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 Fixed some grammar errors, and amended out-of-date requirements. 4.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)

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

Game/Seat – One player versus the dealer.

Table – A set of games or seats occurring at once in a virtual lobby.

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 5 major abstract modules: the Client-Server, Table, Game, Player, and Bank Modules.
- The Client-Server Module handles the GUI, establishes a client-to-server connection over TCP/IP protocol, and handles the login and registration process for a user.
- The Table Module represents a blackjack table that will host up to 7 players in a blackjack game following the standard blackjack rules using a 52-card deck.
- The Game Module represents seats at the virtual blackjack table which are to be occupied by players and handles players' minimum bets and how much they wager/win/lose.
- The Player Module will represent individual players who join blackjack games, and keep necessary information about the player.
- The Bank Module will handle wagers and transactions between the players and the dealer.

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. Player Module Requirements:

- Players must be able to authenticate their own login using a self-issued username (6 to 20 alphanumeric characters in length) and self-issued password (8 to 15 alphanumeric characters in length).
- Players will have access to personal information on their account, including wins, losses, past bets, etc.
- Players must be provided the option to change their username, password, and bank payment method, and the option to delete their account.
- If a player's balance falls below the minimum bet, they should be able to deposit funds through their bank/card.
- The player's balance should be adjusted after each game session and when leaving the table.

3.1.3. Game Module Requirements:

- The digital BlackJack decks must follow a standard 52-card format.
- Players must have the option to hit and stand.
- Players must be able to view their own cards.
- Cards dealt to players must be randomized to prevent bias.

3.1.4. Table Module Requirements:

- Blackjack tables must be able to support one to seven simultaneous games.
- When new players want to join, if there are no open slots at any tables, a new table must be created.
- Players will be able to see other instances of concurring Blackjack games on the table and are able to view other player’s cards.

3.1.5. Bank Module Requirements:

- Participating players must be required to hold a balance in their account that passes a minimum threshold in order to play the game.
- Players are able to add and withdraw funds from a “bank”.
- Players can check their current balance.

3.1.6. Client-Server Module Requirements:

- The graphical user interface (GUI) must be intuitive and user-friendly.
- The GUI should facilitate the server-to-client connection.
- The application should establish client to server communication over a TCP/IP protocol.
- 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 server must maintain uninterrupted operation, ensuring 24/7 availability.
- The application should handle user login and registration processes securely.

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," "Access Your 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 "Access Your Bank" option from the main page should lead to the bank page.
- On the bank page, users must have the option to edit their card details, which should direct them to a dedicated card details editing page.

Card Details Editing Page

- The card details editing page should enable users to update their card information.
- The bank page should also include a "Balance" option, which allows users to deposit money, check their balance, and withdraw funds from their balance.

Balance Page

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

Profile Page

- The user profile allows users to see information about their BlackJack game history, including wins, losses, past wagers, etc.
- The Profile also provides options for the user to modify information about their account, including username, password, payment method, and an option to delete their account.

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 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 App Module 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 modules 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 Card Details Editing Page to facilitate card information updates.
- The Bank class must interact with the Player class to retrieve and update user account details, including balances and card information.

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 works with the Game class to keep track of scores, bets, and card distributions for each player.

App Module Interaction

- The App module should display user interface elements for each page, including buttons for navigation and input fields for data entry.
- All pages should interact with the App module 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 data 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.