Software Requirements Specification for BlackJack

Prepared by Reymundo Guerrero, William Schneider, Sumanth Reddy Bekkem

Table of Contexts

Revision History

Name	Data	Reason for Change	Version
Rey, Will, Sumanth	9/19/2024	Initial Draft	1.0

2. Introduction

2.1 Purpose

The primary objective of this document is to define the requirements for our Java Application-based Blackjack game. It outlines the game's functionalities, including gameplay mechanics, user interface design, betting system, and game statistics tracking. The application aims to replicate the classic blackjack experience, providing an immersive and realistic gaming environment. This document will guide the implementation and testing phases to ensure smooth gameplay and flawless functionality, with the ultimate goal of providing an outstanding user experience.

2.2 Project Scope and Product Features

The Blackjack game will include a scoreboard tracking username, Wins, Losses, Win-Loss Percentage, and Points Won/Lost. Users can play only by logging in, their statistics will only appear on the scoreboard if they create or log in to an account. Each player starts with 100 points to use for betting, with a maximum bet of 100 points per game.The game mimics the traditional Blackjack card game, where players compete against a dealer to reach a hand value closest to 21 without exceeding it. Users can access their game records, such as the number of wins and losses, through the scoreboard in the application. This project integrates with Postman for API requests, allowing automation and testing of functionalities like login, signup, and game management. All data, including user details and gameplay statistics, are stored in a PostgreSQL database.

2.3 Vision

This project will be using Java to create a game application to simulate Blackjack. The application will include a scoreboard showing user statistics. Our vision is to allow users to play Blackjack and have the computer automatically keep track of their progress. This includes tracking wins, losses, and points earned. Additionally, the game will allow users to log in or play on a guest account and play against the dealer, and track their progress over time through a secure, and user friendly interface. When a user is finished, they can logout, if it's a guest account the data would be removed once the session is closed. Later, if they have an account they can return and continue playing, and the computer will remember their winnings from

previous games. The application would allow users to merge statistics from guest sessions into their profiles when they log in. Users will be able to place bets using virtual points, review their game history and compete to enhance their personal statistics The aim is to provide a seamless experience where users can play uninterrupted and have their progress saved for future sessions, enabling them to continue playing with all their previous data intact. Our vision is to expand the game into a multiplayer platform where users can compete with others and track their performance on global leaderboards.

2.4 References

Wei Jie, “Sample Specification Document” City College of New York http://www-cs.ccny.cuny.edu/~csjie/322/spec_sample.pdf

Ravi Bandakkanavar “Software Requirement Specification document with examples” Krazy Tech https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

“BlackJack” MassGaming https://massgaming.com/wp-content/uploads/Rules-Blackjack-6-20-18.pdf

3. Overall Description

3.1 Product Perspective

The Blackjack application is a new system that recreates the classic card game. It includes a points-based betting system to simulate the gambling aspects of the original game. The context diagram in Figure 1 outlines the main components and system interfaces for release 1.0. The system is designed to evolve in future releases, eventually expanding into a multiplayer platform where each player has individual statistics and profiles.

3.2 User Classes and Characteristics

User Profile: Each user has a dedicated profile within the system, containing their username, password, and game statistics such as the number of wins, losses, and their win-loss percentage. Additionally, the profile includes their points balance, which reflects their in-game currency.

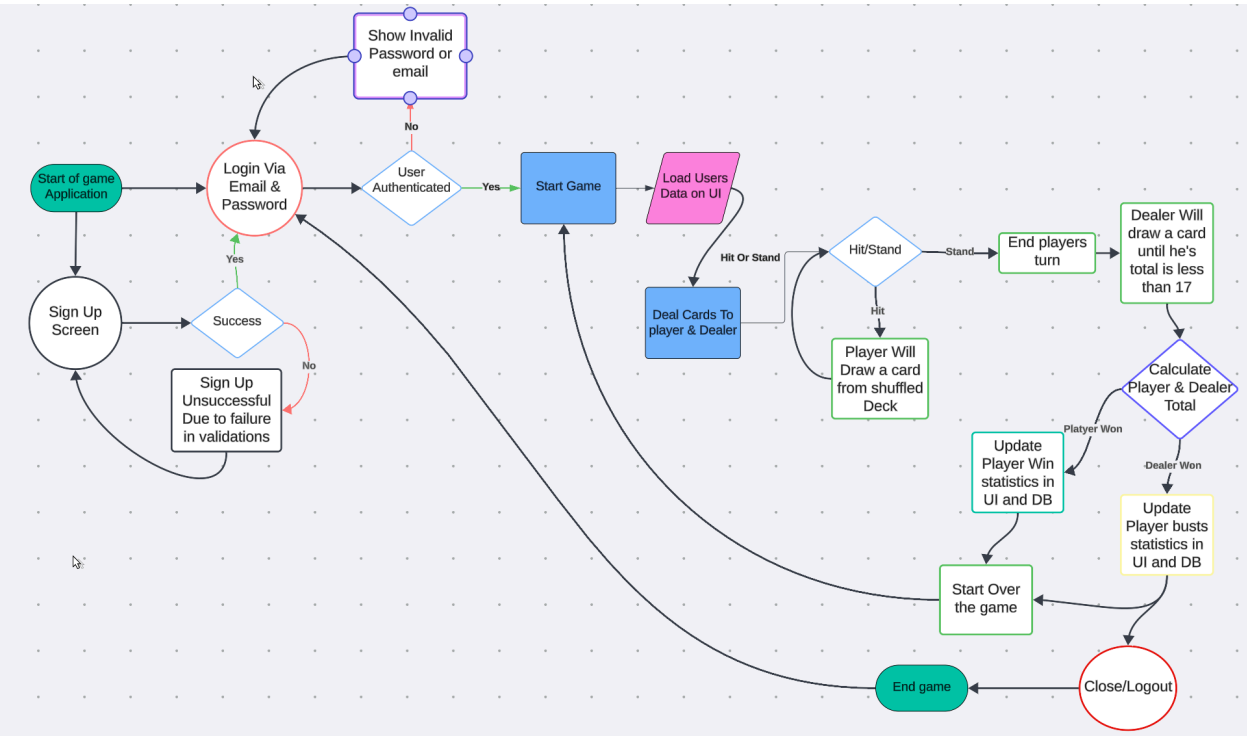
Persistent Data: The persistent data includes all game-related information such as betting history, earnings, and statistics. This data is stored in the PostgreSQL database and can be easily accessed every time the user logs in.

Authentication: To access their profile, users must log in. This process involves submitting a valid username and password. Once authenticated, users can start their gameplay from where they last left off.

Place Bets: Authenticated users are able to place bets using their point balance, which is dynamically updated in real-time to reflect any changes in their account. Users who lose all their point balance are able to buy in at 100 points. This provides users with the ability to make informed betting decisions based on their current point balance.This feature will be developed once all the basic game and login/sign up features have been implemented.

Track Statistics: Their profile is designed to automatically update with the most recent statistics following each game, offering a tailored and up-to-date experience.

Profile Management: Users are able to log in and manage their profile information, such as updating personal details and preferences. Additionally, they can also access and review their historical performance data, allowing them to track their progress



Web link to above figure:
https://lucid.app/lucidspark/ded50a11-c445-4be3-a5c1-689698f1acc3/edit?invitationId=inv_eea8c966-76e1-4e5a-b2a1-1ac35652c43e

Figure 1
Context Diagram for release 1.0 BlackJack

3.3 Operating Environment

OP-1: The game application will be requiring the Java Development Kit (JDK) version 8 or above to compile and run the application. The system must have the appropriate Java Runtime Environment (JRE) installed. Unsupported or older versions of Java may cause runtime, compilation and spring dependency errors or prevent the application from launching.

OP-2: The system needs to support API interactions through Postman. Postman must be able to make requests to the application once the user is loggedIn, which assumes an environment that allows HTTP/HTTPS communication.

OP-3: The application will rely on a PostgreSQL database for storing user and game metadata. Any database server running incompatible versions may cause issues with data storage and retrieval. Postgres database availability is a mandatory constraint in order for the application to boot up. And the port on which the database should run is 5432.

3.4 Design and implementation Constraints

CO-1: The system's design, code, and documentation shall conform to the internal java Application Development Standard, Version 2.0.

CO-2: The system shall use Postgres as the primary database engine for storing user profiles, statistics, and game data.

CO-3: All backend scripts shall be written in Java using the Spring Framework for API development and data handling.

3.5 User Documentation

UD-1: The user documentation will include clear steps for users to sign up, log in, and start playing the game. It will explain how to create an account, log in with an email and password, and the basic rules of playing Blackjack within the application.

UD-2: The documentation will provide instructions on how to view personal game statistics, such as wins and losses, and how to play against the dealer. It will also guide users on using the “Hit” and “Stay” buttons during gameplay and explain when the game ends based on the card total.

3.6 Assumptions and Dependencies

AS-1: It is assumed that the user’s system has Java 8 (or higher) installed to run the game without issues. The game won't work correctly if Java isn't set up properly.

AS-2: The application depends on a working PostgreSQL database for storing user data and game statistics. It assumes that the database server is available and properly configured at all times when the user wants to play the game in their local machine.

4. System Features

4.1 Login:

Users can log in to the system using their username and password. Once logged in, they gain access to game features and their personal statistics.

4.2 Sign Up:

Users who are new to the application can create an account by entering an unique username, an optional email, and a secure password. This allows them to access the game and track their progress once they signed up and logged in.

4.3 Guest Account:

Guest users have the option to play the game without the need to log in with a registered account. During their session, they are able to fully engage in the gameplay, including placing bets, participating in games against the dealer, and temporarily accessing in-game statistics. However, any progress made and statistics accumulated during the session will not be saved once the session ends.

4.4 Merging Guest Account Data:

When a guest user creates a new account, during the registration process, they have the option to transfer their session data into their permanent profile. By choosing this option, guest users can ensure that their progress, including any bets placed, wins, and losses, will be retained and carried over when they log in using their permanent profile. This feature allows for a seamless transition from a guest account to a permanent account, preserving all the relevant session data. This feature is only available when creating a new account. Guest users' data cannot transfer to an existing account. This will be an additional feature we might develop if time permits.

4.5 Start Game Against Dealer:

Once logged in, players can begin a Blackjack game, where they compete against a dealer. The goal is to reach a card total of 21 without going over, or to get closer to 21 than the dealer. In

this game the game builds a deck of cards which are shuffled randomly and automatically for each game the user plays. So that the game is fair every time against the dealer.

4.6 Player Can Hit/Stay on His Choice:

During the game, players can choose to "Hit" (draw another card) or "Stay" (stop drawing cards). Players can continue hitting until they are satisfied with their hand or risk going over 21. In this game the game builds a deck of cards which are shuffled randomly and automatically for each game the user plays. So that the game is fair every time against the dealer.

4.7 Logout:

Players can securely log out of their account after finishing their game or checking their statistics. Logging out ensures their session is closed and their data is protected. All the game data will be permanently persisted in the postgres database before logging out. Which will ensure the data reliability.

4.8 Automation Requests Using Postman:

This is an additional feature, which will be implemented where the system allows automated testing and requests through Postman. Users can simulate actions like signing up, logging in, and playing the game by sending API requests, making it easier to test and interact with the application programmatically. Even though in real world applications games won't be implemented in REST based services we decided to add this functionality because we thought it might add value to the overall application and project.

4.9 Additional features we might implement:

4.10.1 Forgot password: When a user forgets their password they will be able to create a new password by clicking on forgot password. Upon clicking the "forgot password" button they will be redirected to a "forgot password" page where they will need to enter their email id, new password, and verification code. A verification code will be sent to his email in order to reset his password.

5. External Interface Requirements

5.1 User Interfaces

UI-1: The game provides a simple interface for players to interact with, including buttons for "Hit", "Stay", and "Logout" during gameplay. The UI should be user-friendly and display game statistics and outcomes clearly for logged-in users.

5.2 Hardware Interfaces

No hardware interfaces have been identified

5.3 Software Interfaces

SI-1: The game must support external requests through Postman, allowing users to perform actions like login, signup, and gameplay via HTTP API requests. The system should handle requests and responses in JSON format.

SI-2: The game requires a connection to a PostgreSQL database to store and retrieve user details and game statistics. The database interface must support SQL queries for inserting, updating, deleting and fetching data. And the system should be able to communicate with databases running on 5432 port.

5.4 Communication Interfaces

CI-1: This is an additional feature or requirement which we will implement if time permits. The application will send an verification email to users email in case of a forgotten password. This will be requiring a communication interface between gmail and java applications.

CI-2: The game provides an API layer that allows external tools like Postman to send requests for automating tasks such as signing up, logging in, and playing games via HTTP protocols.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

PI-1: The system should respond to user actions (like login, hitting, or standing) within 2 seconds to provide a smooth gameplay experience. Delays in game interactions should be minimal to maintain user engagement.

PI-2: The installed PostgreSQL database must rigorously handle queries related to user login, signup, and game statistics, ensuring that user data is fetched and updated in less than 2 seconds.

PI-3: Since this game runs individually on each system scalability will not be an issue. As each system can run the application locally with minimum CPU, RAM requirements. In future when the system is expanded to support multiple users, scalability and runtime efficiency will be considered when each player plays the game simultaneously.

6.2 Safety Requirements

No safety requirements have been identified.

6.3 Security Requirements

SI-1: We will be using HTTPS for all data exchanges between the game and the server via Postman. This helps prevent attackers from intercepting sensitive information during online play.

SI-2: Needs to check user inputs in text fields that are being entered by the user to prevent malicious code from being executed. This helps protect against attacks like SQL injection, which can compromise the database.

SI-3: Ensuring only authorized users are allowed to play the game by implementing login, Signup, Logout functionalities.

6.4 Software Quality Attributes

Reliability: The game should work consistently without application crashes or bugs due to issues like memory leak, stackover flow error etc.. Players should be able to rely on the game to function correctly every time they play.

Performance: The game should run smoothly and efficiently, with not so noticeable lags or UI delays. Players shouldn't experience long loading times while trying to load user statistics from the postgres database, ensuring a fun and engaging experience.

Usability: The game should be understandable and easy to play. Users should have a smooth experience, with clear instructions and clear cut controls.

Appendix A: Data Dictionary and Data Model

ID - Generated identifier (primary key)

Username - User's name shown on the scoreboard

Password - User's password to log into their account

Points - Number of points won/lost across all games played by a user. These can be negative or positive.

Wins - Number of games won by the user.

Losses - Number of games lost by the user.

Games played - Number of games played by the user.

Gmail - If the account recovery feature is added, this attribute will be the user's recovery email address.

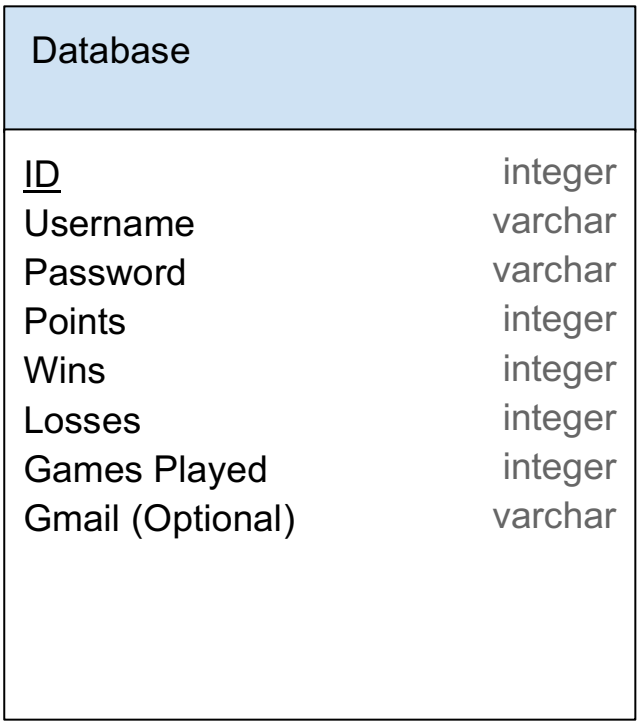


Figure Two:
Database model

Appendix B: Inspection Report

Issue	**Status**	**Solution**
Omitted the features about the guest account.	Resolved	Wrote the features about the guest accounts and ensure there were no ambiguity about how data is handle with the guest account
The Context diagram was limited in its approach to graphically show all features and aspects of the application.	Resolved	The Context Diagram was enhanced to show all aspects of the application
Lacked definitions of the attributes in the database	Resolved	Added definition for each attribute in the database and identify its primary key
Vision was omitted from the SRS document	Resolved	Add the Vision section to the SRS document detailing the vision of the application.