Group 2 - BlackJack

Software Requirements Specification

Revision History

Date	Revision	Description	Author
09/21/2023	1.0	Initial Version	Patrick, Syeda, Dequan, Shadab
09/24/2023	1.1	Added Table, Bank, Game, and Player module requirements	Patrick, Syeda, Dequan, Shadab
09/26/2023	1.1.1	1. Added App module and its requirements.	Shadab(1)
9/28/2023	1.2	1. Added constraints. 2. Added assumptions and dependencies 3. Added purpose and overview	Dequan(1), Patrick (1&2) , Syeda(3)
9/30/2023	1.3	1. Added product architecture. 2. Added product functionality. 3. Added product perspective 4. Added definitions	Dequan(1&2), Syeda(3), Patrick(4)
10/01/2023	1.3.1	1. Removed a requirement from the bank module. 2. Modified info in 2.2 and 2.3	Patrick(1), Dequan(2)
10/02/2023	1.4	1. Added common requirements. 2. Added and modified some of the existing module requirements.	Group(1), Shadab(2)
10/03/2023	1.5	1. Added internal and external interface requirements. 2. Added non functional requirements(security, privacy, environmental & performance). 3. Added and modified some non-functional requirements	Shadab(1&2), Patrick(3)
10/04/2023	1.6	1. Added use specification document. 2. Added class diagram for the entire system 3. Added join game sequence diagram. 4. Added player gameplay sequence diagram.	Dequan(1&2), Syeda(1&2), Patrick(1,3,4)
10/05/2023	1.7	1. Added menu use case diagram. 2. Added blackjack use case diagram. 3. Fixed some grammar errors, and amended out-of-date requirements. 4. Updated references 5. Renamed App module to Client-Server module and redid some of its requirements	Patrick(1&2), Group 2(3), Syeda(4), Shadab(5)
11/1/2023	1.8	1. Updated UML Class Diagram	Group 2
11/2/2023	1.9	1. Redid the module requirements to align with the new UML class diagram 2. Modified the product architecture to match the new UML class diagram 3. Modified the external and internal interface requirements to match the new UML class diagram 4. Modified the definitions to be more accurate	Shadab(1,2,3, 4)

		to the current design	

Table of Contents

1. Purpose.

1.1. Scope

This document will catalog the user, system, and hardware requirements for the BlackJack system.

1.2. Definitions, Acronyms, Abbreviations

Seat – A player at the table.

Table – A game lobby with 7 seats/spots available to occupy by a player.

Hit – When a player asks for a new card from the dealer.

Stand – When a player holds the total of their cards and ends their turn.

Bust – When the player or dealer's card total is greater than 21.

1.3. References

Use Case Specification Document – Contains detailed description of essential use cases.

Class Diagrams – Depicts the class structure and relationships with the Multiplayer BlackJack game.

Sequence Diagrams – Presents interactions and workflows between system components.

UML Use Case Diagram - Illustrates actors and use cases in the form of UML diagrams.

1.4. Overview

This document outlines the requirements for the development of a Multiplayer BlackJack Game. The game system is designed to support multiple players sharing a virtual table to play against an unbiased AI BlackJack Dealer. Players will also be able to deposit and withdraw money to and from their virtual BlackJack bank, allowing players to wager bets during their game sessions.

2. Overall Description

2.1. Product Perspective

A Multiplayer BlackJack game that aims to give players a fun and involved experience.

2.2. Product Architecture

- The system will be organized into 3 major abstract modules: the Client-Server, Game, and Bank Modules.
- The Client-Server Module consists of the GUI, Client, and Server class that work together to establish a client-to-server connection over TCP/IP protocol and handle the login and registration process for a user.
- The Game Module consists of the Table Controller class that possesses the table class that works with the dealer, deck, and card class to represent a blackjack table that will host up to 7 players in a blackjack game following the standard blackjack rules using a 52-card deck and handles players' minimum bets and how much they wager/win/lose.
- The Bank Module consists of the Bank class that works with the Player class to display necessary information if needed and keeps track of all the financial information of every player that is registered.

2.3. Product Functionality/Features

- The system will support multiplayer blackjack games that will include up to 1-7 players at individual virtual tables.
- Players will be allowed to create accounts and authenticate their logins via username and password giving players access to the system.
- A banking system will allow players to add funds to their accounts, set wagers, and cash out their accounts.
- The system will include an intuitive graphical user interface to ensure an enjoyable and easy-to-understand user experience.
- The system will update the interface in real-time to keep up with the current state of the game, including cards, scores, and bets.

2.4. Constraints

- The system shall be written in the Java programming language.
- The system shall be written with a widely supported Java version that is compatible with most Java Runtime Environments.
- The system must be written to be compatible with any modern operating system.
- To ensure wide-range compatibility with various hardware configurations, the system will not be built with any hardware-specific requirements.
- The system must be capable of handling 1-7 concurrent players per table.

2.5. Assumptions and Dependencies

- Players will be given the option to run the game in a single-player mode, however, a multiplayer mode is dependent on the client's access to the internet.
- The game will have various features for connection disruption exception handling, but a seamless performance of many of the system's features (banking, user profile, lobbies) will rely on stable internet bandwidth.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

- The game system should be able to maintain continuous operation, with necessary downtime scheduled for maintenance tasks.
- The system should handle errors gracefully, providing informative error messages to users in case of invalid inputs, failed transactions, or unexpected errors.

3.1.2. Client-Server Module Requirements:

- The graphical user interface (GUI) must be intuitive and user-friendly.
- The GUI must ensure a seamless transition between different pages.
- A "Back" button should be provided on relevant pages to allow users to navigate to the previous page.
- The GUI should facilitate the server-to-client connection.
- The application should handle the user login process.
- The server must maintain uninterrupted operation, ensuring 24/7 availability.
- Players must be able to authenticate using a username (6 to 20 alphanumeric characters in length) and password (8 to 15 alphanumeric characters in length) to gain access to the server.
- Players must be provided the option to change their display name, password, and the option to delete their account.
- The application should establish client-to-server communication over a TCP/IP protocol.

3.1.3. Game Module Requirements:

- The deck used must be in a standard 52-card format and the game follows standards Blackjack rules.
- Cards dealt to players must be randomized to prevent bias.
- Participating players must be required to hold a balance in their account that passes a minimum threshold in order to play the game.
- Blackjack tables must be able to support one to seven players simultaneously.
- Players will be able to see other instances of concurring Blackjack games on the table and are able to view other player’s cards.
- When new players want to join, if there are no open slots at any tables, a new table must be created.
- Players must be able to view their own cards.
- Players must have the option to hit and stand.
- The player's balance should be adjusted after each blackjack round at the table.

3.1.4. Bank Module Requirements:

- Players are able to add and withdraw funds from their current balance.
- Players can check their current balance.
- Players will have access to the personal information of their account, including wins, losses, past bets, etc.

3.2. External Interface Requirements

The external interface has different pages that interact with each other to create a user-friendly and functional interface for the software application. Each requirement specifies a clear action that the user can perform on each page, ensuring that the user's interactions with the system are well-defined and intuitive. The pages are written in a sequential order to show how the user experience would be.

Register or Login Page

- Users have two options to choose from. Logging in or Registering.
- Users must be able to access a registration page to create a new account if they're new.
- Users should be able to access a log-in page to authenticate themselves if they're an existing user.

Main Page

- The main page must provide users with three options: "Join a Table," "Bank," and "Settings."

Join Table Page

- Selecting the "Join a Table" option from the main page must navigate the user to a table in order to play a Blackjack game.

Bank Page

- The "Bank" option from the main page should lead to the bank page.
- On the bank page, users have the option to check their winnings and losses and prior transaction history.
- Players must also have the option to select deposit or withdraw which takes them to the balance page.

Balance Page

- The balance page must provide users with the functionality to deposit money, check their current balance, and initiate withdrawals.

Settings Page

- The user profile allows users to see information about their BlackJack game history, including wins, losses, etc.
- The Profile also provides options for the user to modify information about their account, such as their display name and password.

3.3. Internal Interface Requirements

The internal interface requirements specify how the pages and modules within the application should interact and process data, ensuring that data flows correctly and user actions trigger the appropriate processes:

Interaction between Pages and Modules

- The Register or Login Page must interact with the Player and Server class for user authentication and registration.
- The Main Page must provide options to access the Join Table Page, Bank Page, and Settings Page, with each option initiating the corresponding module.

Player Module Interaction

- The Player class should work with the Server class to handle user authentication and store user data, including usernames, passwords, display names, and balances.

- Upon successful login, the Player class should provide the user's data to other classes as needed.

Bank Module Interaction

- The Bank Page should utilize the Bank class to enable users to deposit money, check balances, and withdraw funds.
- The Bank Page must interact with the Player class to display their winnings and losses and prior transaction history.
- The Bank class must interact with the Player class to retrieve and update user account details, such as their current balance.

Table Module Interaction

- The Table class should support multiple players and manage the progress of blackjack games.
- The Join Table Page should interact with the Table class to allow users to join and participate in active games.
- The Table class also keeps track of scores, bets, and card distributions for each player.

App Module Interaction

- The GUI class should display user interface elements for each page, including buttons for navigation and input fields for data entry.
- All pages should interact with the GUI class and other necessary classes to display information, collect user input, and navigate between pages seamlessly.

Settings Module Interaction

- The Settings Page should allow users to change their display name and password, with updates managed by the Player class.
- Changes made in the Settings Page should reflect in the user's account info managed by the Player class.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

- User authentication data, including passwords, must be securely stored and transmitted using industry-standard encryption protocols to protect user privacy.
- All user interactions with the system, including login, financial transactions, and card details editing, must be logged for auditing purposes.
- The system must implement measures to prevent unauthorized access, including brute force protection and account lockout mechanisms.
- User personal data, including usernames, passwords, and card details, should be treated with strict confidentiality and not shared with unauthorized parties.
- Users should have the option to delete their accounts and associated data permanently if they choose to do so.
- The system must be able to deal with various levels of error handling to avoid exposing sensitive system information.

4.2. Environmental Requirements

- The application must be platform-independent and capable of running on various operating systems to accommodate a broad user base.

4.3. Performance Requirements

- The system should aim for low latency in processing user actions to provide a responsive and enjoyable gaming experience.
- Response times for key operations, such as logging in, joining a table, and processing financial transactions, should be within acceptable limits (e.g., under 2 seconds) to maintain user engagement.
- The system should be capable of supporting concurrent gameplay by multiple users without degradation in performance, ensuring smooth and fair gameplay.
- Load testing should be performed to determine the system's capacity for handling simultaneous connections, with the system designed to accommodate an unlimited amount of users simultaneously.
- Network communication should be optimized to minimize data transfer overhead, ensuring efficient utilization of available bandwidth.

1. Use Case Specification Document

Use Case ID: UC001
Use Case Name: Login/Logout
Relevant Requirements: Valid username and password.
Primary Actor: User
Pre-conditions:

1. The system is operational and accessible.
2. The user registration is complete and authentic usernames and passwords have been saved.

Post-conditions:

1. After successfully logging in, the user has access to all the features.
2. Logging out brings back the user to the starting position.

Basic Flow or Main Scenario:

1. The user opens the application.
2. A login screen is displayed by the system.
3. The user enters their login information.
4. The system verifies the credentials.
    a. The user is successfully signed in if the username and password match an entry in the database.
    b. The system displays an error notice if the credentials are incorrect.
5. The user has access to content or features that are limited if they are logging in.
6. The user decides to logout.
7. The system locks the user out and reloads the login page.

Extensions or Alternate Flows:

1. Invalid credentials
    - The system displays an error notice and allows the User to re-enter the information if the user enters an invalid credential.

Exceptions:

1. System Unavailability
    - The user cannot log in if the system is down or unavailable. It shows an error message.

Related Use Cases:

1. UC002: Access Bank Information
2. UC003: Playing the game

Use Case ID: UC002
Use Case Name: Access Bank Information
Relevant Requirements: A valid bank account with sufficient funds.
Primary Actor: Player
Pre-conditions:

1. Player profile exists in program database
2. The player is logged into the corresponding player account

Post-conditions:

1. The player has access to their account balance and transaction history.
2. Options to deposit and withdraw funds are presented.

Basic Flow or Main Scenario:

1. The player is at the program's main menu.
2. Player requests to view their fund information.
3. System pulls bank account information from the player object.
4. The system displays the bank account balance, along with options to deposit and withdraw.

Extensions or Alternate Flows:

1. Player requests to deposit funds.

    a. System prompts the user to enter the card information or to choose a previously used payment method.
    b. The user chooses a payment method.
    c. The user selects the fund amount to transfer.
    d. The system transfers funds from the payment method into the player's account information.
2. Player requests to withdraw funds.
    a. System prompts the user to select which payment method to transfer funds into.
    b. User selects method.
    c. The system transfers funds from player account information into a selected transfer method.

Exceptions:

Related Use Cases:

1. UC001: Login/Logout
2. UC003: Playing the game

Use Case ID: UC003
Use Case Name: Playing the game
Relevant Requirements: All players must have sufficient funds to play a game.
Primary Actor: Player, Dealer
Pre-conditions:

1. An authenticated player account with sufficient funds is ready to play a game.

Post-conditions:

1. The player either wins or loses their wager and a new round begins.

Basic Flow or Main Scenario:

1. The player(s) sets a wager for the round.
2. The dealer deals out cards.
3. Each player's hand is summed. Those with sums exceeding 21 "bust" and lose the round. Those still in play will have their sums compared to the sum of the dealer's hand. Players with a higher sum than the dealer's are declared winners, those with sums lower than the dealer, lose.
4. Losers will lose their wager, and winners are rewarded with winnings.
5. The next round is ready to begin but the system verifies that each player has sufficient funds before it starts.
6. The process repeats

Extensions or Alternate Flows:

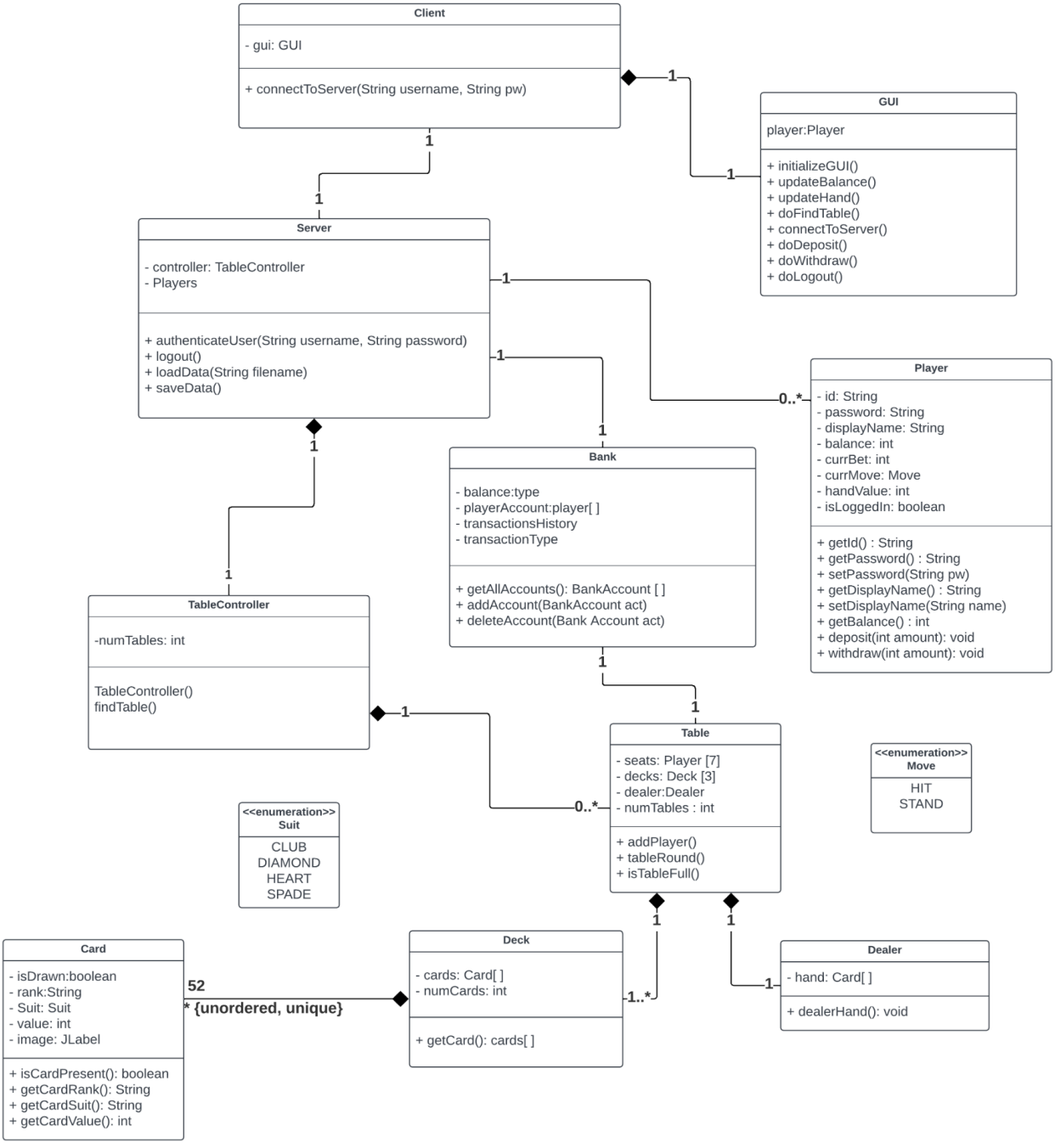
Exceptions:

1. The player is attempting to continue playing another round while having insufficient funds.

Related Use Cases:

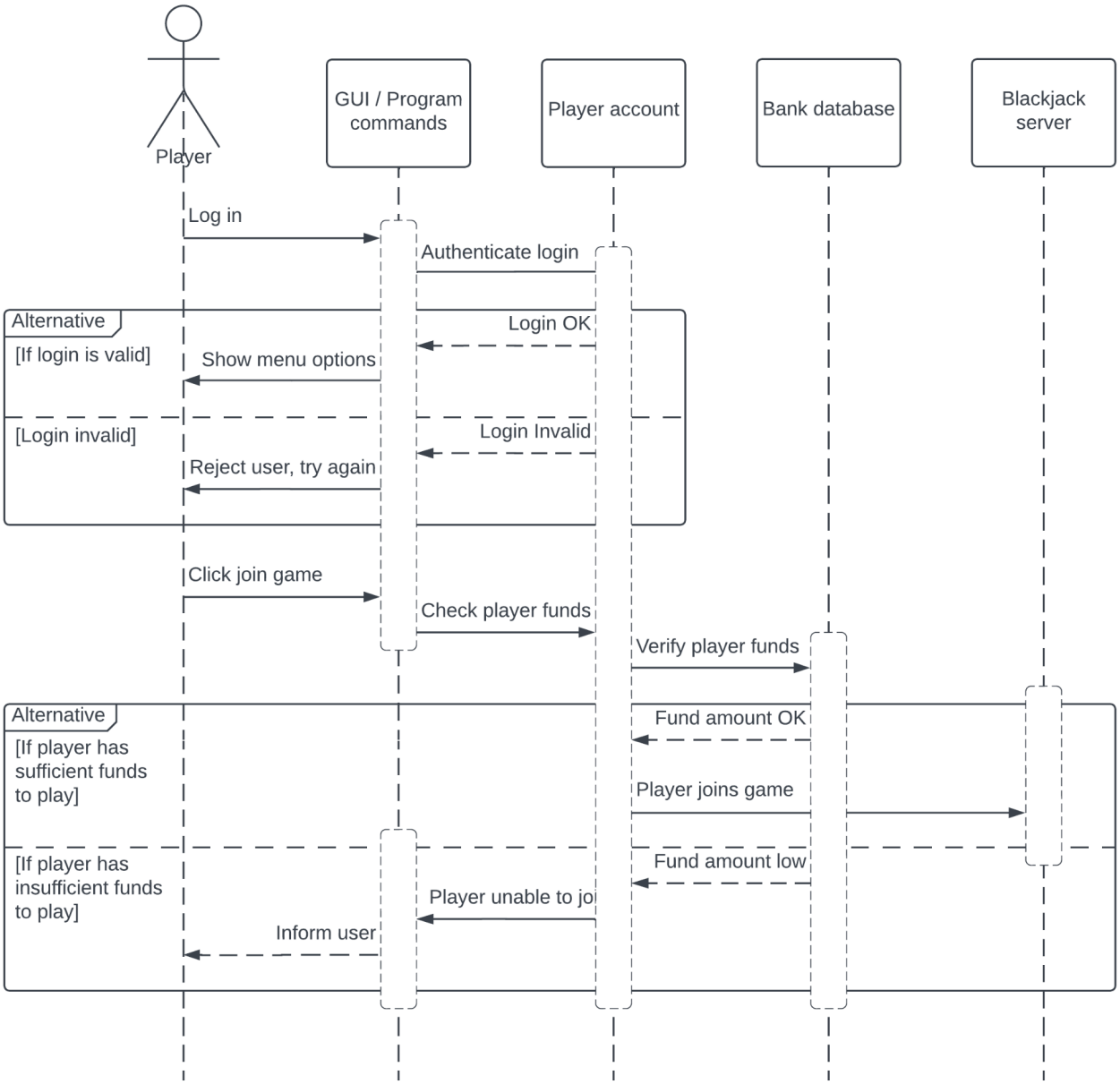
1. USC001 Login/Logout
2. USC002 Access Bank information

2. Class Diagram Document

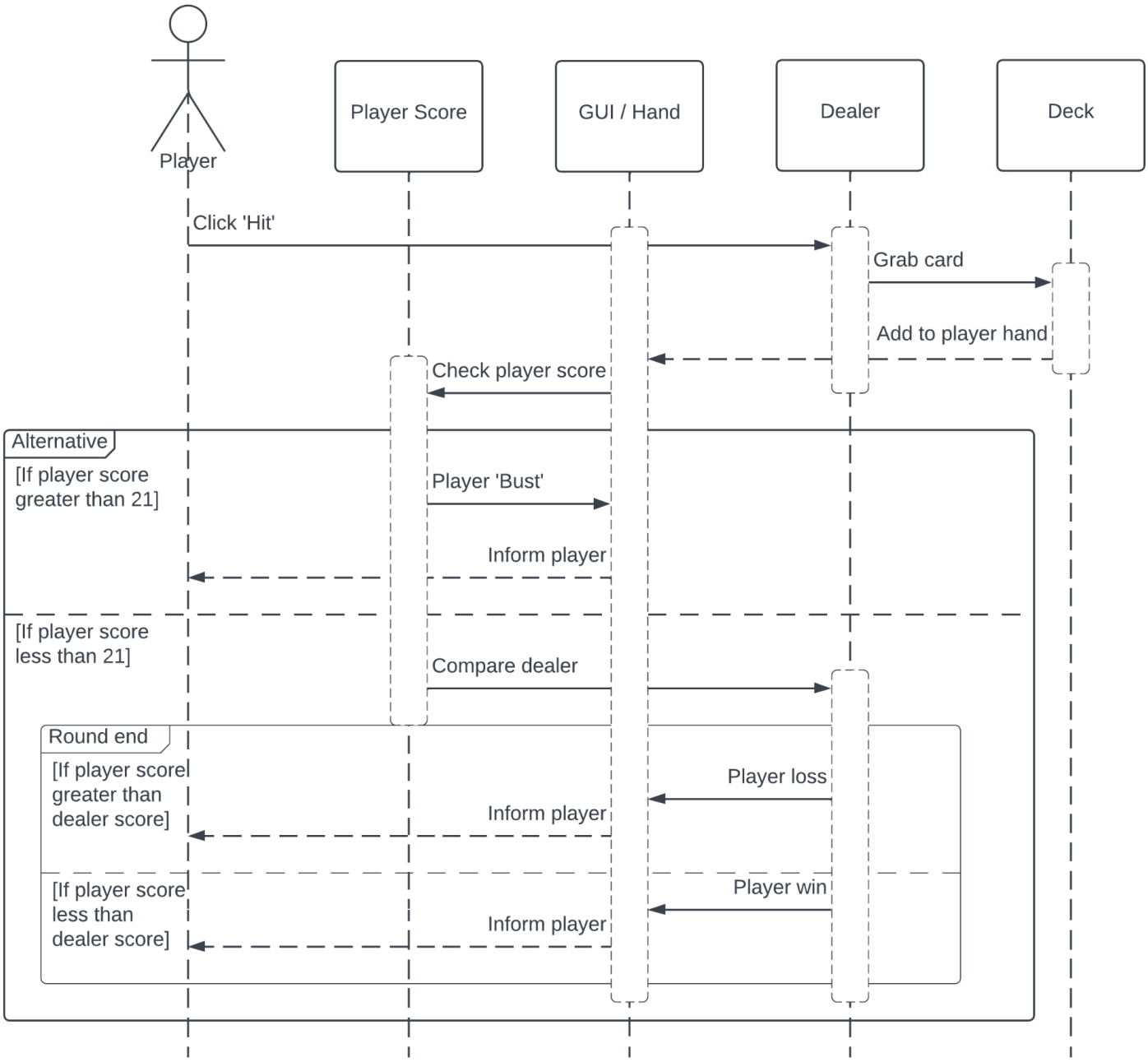


3. Sequence Diagrams Document

Join Game Sequence Diagram



Player Gameplay Sequence Diagram



4. Use Case Diagram