Group Project
Multi-Player Blackjack

CS-401-01
Software Engineering

Group 2

Andrew Bustos
Amanuel Gebreysus
Steven Tran
Diego Ureno
Haolin Zhang

Software Requirements Specification

Revision History

Date	Revision	Description	Author(s)
6/21/22	1.0.0	Requirements document created.	Diego Ureno, Andrew Bustos, Steven Tran, Haolin Zhang, Amanuel Gebreysus
6/22/22	1.0.1	Finished fleshing out module requirements.	Andrew

Table of Contents

1. Purpose

This document outline the requirements for the Multi-Player Blackjack Project

1.1. Scope

This document will catalog the user, system, and hardware requirements for the Multi-Player Blackjack project. This document will **not** show how these requirements will be implemented.

1.2. Definitions, Acronyms, Abbreviations

UN	Username
PW	Password

1.3. References

Use Case Specification Document - *Use cases*
UML Use Case Diagrams Document - *Use Case Diagram*
Class Diagrams - *Class Diagram*

1.4. Overview

The Multi-Player Blackjack Project is designed to create a gaming system, which allows clients to log in, add funds to their accounts, play games of Blackjack (standard rules applied) with other players, or cash out their accounts.

2. Overall Description

2.1. Product Perspective

2.2. Product Architecture

The system will be organized into *(insert number here)* modules: the *(insert name here)* module, the *(insert number here)* module, and the *(insert number here)* module.

NOTE: System architecture should follow standard OO design practices.

2.3. Product Functionality/Features

The features of the system are as follows:

- Allows clients to log in to their accounts
- Add funds to client accounts
- Cash out funds from client accounts
- Play multiplayer Blackjack (standard rules)

2.4. Constraints

- Real players only (No AI players)
- Players must have an account (No Guest accounts)
- Game must use at least 1 deck of cards
- System must be written in Java w/ GUI
- System must operate over TCP/IP
- System must have server and client application
- System must have a GIT source control repository
- System must have a Junit Test suite
- No web or HTML component
- No databases, libraries, frameworks, or other unapproved technologies

2.5. Assumptions and Dependencies

It is assumed that:

- The standard rules of Blackjack will be applied.
- The deck in play will have 52 cards in total.
- An Ace from the deck will have the value of 1 or 11.
- The minimum number of players required for a game is 2.
- The maximum number of players in a game will be 5.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements

3.1.1.1. Users should be able to login in with their respective accounts.
3.1.1.2. Users should be able to create new accounts.
3.1.1.3. Users should be able add funds or cash out funds from their accounts.
3.1.1.4. Users should be able to play Blackjack with other players.

3.1.2. Client Module Requirements

3.1.2.1. Clients must be able to receive constant error feedback when logging in/signing up.
3.1.2.2. Clients must be able to view/modify their current account balance when not playing a game.
3.1.2.3. Clients must maintain a persistent connection to the server, and must maintain a persistent cash balance between games.
3.1.2.4. Clients must receive constant feedback of how the game progresses, including knowing when they win/lose.
3.1.2.5. Clients must be able to create lobbies and start games, as well as leave them once they run out of cash or when the game ends.

3.1.3. Server Module Requirements

3.1.3.1. The server must send error feedback to clients when the requested username is not found or when the password is incorrect.
3.1.3.2. The server must create new lobbies once requested by clients and keep track of existing lobbies.
3.1.3.3. The server must update the client's account balance once the latter disconnects.
3.1.3.4. The server must keep track of all currently logged in clients.
3.1.3.5. The server must keep track of how many clients are currently in a lobby.

3.2. External Interface Requirements

3.2.1. The system must have a GUI interface that operates over TCP/IP. All code should be in Java.

3.3. Internal Interface Requirements

3.3.1. The system must process a user's ID/account with a username and password.
3.3.2. The system must accurately reflect each client's balance before/after connecting to the server.

3.3.3. User accounts must process user’s request to deposit/withdraw funds

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1. Players must not be able to see the password information of other players.
4.1.2. Players must not be able to see other player's cards (Cheating Minimized)

4.2. Environmental Requirements

4.2.1. System cannot require any software that uses web, HTML components, databases, libraries, and other unapproved technologies.
4.2.2. System code must be written in Java.
4.2.3. System must operate over a server and client application (TCP/IP)

4.3. Performance Requirements

4.3.1. System must be able to run GUI and server/client applications normally.
4.3.2. System must render all UI pages in no more than 6 seconds for dynamic pages
4.3.3. Static pages, if any, must be rendered in less than 6 seconds.