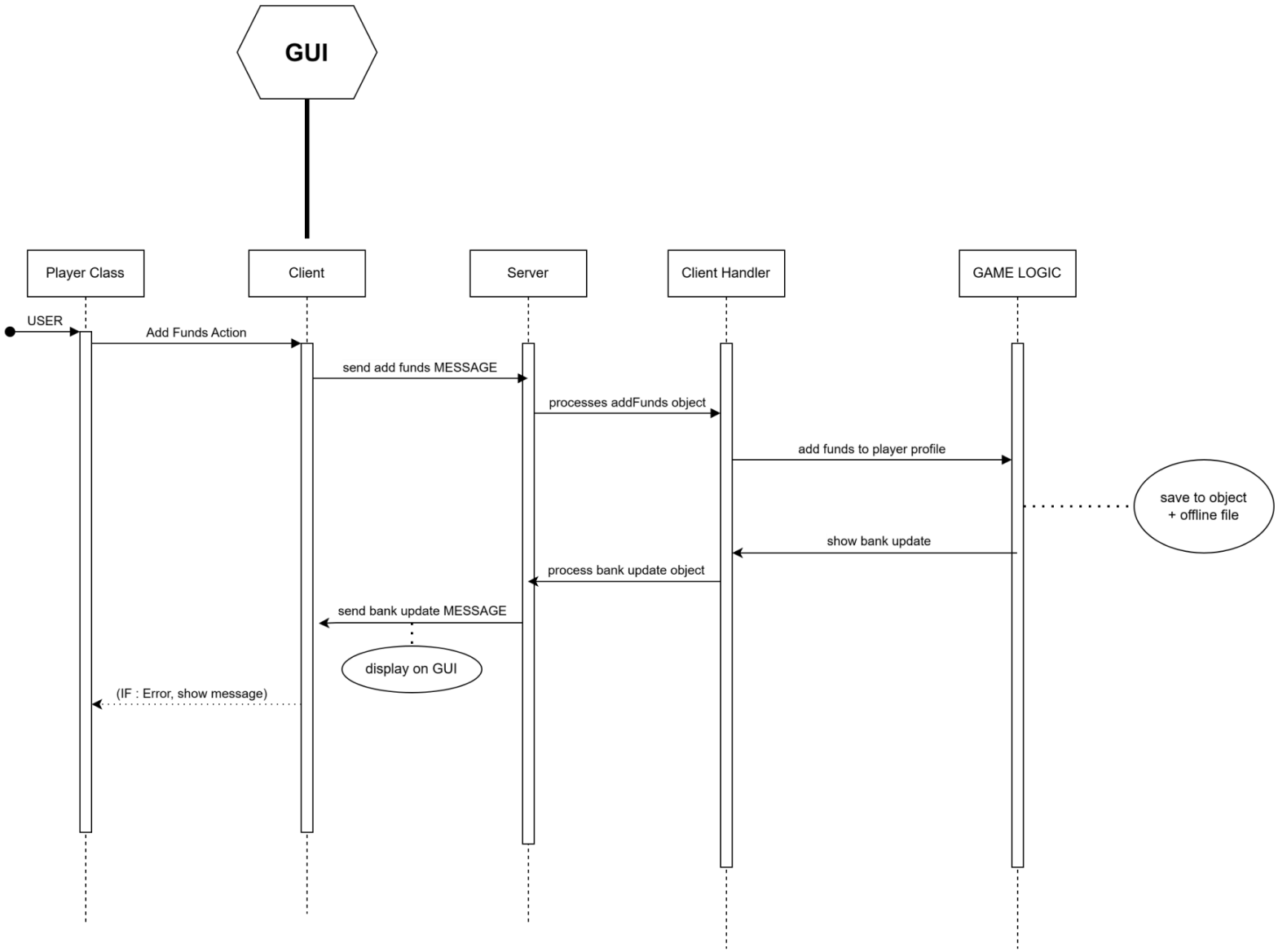
Use Case: Dealer Action → Start Game and Dealer Hit



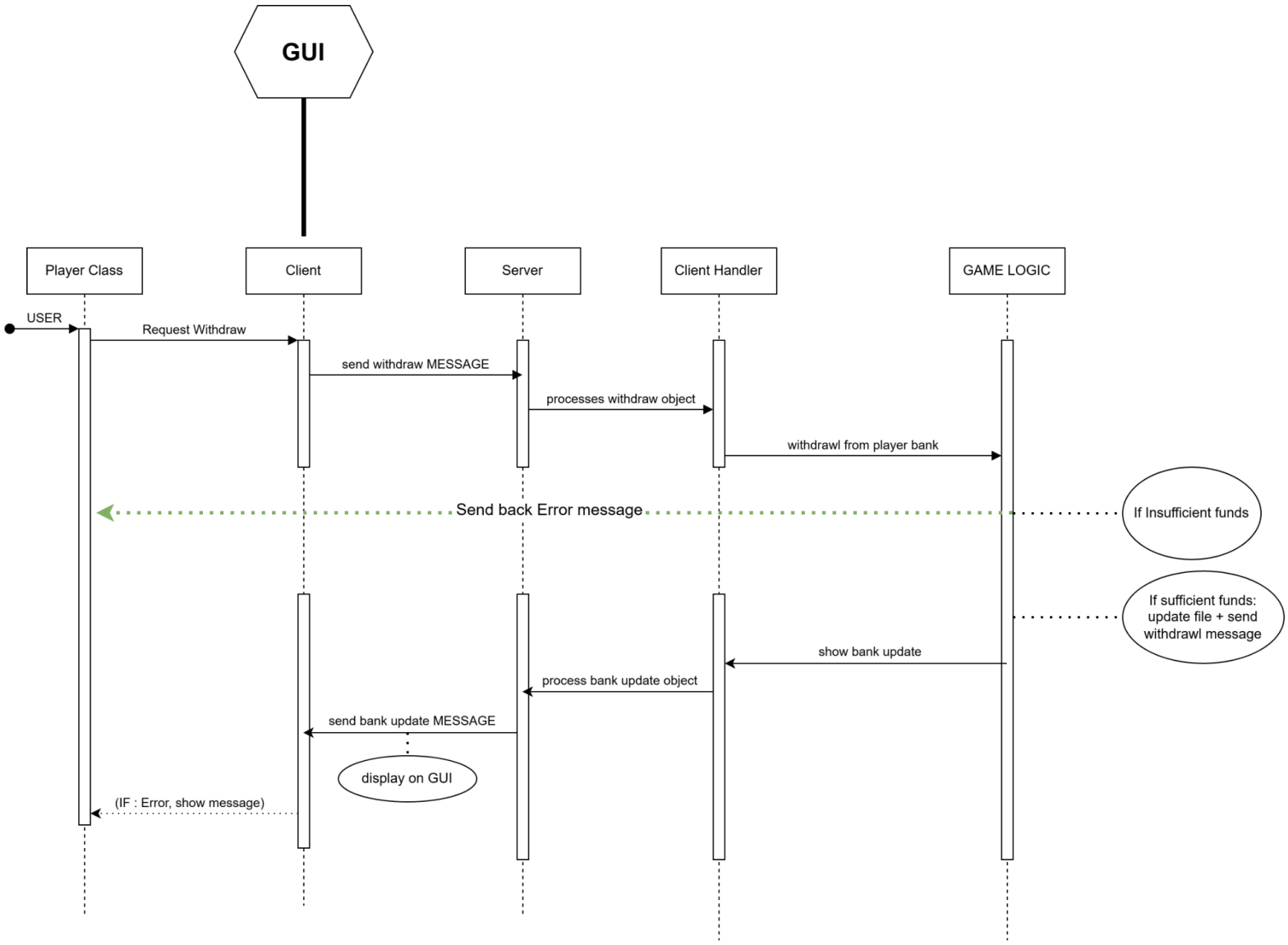
Use Case: Dealer Action → End Game



Use Case: Player Action → Add Funds



Use Case: Player Action → Withdraw Funds



Use Case: Player Action → Request History