# Multiplayer Blackjack Game

Software Requirements Specification

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 02/25/2025 | 1.0 | Worked on section 1 | Basim Shahzad |
| 02/26/2025 | 1.1 | Worked on section 2 | Basim Shahzad |
| 02/26/2025 | 1.2 | 3.1.1 Common Requirements  3.1.2 Multiplayer Module Starter  Added 3.1.2.1 – 3.1.2.5  3.1.3 Stats Tracker Module Starter  Added 3.1.3.1 | Riley Fischer |
| 02/26/2025 | 1.3 | Worked on Section (4.1 & 4.2) | Basim Shahzad |
| 02/26/2025 | 1.4 | Added 3.2.1, 3.2.2 and 3.3.1 | Riley Fischer |
| 02/27/2025 | 1.5 | Added 3.1.3, 3.1.4 | John Nguyen |
| 02/28/2025 | 1.6 | 3.1.5 Game Module Requirements  Added 3.1.4.2, 3.1.4.5 – 3.1.4.8  Removed 3.1.4.1  Added 3.1.3.4, 3.1.3.5 | Martin Garcia |
| 02/28/2025 | 1.7 | Added intial ideas for user case document | John Nguyen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Purpose [4](#__RefHeading___Toc19440719)

1.1. Scope [4](#__RefHeading___Toc19440720)

1.2. Definitions, Acronyms, Abbreviations [4](#__RefHeading___Toc19440721)

1.3. References [4](#__RefHeading___Toc19440722)

1.4. Overview [4](#__RefHeading___Toc19440723)

2. Overall Description [5](#__RefHeading___Toc19440724)

2.1. Product Perspective [5](#__RefHeading___Toc19440725)

2.2. Product Architecture [5](#__RefHeading___Toc19440726)

2.3. Product Functionality/Features [5](#__RefHeading___Toc19440727)

2.4. Constraints [5](#__RefHeading___Toc19440728)

2.5. Assumptions and Dependencies [5](#__RefHeading___Toc19440729)

3. Specific Requirements [6](#__RefHeading___Toc19440730)

3.1. Functional Requirements [6](#__RefHeading___Toc19440731)

3.2. External Interface Requirements [6](#__RefHeading___Toc19440736)

3.3. Internal Interface Requirements [7](#__RefHeading___Toc19440737)

4. Non-Functional Requirements [8](#__RefHeading___Toc19440738)

4.1. Security and Privacy Requirements [8](#__RefHeading___Toc19440739)

4.2. Environmental Requirements [8](#__RefHeading___Toc19440740)

4.3. Performance Requirements [8](#__RefHeading___Toc19440741)

# Purpose

The purpose of this document is to outlines the software requirements for the Multiplayer Blackjack Game. The document details functional and non-functional requirements, system architecture, and constraints.

## Scope

The Multiplayer Blackjack Game is a Java-based application with a GUI that operates over TCP/IP. It will include the following core features:

* Multiplayer functionality (6 players max per game)
* Real players as dealer and players (no bots)
* Account management (fund deposits & withdrawals)
* Anti-cheating mechanisms (card counting detection, random shoe replacement)
* Game logic( betting, turns, win/lose conditions)
* Leaderboard for tracking player stats
* Lobby system for concurrent games

## Definitions, Acronyms, Abbreviations

* GUI: Graphical User Interface
* TCP/IP: Transmission Control Protocol/ Internet Protocol
* Shoe: A collection of multiple decks used in Blackjack to prevent card counting
* Dealer: The player responsible for dealing cards in the game
* Card Counting: A cheating strategy where players keep track of the cards already dealt from a deck, to determine their betting decisions.

## References

## Use Case Specification

UML Use Case Diagram(s)

Class Diagram(s)

Sequence Diagram(s)

## Overview

This document provides a detailed overview of the system architecture, constraints, functional and non-functional requirements for the Multiplayer Blackjack Game.

# Overall Description

## Product Perspective

This application is a multiplayer game that runs on client-server architecture. The system will consist of:

* Client application: GUI based interface for players and dealers.
* Server Application: Handles the current state of the game, player action, and turn management.

## Product Architecture

The blackjack game will consist of \_\_\_\_ modules: the multiplayer module, the stats tracker module, the cheating mitigation module, and the game module.

The system will be organized into \_\_\_ major modules: the \_\_\_ module, the \_\_\_ module, and the \_\_\_\_\_ module.

Note: System architecture should follow standard OO design practices.

## Product Functionality/Features

* Multiplayer Gameplay: Players can join games and compete in Blackjack.
* Turn-Based System: Players take turns in sequential order, and players must make their move in a certain time limit.
* Funds Management: Players can deposit and withdraw money
* Game Logic: Betting, hitting, standing, and winning will be accurately implemented
* Anti-Cheating Logic: Card counting detection, and randomized shoe replacement.
* Leaderboard: Tracks player earnings and win/loss records

## Constraints

* No bots allowed; all participants must be real people
* Maximum of 6 players per game
* Only as many tables as there are available dealers
* No spectator mode
* No communication amongst players

## Assumptions and Dependencies

* Players will follow the rules outlined to them
* Players will have a stable internet connection

# Specific Requirements

## Functional Requirements

### Common Requirements:

3.1.1.1 Users should be able to log in as either players or dealers.

3.1.1.2 The game should be fully GUI driven.

3.1.1.3

### Multiplayer Module Requirements:

3.1.2.1 No bots are allowed to be playing. It should be played purely by humans.

3.1.2.2 There should be exactly 1 dealer and 6 players per table.

3.1.2.3 There should always be the same number of active tables as there are dealers.

3.1.2.4 The gameplay should follow a turn-based structure. One player must be active at a time with the dealer.

3.1.2.5

### Stats Tracker Module Requirements

3.1.3.1 Each player should have full history of all the games they’ve played.

3.1.3.2 Transaction history like add funds or withdrawl money from players.

3.1.3.3 Players, dealers lose or win history

3.1.3.4 The system should record and display the following stats of each player:

* Total games played
* Total wins, losses, and pushes
* Win/loss percentages
* Highest winning in a single game
* Average bet per game

3.1.3.5 Game stats should be stored in a database (CSV file).

### Cheating Mitigation Module Requirements:

3.1.4.1 The System should prevent card counting by implementing an automatic reshuffle when 50% of the second deck has been used

3.1.4.2 Switching dealer for every table every 30 – 40 rounds.

3.1.4.3 Players should not be able to modify or inject any client side code to manipulate the game state, card distribution, or betting system.

3.1.4.4 The system should monitor betting patterns and flag suspicious activity.

3.1.4.5 Players should not be allowed to disconnect or refresh from the game to manipulate outcomes.

3.1.4.6 The system should ban players who break any of these rules.

* + 1. **Game Module Requirements**

3.1.5.1 The system should follow standard blackjack rules, where players attempt to reach a hand value of 21 without exceeding it.

3.1.5.2 The dealer must follow predefined rules, such as hitting on 16 or less and standing on 17 or higher.

3.1.5.3 Each game should support the following player actions:

* Hit: Draw an additional card.
* Stand: End turn without drawing more cards.
* Double Down: Double the bet and receive exactly one more card.
* Split: If the player has two cards of the same rank, they can split them into two separate hands (if the betting balance allows).
* Bust: Player exceeds 21

3.1.5.4 Players should be able to place bets within a predefined betting range before the game starts.

3.1.5.5 The game should handle blackjack payouts based on a 3:2 ratio and normal win payouts based on a 1:1 ratio.

3.1.5.6 A tie (push) should result in the player's bet being returned.

3.1.5.7 The system should enforce time limits for player actions to ensure smooth gameplay and prevent stalling.

3.1.5.8 The game state should persist in case of player disconnection. Forfeiting players bets.

3.1.5.9 The game should maintain a clear UI displaying:

* Player’s current hand and value.
* Dealer’s visible card(s).
* Available player actions.
* Current bet and balance.

## External Interface Requirements

3.2.1 The user interface for the client application must display all relevant information and provide means for the user to log-in, log-out, choose their user type (player or dealer), join a table, play blackjack.

3.2.2 The interface for the server application must log the activities of the active blackjack games.

## Internal Interface Requirements

3.3.1 The server should be responsible for processing player actions and inform the other players of what each other are doing. This will keep the game in sync.

Provide module specific requirements as appropriate. SR10

Example:

3.3.1 SR17 The system must process a data-feed from the grading system such that student grades are stored along with the historical student course enrolments. Data feed will be in the form of a comma-separated interface file that is exported from the grading system nightly.

3.3.2 SR24 The system must process a data-feed from the University billing system that contains new student records. The feed will be in the form of a comma-separated text file and will be exported from the billing system nightly with new student records. The fields included in the file are student name, student id, and student pin number.

# Non-Functional Requirements

## Security and Privacy Requirements

4.1.1 The SR8 system must validate user credentials before allowing logins for players or dealers

4.1.2 The player should not be able to modify their own or any other player’s game stats

4.1.3 Players should not be able to join during a game in progress, but players already in

a game should be able to leave.

## Environmental Requirements

4.2.1 The system must be a cross-platform application.

4.2.2 Main server will be hosted on one of the group member’s computers. SR25

4.2.3 SR26 System must be deployed on existing Linux-based server infrastructure.

## Performance Requirements

4.3.1 SR27 System must render all UI pages in no more than 9 seconds for dynamic pages. Static pages (HTML-only) must be rendered in less than 3 seconds.