# Group 5: BlackJack Game

# Software Requirements Specification

# Revision History

| Date | Revision | Description | Author |
|---|---|---|---|
| 10/3/23 | 1.0 | Initial Version | Quan Phanthaamat, Nikola Paunovic,Celeste Perez, Santiago Peña |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Table of Contents

# 1. Purpose

This document outlines the requirements for a BlackJack game that allows multiplayer gameplay using TCP/IP and an implemented GUI.

## 1.1. Scope

This document will catalog the user, system, and hardware requirements for the BlackJack game. It will not, however, document how these requirements will be implemented.

## 1.2. Definitions, Acronyms, Abbreviations

A *deck* consists of 52 cards with four suits, each with 13 cards. There are four suits on a deck of a card, hearts, diamonds, spades, and clubs.

A *bust* means that a player exceeds 21 and loses the game.

## 1.3. References



## 1.4. Overview

The BlackJack game is a virtual representation of the real card game played today. It lets players use their virtual funds to play with and/or against other players for fun abiding the game rules. This game will NEVER request payment from the user using real-world currency.

# 2. Overall Description

## 2.1. Product Perspective

Ever since the 1700s, BlackJack has been one of the classic card games played by people throughout the world. This project aims to bring the excitement of the game to everyone, without any financial risk. This game will allow players to create their own unique user accounts and play with and against others over an internet connection. There are no real currency transactions, as users will be provided with generous amounts of in-game currency.

## 2.2. Product Architecture

The system will be organized into three major modules: the game module, the player module, and the dealer 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):

2.3.1. Standard BlackJack rules further described in specific requirements must be implemented

2.3.2. Users must be able to use their own unique accounts to play

2.3.3. The game must be playable over a TCP/IP

2.3.4. The game must have a fully working GUI for all features available to the players

## 2.4. Constraints

2.4.1. Users must have the Java Runtime Environment (JRE) installed on their system to run this game.

2.4.2. Users must have a stable connection to the internet and be able to connect to external servers.

2.4.3. The game must be completely written in Java without any other dependencies.

## 2.5. Assumptions and Dependencies

2.5.1. It is assumed that the game server is able to handle the bandwidth of all the users

connecting to it.

# 3. Specific Requirements

## 3.1. Functional Requirements

### 3.1.1. Common Requirements:

3.1.1.1. Users should be able to create a unique user ID and password combination

3.1.1.2. Passwords must be at least 8 characters long and contain at least 1 digit

3.1.1.3. Users should be allowed to log in with their user ID and password combination.

3.1.1.4. The main screen of the game should provide some information, like the number of players currently waiting for a table, or the amount of players already at a table if one is available.

3.1.1.6. Each user should be able to see their own available funds.

3.1.1.5. Users should be able to use their available funds in the game, and they will be provided with these funds in regular intervals.

3.1.1.6. The rules of BlackJack should be implemented (further described in 3.1.2. and 3.1.3.).

3.1.1.7. There should be logging implemented for every game that logs connected players, scores, and the players who won and lost

### 3.1.2. Game Module Requirements:

3.1.2.3. Each game should have 8 decks available for cards to be drawn.

3.1.2.2. Each deck consists of 52 cards divided into 4 suits of 13 cards (hearts, diamonds, spades, clubs).

3.1.2.3. Jacks, Queens, and Kings should all be equal to 10 points.

3.1.2.4. Aces can either have a value of 1 point or 11 points.

3.1.2.5. All other cards must be equal to their face value.

3.1.2.6. There must be a minimum of at least 2 players in order for a game to be started. While only one player is connected and waiting, they should be in a waiting room.

3.1.2.7. In order for the player to join and participate in a game, the player must put in a minimum bet of $1 (virtual currency).

3.1.2.8. Each game should keep track of how much money is on the table, and the amount of times each player won.

3.1.2.9. A bust happens when a player or the dealer draw cards with the total point values over 21.

3.1.2.10. A round is over once all player(s) or the dealer have either won or lost.

3.1.2.11. All cards are dealt starting from the dealer and going around the table. Turns are taken starting from the person after the dealer.

### 3.1.3. Player Module Requirements:

3.1.3.1. Players should have the ability to add funds to their account and be able to bet a minimum amount per round (Funds are all virtual and provided through the game. No actual payment required.).

3.1.3.2. Each player is initially given 2 cards (drawn from the shuffled decks).

3.1.3.3 If a player's total hand value is equal to 21, that player wins 1.5 times their bet.

3.1.3.4 The player(s) should be prevented from playing the current round if their hand is higher than 21.

3.1.3.5 The player(s) should have the ability to "hit" or "stay" after the first cards were dealt to each player. Players who hit draw an extra card, players who stay keep their hand.

3.1.3.6 All players must be able to see all cards in their hand.

3.1.3.7 A player can also win if their hand is higher than the dealer's hand without either of them busting.

### 3.1.4. Dealer Module Requirements:

3.1.4.1. If the dealer goes over 21 every player wins twice of what they bet.

3.1.4.2. The dealer draws 2 initial cards, but one of them has to be revealed to other players.

3.1.4.3. When the dealer gets their turn, they must reveal the other card to the players.

3.1.4.4. If the dealer's hand total is 16 or under, they are required to draw another hand

3.1.4.5. If the dealer's hand total is 17 or over, they are required to keep their hand.

3.1.4.6. If the dealer wins (all other players bust or have their points lower than the dealer), they receive all of the other players' bets.

## 3.2. External Interface Requirements

3.2.1. Stable GUI: All players should be able to fully interact with all features they need using a built-in GUI solution. Examples of these features include:

3.2.1.1. Logging in

3.2.1.2. Waiting in lobby

3.2.1.3. Seeing their available funds

3.2.1.4. Seeing their hand and being able to hit or stay (as described in 3.1.3.)

3.2.1.5. Seeing the dealer's cards (as described in 3.1.4.)

3.2.1.6. Seeing their score (number of wins and losses)

3.2.2. The program must communicate with other devices through a TCP/IP server

3.2.3. The program must be compatible with any device that has JRE installed

## 3.3. Internal Interface Requirements

3.3.1. The program should be built in a modular way with classes that represent different parts of the program

3.3.2. The program should use plaintext files to store user info and game logs

3.3.3. The program should be able to automatically connect to the game servers and put multiple players in a game together.

# 4. Non-Functional Requirements

## 4.1. Security and Privacy Requirements

4.1.1. Every user must have a username and password to register and play the game

4.1.2. The username and password database will not be shown to other players explicitly, but will not be stored in an encrypted manner

## 4.2. Environmental Requirements

4.2.1. The server should be run on a computer with sufficient processing power to handle all requests

4.2.2. The network should be able to handle all requests

4.2.3. The program must only use Java as a dependency

4.2.4. The program should be operable using a mouse and a keyboard

## 4.3. Performance Requirements

4.3.1. The server must respond to player requests in a timely manner

4.3.2. The server must be able to handle as many players as reasonable

4.3.4. The server must be available at all times when it is scheduled to be running

# Use Case Specification Document

Use Case ID: BJG_5_1
Use Case Name: login
Relevant Requirements: See *Common* Requirement
Primary Actor: dealer and player
Pre-conditions: Both users need to have some knowledge about playing BlackJack. Players need to have some funds in order to play.
Basic Flow or Main Scenario:

1) Either user can initiate the login by typing their username and password.
2) The system authenticates and gives a successful or an error message.

Extensions or Alternate Flows: If login was not successful, then the user is denied access. They can either try again or create an account.
Exceptions:

Use Case ID: BJG_5_2
Use Case Name: play game
Relevant Requirements: See *Game Module* Requirement
Primary Actor: dealer and player
Pre-conditions: Before the game starts, dealers should have randomly shuffled the card.
Basic Flow or Main Scenario:

1) The rules of BlackJack are implemented.
2) Both players and the dealer get two cards each, players have the two cards face up, while the dealer's one card is faced down.
3) Players can either hit, stand, or surrender each turn.
4) If players go over 21, then they lose the game.
5) If players hit 21, then they only win the game if the dealer does not have 21.

Extensions or Alternate Flows:
Exceptions:

Use Case ID: BJG_5_3
Use Case Name: check funds
Relevant Requirements: See *Player Module* Requirement
Primary Actor: player
Pre-conditions: Players should have been able to successfully login.
Post-conditions: Player is allowed to join a open table
Basic Flow or Main Scenario:

1) User attempts to join/create a table.
2) The system checks the players current funds in their account.
    a) If the player has insufficient funds:
        i) Denied ability to join or create table
        ii) User is given the option to add funds to account
    b) If the player has sufficient funds
        i) Player is allowed to create/join a table
        ii) Player is prompted to bet an amount they choose

iii) Game can begin once all players are ready

Extensions or Alternate Flows: Players are able to add or remove funds from their account.
Exceptions: Invalid credentials or Error adding funds.

Use Case ID: BJG_5_4
Use Case Name: shuffle cards
Relevant Requirements: See *Dealer Module* Requirement
Primary Actor: dealer
Pre-conditions:
Dealers should have been able to successfully login.
Post-conditions: All decks will be reshuffled to ensure randomness.
Basic Flow or Main Scenario:

1) If the dealer has gone through four decks
    a) Shuffle the cards and reuse the deck

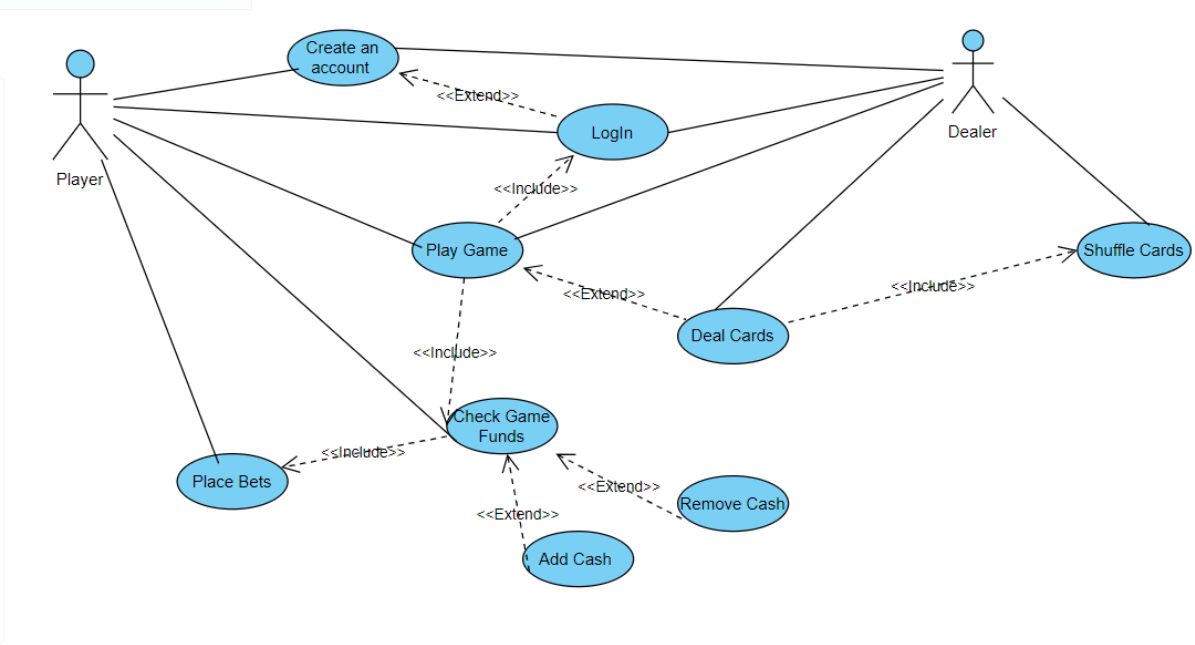
Extensions or Alternate Flows:
Exceptions: The game doesn't go through four decks or some sort of system error.

Use Case ID: BJG_5_5
Use Case Name: create an account
Relevant Requirements: See *Common* Requirement
Primary Actor: player and dealers
Pre-conditions: No accounts associated with either users.
Post-conditions: Player will have access to the full interface(Add funds, play games, etc)
Basic Flow or Main Scenario:

1) A new user will have the option to sign up for a new account
2) The system will ask the user for a username and password
3) The system validates the information and creates a new account for the player

Extensions or Alternate Flows: If they have an account with the system already, a message will let them know that an account has already been created with that username.
Exceptions: Invalid information, credentials that already exist, or possibly some server error

UML Use Case Diagrams Document



Class Diagrams