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

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.
3.1.3.4 The Server should save Player data every time win/loss or bank status changes.

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.

3.1.4 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.

Use Case Specification Document

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.

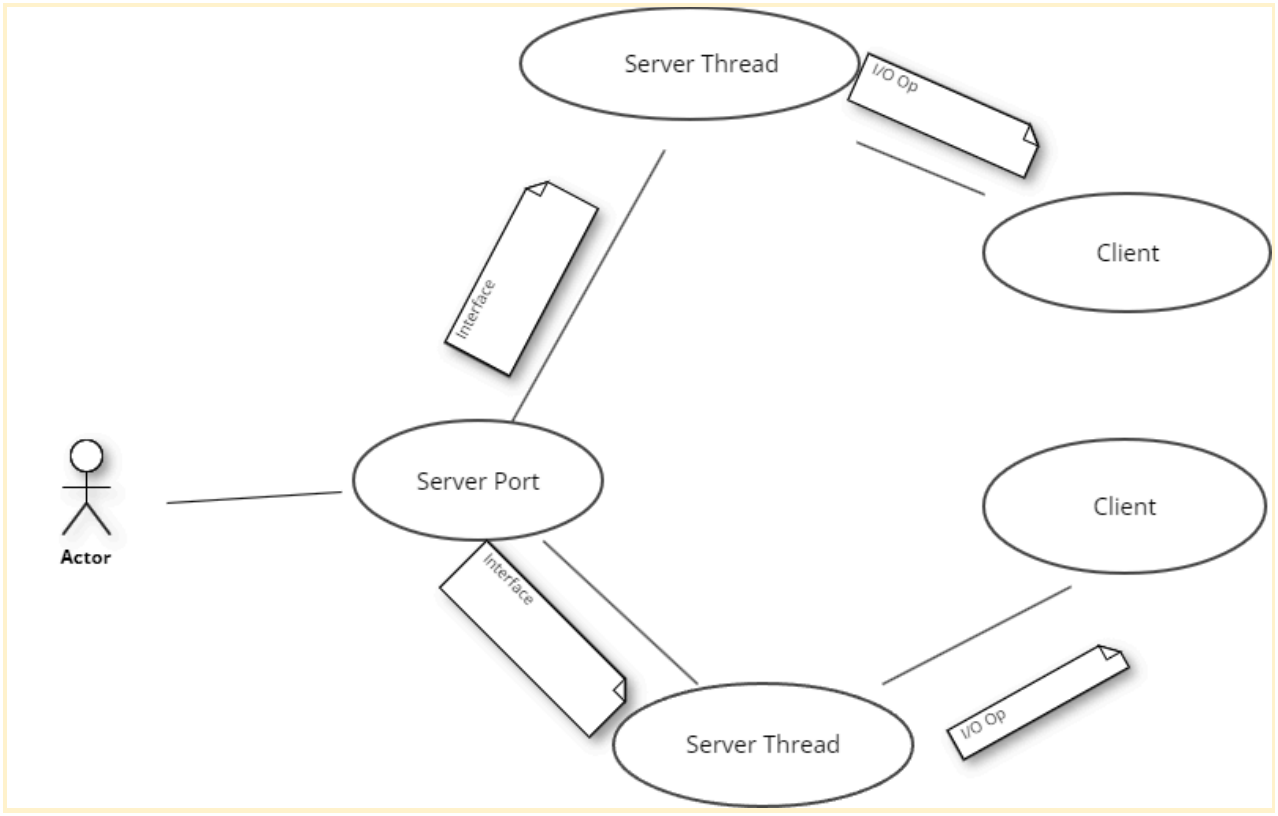
Use Case ID: 004
Use Case Name: Player Action
Relevant Requirements: 3.1.5.1, 3.1.7.6
Primary Actor: Player
Pre-conditions: Player is logged in and playing the round.
Post-conditions: Player is able to hit, stand, or double down.
Basic Flow or Main Scenario: Player selects the option to either hit or 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: 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.

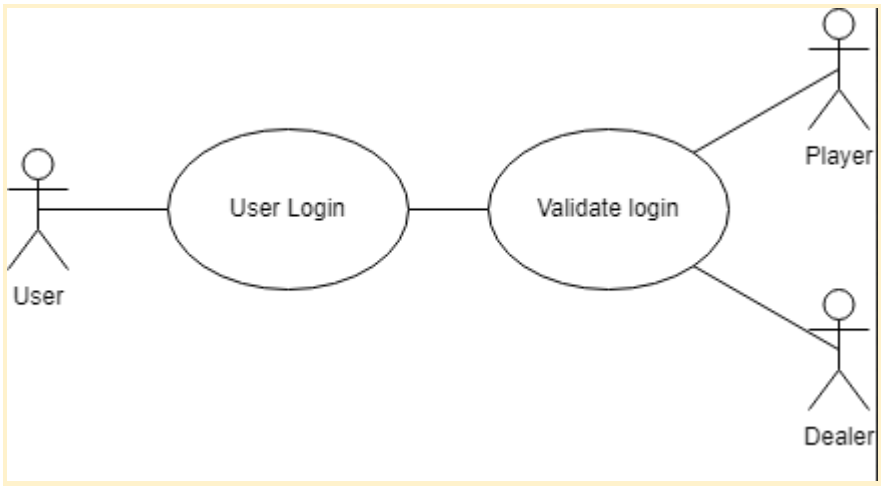
Use Case ID:006
Use Case Name: Server Client Interaction
Relevant Requirements: 3.1.2,3.1.3
Primary Actor: Systems Administrator
Pre-Conditions: The Computer System is on
Post-Conditions: A network connection is established with one port designated as a server. There can be multiple or zero clients connected 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 on-coming 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 Specification Document

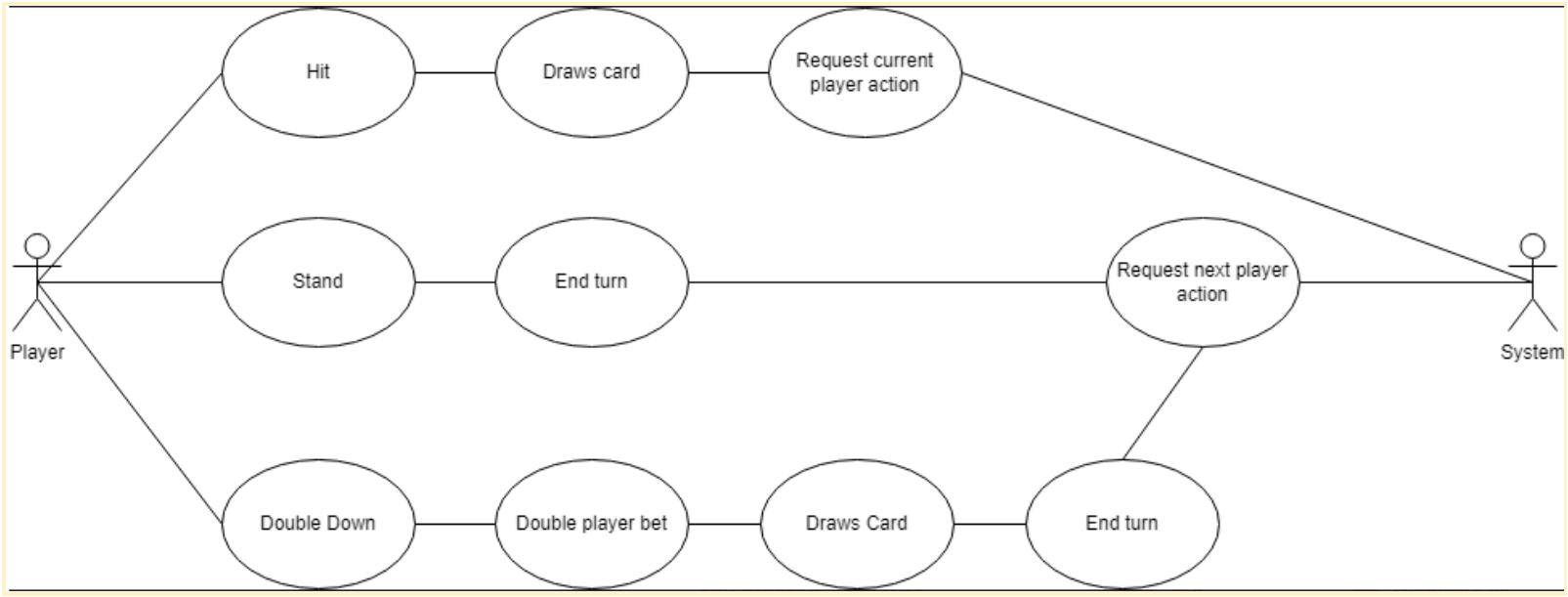
Use Case: Client Server



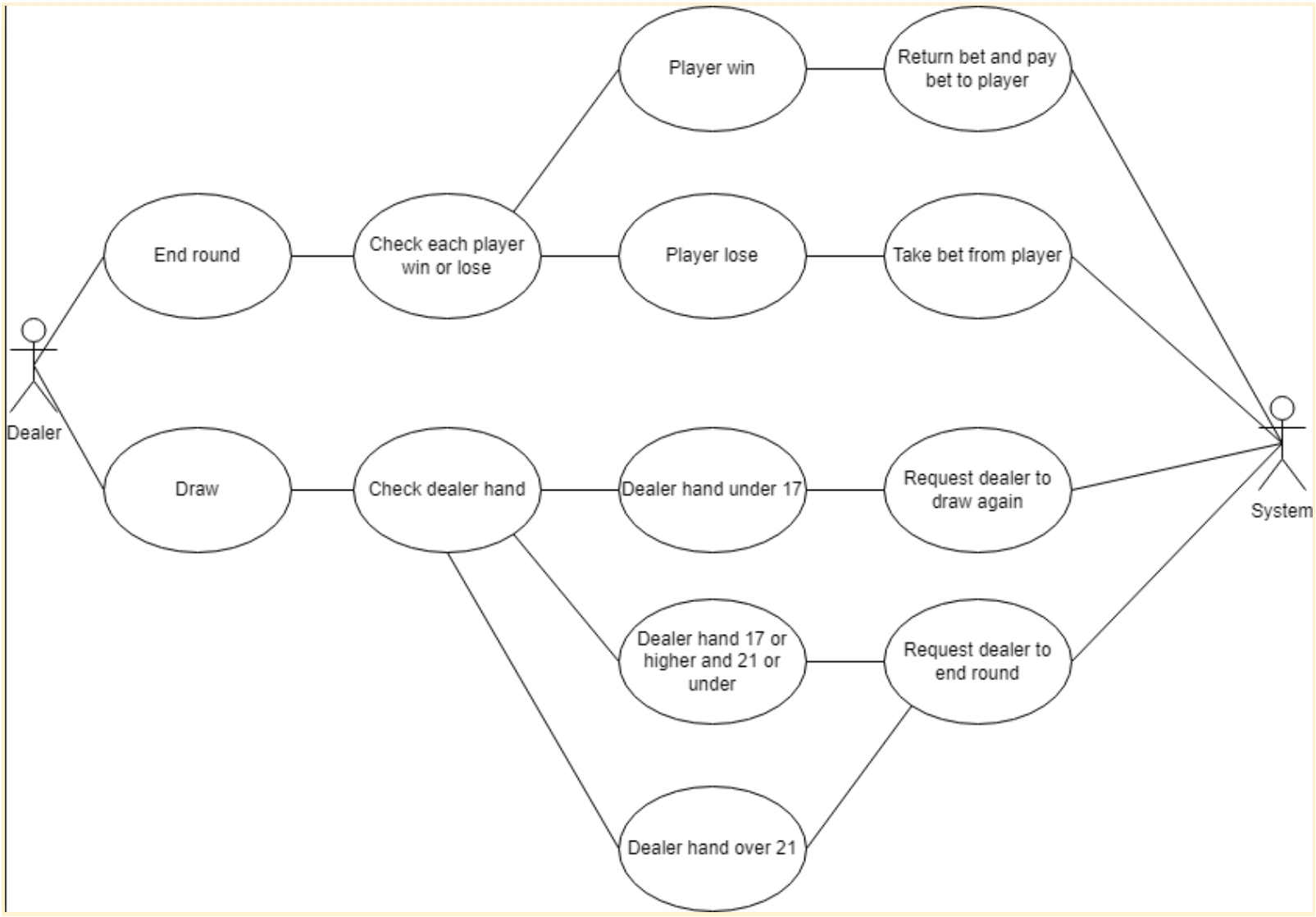
Use Case: User Login



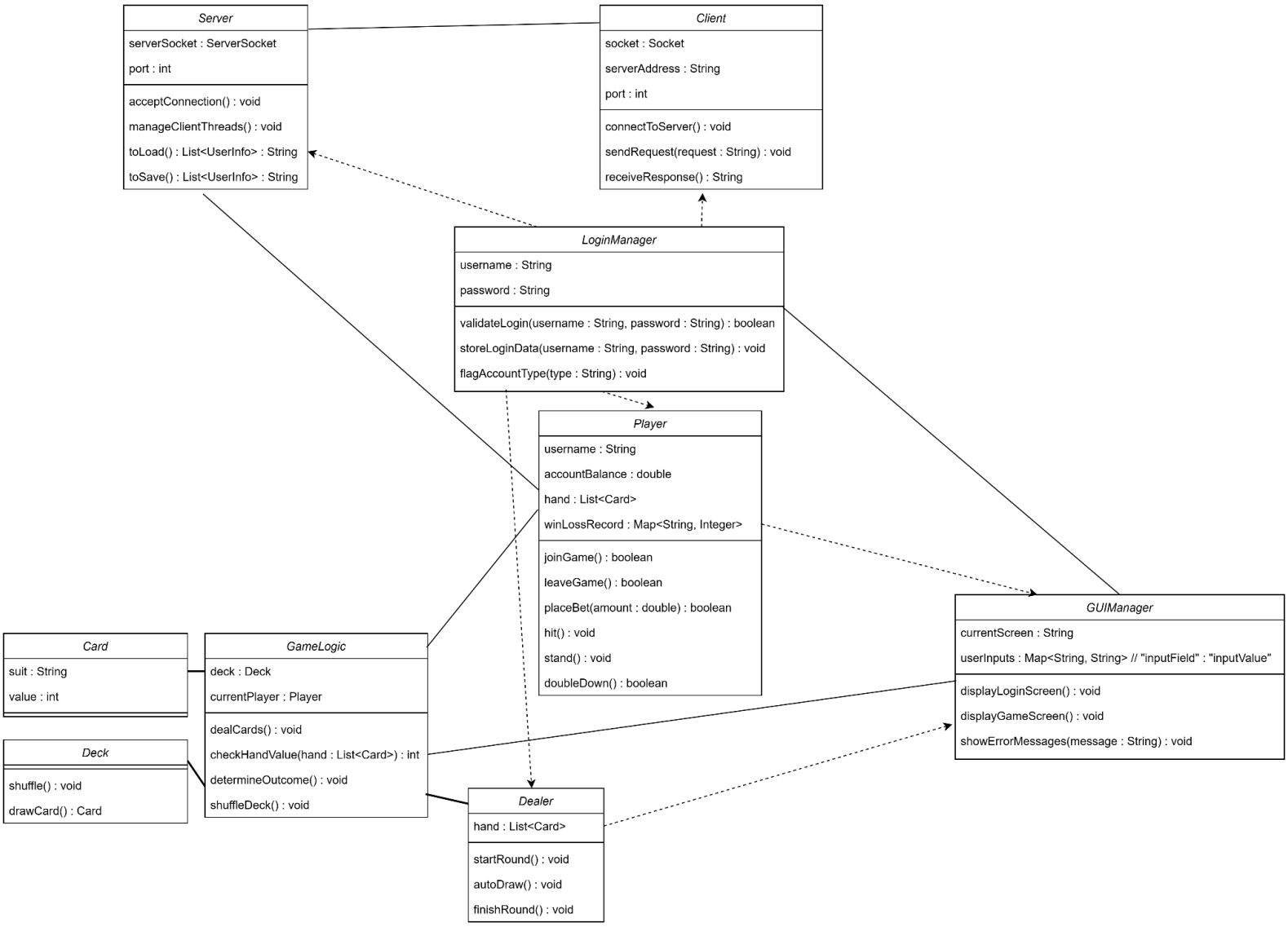
Use Case: Player Action



Use Case: Dealer action

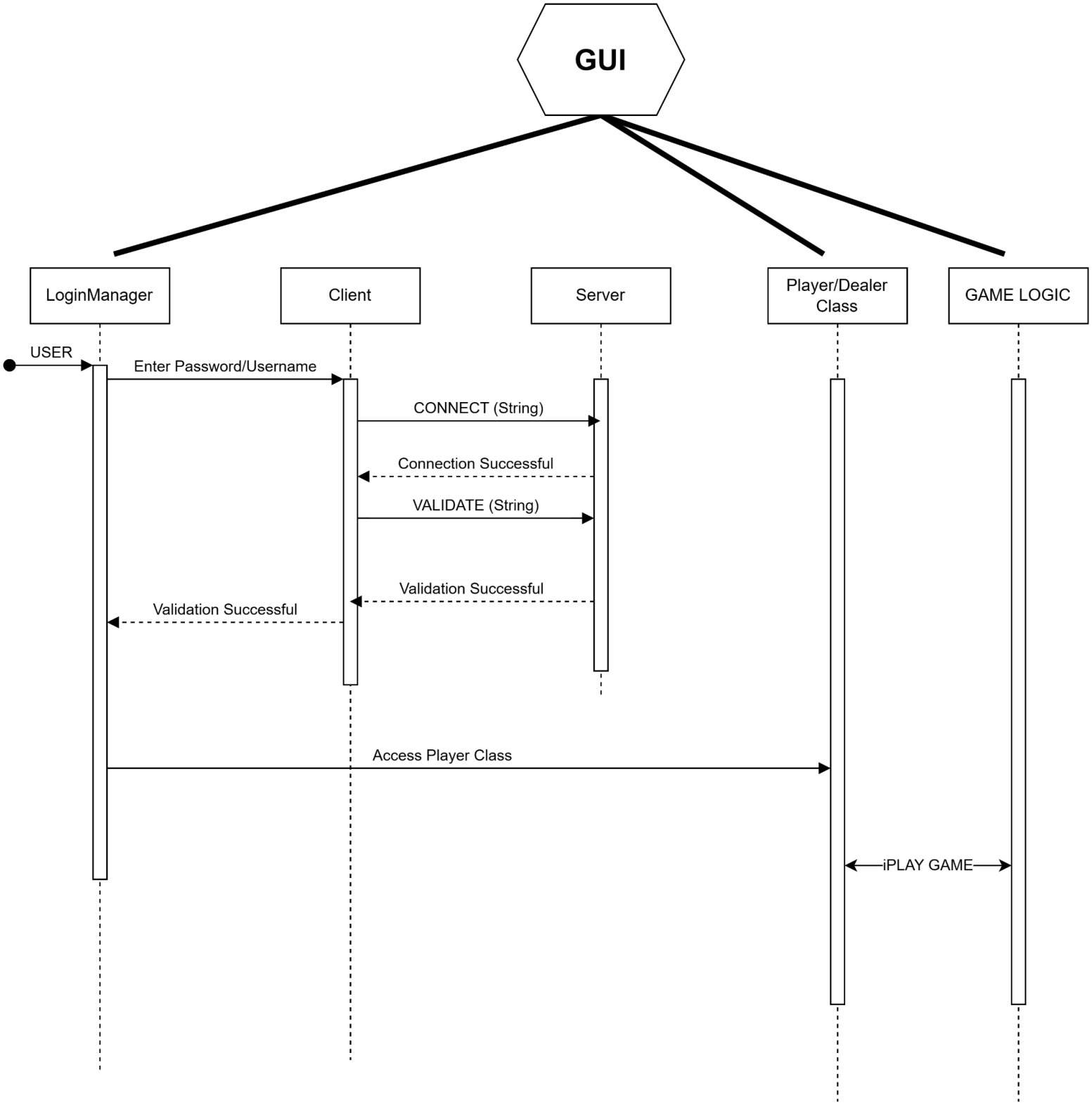


Class Diagram



Sequence Diagram

Use Case: User Login



Use Case: Player Options (Hit/Stand)