

Link to group GitHub Repository
<https://github.com/sheed17/CS401Group2>

SRS

CS401 Blackjack Game Project

Software Requirements Specification By

Luis, Alexis, Aldrei, Rasheed

Revision History

[illegible]

Table of Contents

1. PURPOSE	4
1.1. SCOPE.....	4
1.2. DEFINITIONS, ACRONYMS, ABBREVIATIONS	4
1.3. REFERENCES	5
1.4. OVERVIEW	5
2. OVERALL DESCRIPTION.....	6
2.1. PRODUCT PERSPECTIVE.....	6
2.2. PRODUCT ARCHITECTURE	6
2.3. PRODUCT FUNCTIONALITY/FEATURES	6
2.4. CONSTRAINTS.....	6
2.5. ASSUMPTIONS AND DEPENDENCIES.....	6
3. SPECIFIC REQUIREMENTS.....	7
3.1. FUNCTIONAL REQUIREMENTS	7
3.2. EXTERNAL INTERFACE REQUIREMENTS	9
3.3. INTERNAL INTERFACE REQUIREMENTS	10
4. NON-FUNCTIONAL REQUIREMENTS	11
4.1. SECURITY AND PRIVACY REQUIREMENTS	11
4.2. ENVIRONMENTAL REQUIREMENTS	11
4.3. Performance Requirements.....	11
5. USE CASE SPECIFICATION DOCUMENT	12
6. CLASS DIAGRAM DOCUMENT	18
7. UML USE CASE DIAGRAM DOCUMENT	20
8. SEQUENCE DIAGRAM DOCUMENT	23

1. Purpose

This document outlines the requirements for our Blackjack game project.

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.

1.1. Definitions, Acronyms, Abbreviations

BlackJack Terms

Blackjack (card hand): A combination of two cards that includes an Ace and any card with a value of 10 (such as 10 of spades or any face card).

Dealer: A non-player server-side role responsible for dealing cards, managing the deck, and playing its hand according to the standard rules

Player: A user client connected to the system that places bets and plays a hand against the dealer

Hand: The set of cards possessed or “held” by a single player

Push: A tie between a player’s hand and the dealer’s hand

Double Down: A player’s action to double their bet in exchange for drawing an additional card.

Hit: A player's action to request an additional card from the dealer

Split: A player’s action to split their hand into two different hands with their own respective bets, on the condition that the first two cards of the initial hand are of the same rank

Stand: A player's action to refuse any additional cards, finalizing the current hand.

Bust: A hand whose value exceeds 21, resulting in a loss for the hand’s owner.

Upcard: The first card dealt by the dealer. Dealt face-up and visible to all players.

Application Terms

Server Application: The central Java application in charge of hosting the game’s logic, managing player accounts, and handling network communication

Client Application: The Java GUI application through which the player interacts with the game and connects to the server.

Game Table: An instance of a Blackjack Game on the server where multiple players can participate in the gam

1.2. References

Use Case Specification Document

UML Use Case Diagrams Document

Class Diagrams

Sequence Diagrams

1.2. Overview

The project our group is working on is a multiplayer Blackjack game that operates using a client application and a server application. The game should be scaled infinitely meaning that the game should work with any given number of players.

2. Overall Description

2.1. Product Perspective

The product we are designing is a Multiplayer Blackjack game that is designed to be scaled infinitely, meaning that it will work for any given number of players, theoretically. The game itself will have measures against cheating and will ensure that all play is fair.

2.2. Product Architecture

The system will be organized into 5 major modules: the player module, the dealer module, the game table module, the main menu/system module and the shoe 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):

2.4. Constraints

- 2.4.1 The system MUST have two applications, one for the server and another for the client.
- 2.4.2 The program must contain a GUI that operates over TCP/IP.
- 2.4.3 The system does not include a web or HTML component.
- 2.4.4 The system cannot utilize databases, libraries, or frameworks.

2.5. Assumptions and Dependencies

- 2.5.1 It is assumed that the device accessing the multiplayer blackjack game has Java SE 8 or above installed.
- 2.5.2 It is assumed that the server supporting the multiplayer blackjack game is constantly running, providing support for the players and staff to connect.
- 2.5.3 It is assumed that the player has a stable internet connection to access the multiplayer blackjack game with.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Players will be able to log in using their credentials. Credentials are made up of username and their password.

3.1.1.2 Upon opening the application for the first time, users will be greeted with a page to log in if they have an existing account, or a registration page if they are new to the application

3.1.1.3 Upon creating a new account, users will see tooltips explaining what every button does and how to navigate the system.

3.1.1.4 The system should be able to handle invalid inputs from the user such as invalid passwords that don't meet requirements, etc. This will be done without interrupting the system processes.

3.1.1.5 All modules should be written with scalability in mind. This means that the game should support as many players as the hardware allows. This requires that there be no hard limits within the system's code, as it would prevent any scalability.

3.1.1.6 All modules should feature naming conventions that make it easy to understand what they're responsible for and what methods they'll contain.

3.1.2. User/Player Module Requirements:

3.1.2.1 The user should be able to log into the blackjack application using their username and password.

3.1.2.2 The user will be able to change their username, password.

3.1.2.3 The user will be able to see an overview of the current blackjack tables and the number of players within them. If the table has room and it is in between rounds, the player can then join it if they wish.

3.1.2.4 The user's blackjack game results will be recorded within the system log after completion. The user can access these logs via their user profile.

3.1.2.5 Each user will have a profile where they can view their personal data, their funds, and their game history.

3.1.2.6 Users will be automatically logged out from the system after 10 minutes of inactivity while not inside a blackjack game and after 1 minute of inactivity within a blackjack game.

3.1.2.7 The users will only be allowed to join one blackjack table at a time. They are only allowed to leave the table in between rounds.

3.1.2.8 The system should allow the player to Hit, Stand, Double Down or Split depending on their cards.

3.1.2.9 The system should allow the player to bet between \$5 and \$50,000.

3.1.3. Dealer Module Requirements:

3.1.3.1 The initial deal must be done by the dealer, providing two cards for the dealer and each and every player.

3.1.3.2 The dealer must automatically reveal the hidden card after all players in the game have completed their actions (Stand, Bust, or Double Down)

3.1.3.3 The dealer must continue to draw cards if the total value of its hand is under 16, but upon reaching 17 or above, the dealer must “Stand”.

3.1.3.4 The dealer must resolve the round by comparing its final hand value to each player’s hand values to determine wins, losses, and pushes. x

3.1.4. Shoe Module Requirements:

3.1.4.1 The system should utilize the standard casino six-deck setup. This means that 6 decks of cards will be shuffled together and will be used to play the game. Each deck is the standard 52-card pack, resulting in a total of 312 cards.

3.1.4.2 Using the 6-deck setup for blackjack, there will be 24 of each card. 24 Aces, 24 of each face card, 24 of number cards (2-10).

3.1.4.3 A “blank card” will be used to determine when it is time to shuffle the cards again. In our case, the blank card will be a randomly generated number from 20-50. When the number of cards left in the deck has reached this number, the deck must be reshuffled.

3.1.4.4 Upon a card being drawn, the number of available cards will be decremented. This is to be used in tandem with the “blank card” mentioned in requirement 3.1.4.3.

3.1.4.5 If all players leave the current table, the deck of cards will be automatically reshuffled to provide the same odds once a new game session begins.

3.1.5. Game Table Module Requirements:

3.1.5.1 The system should allow between 1 and 7 players in a single table.

3.1.5.2 The system should allow an unlimited number of tables.

3.1.5.3 The system should only have 1 dealer in a table.

3.1.5.4 The system should display on the table the bets, players’ names and states (Hit, Stand, Double Down, Split, Lose, Bust), cards of the players and dealer with the hand

count, and the timer.

3.1.5.5 The system should have a 20-second timer for each turn.

3.1.5.6 The system should not allow bets during dealing.

3.1.5.7 The system should have a minimum table bet of \$5 and a maximum of \$50,000.

3.1.5.8 The system will assume that the player “Stands” if the player does not provide a response or didn’t choose an action to do for its turn.

3.1.5.9 The system must enforce a 10-second window for players who have disconnected, allowing them to return and resume the round after placing a bet. In case a player does not return during this window, the game will automatically treat them as a player who “stands” for the rest of the round. After the round, the player’s balance will be updated accordingly.

3.1.5.10 The system must ensure that if a player is disconnected from the game before placing a bet that they are removed from the round without balance changes.

3.1.6. Main menu/System Module

3.1.6.1 The system must display a main menu upon startup of the client application.

3.1.6.2 The main menu must provide the player with options such as Login, Register, Join Game, and Exit.

3.1.6.3 The system must verify player credentials upon “Login” using the text file that stores players’ information.

3.1.6.4 The system must allow new players to create an account using “Register” by writing a unique username and password that would also ask the new player to re-enter password to create their account.

3.1.6.5 The main menu must allow players to join an active Blackjack table game session, where multiple players can connect and play simultaneously against the dealer using “Join Game”.

3.1.6.6 The main menu must allow players to exit the application using “Exit.” If the player exits during an active game session, the system must save their updated balance to the players.txt file before closing the client application.

3.1.6.7 The system must display an error message if the player enters an invalid username or password during “Login”

3.1.6.8 The system must ensure that password fields are hidden with asterisks when entered by the player.

3.1.6.9 The system must ensure that the main menu remains accessible until the player selects “Exit”.

3.2. External Interface Requirements

3.2.1 The system should have an interface that allows users to log in or register before being able to view their profile, refill their balance, check their balance, and play.

3.2.1.1 The Profile interface must display the player's name, current balance, and game history.

3.2.1.2 The Check Balance interface must display the player's balance, transaction history, and two buttons to either refill or cash out their balance.

3.2.1.3 The Play interface displays available tables, and the player is provided with the option to choose which table to join. Along with the table, the players and available seats are also displayed.

3.2.1.4 If the player does not have an existing account, the "Register" option is available for the player to create a new account with a username and password in order for their data to be linked properly.

3.2.1.5 If a player already possesses an account, the "Login" option lets the user enter their username and password for them to be able to start playing, view their profile, and withdraw or refill their balance.

3.2.2 The system should have an interface that allows the user to check the tables in the game, showing the players and available seats.

3.2.3 The system should provide interactive buttons for all player actions available during one's turn, with actions being enabled or disabled based on the game state and the player's hand.

3.2.4 The system must provide visual updates based on the game state and outcomes, such as wins, losses, and busts.

3.2.5 The game table screen must display the turn-timer (As mentioned in 3.1.5.5) for each player as soon as their turn begins.

3.2.6 Each table displays a Table ID both in the table selection screen and during the game.

3.3. Internal Interface Requirements

3.3.1 The system should provide any information such as the user's funds, user data, game status and such to the GUI so that it can be displayed to the user.

3.3.2 The modules within the system and their methods should communicate with one another in order to keep the game state and user information up to date. For example, if the user loses a game of blackjack, then the game table module will inform the user module of this and the wagered amount will be deducted from the user's account.

3.3.3 The system must store user information such as username, email, password, account funds, etc. This will be done using a text file that separates the data fields using a comma.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

- 4.1.1 The system must ensure that players can join a game by only using their assigned credentials.
- 4.1.2 The system must ensure that text files are protected and can not be altered or deleted when the game is running.
- 4.1.3 The system must ensure that if players make simultaneous bets, the system can handle both actions correctly by preventing file corruption or overwriting, so that all bets remain accurate and consistent.
- 4.1.4 The system must ensure that each player can only view their OWN private information such as their hand, bet amount, and balance and NOT other player's data.

4.2. Environmental Requirements

- 4.2.1 The system must use TCP/IP sockets for communication between client and server.
- 4.2.2 The system must store all relevant data such as (player username and password, game logs, account balance, etc) in text files.
- 4.2.3 The system must support multiple concurrent client connections over a single server instance.

4.3. Performance Requirements

- 4.3.1 The system must allow 7 players to connect and play simultaneously per table.
- 4.3.2 The system must process and display each player action such as hit, stand, bet to all connected clients instantaneously, excluding any possible delays from the network connection.
- 4.3.3 The system must ensure that file operations such as read or write does not delay gameplay by more than 2 seconds.
- 4.3.4 The system must handle the capacities of multiple players performing simultaneous bets.
- 4.3.5 The system must recover from client disconnections within 10 seconds, allowing the player to join the current state.
- 4.3.6 The system must ensure that GUI updates such as game actions run smoothly without freezing or causing any performance issues.

CS 401 Blackjack Game Project

Use Case Specification Document

Use Case ID: UC-01

Use Case Name: Register Account

Relevant Requirements: 3.1.1, 3.1.6

Primary Actor: Player

Pre-conditions:

1. The player does not already have an existing account.
2. The client application is running and the main menu is accessible to the player.
3. The server application is active and running and able to handle requests.

Post-conditions:

1. A new account is created and stored in the players.txt file.
2. The player is then redirected to the login page.

Basic Flow or Main Scenario:

1. The player selects Register from the main menu.
2. The system then displays the registration form requesting username, password, and re-entry of password.
3. The user inputs the required information.
4. The system then validates that the username and password are unique.
5. The system then validates that the password and password re-entry match.
6. The system then stores the username and password in the player's text file that represents the player's data.
7. The system then confirms success registration to the player.
8. The player is then redirected to the main menu.

Extensions or Alternate Flows:

4.1 If the username already exists, the system will display an error and will ask the player to pick a different one.

5.1 If the password and password re-entry do not match, the system will display an error and request the player to re-enter the password.

5.2 If username and password do not meet the requirements(spaces in between, too short, unallowed special characters) the system prompts the player to re-enter username and passwords.

Exceptions:

1. The system cannot write the new account data to players.txt due to file corruption.
2. During registration, if the client loses connection to the server, an account can't be created.

Related Use Cases: UC-02

Use Case ID: UC-02

Use Case Name: Log in

Relevant Requirements: 3.1.1, 3.1.2, 3.1.3

Primary Actor: Player

Pre-conditions:

1. There must be no active user session.

Post-conditions:

1. The user is logged into the system and can access its functionality.

Basic Flow or Main Scenario:

1. The user enters their username and password.
2. The system validates its credentials.
3. The system allows access to the system.

Extensions or Alternate Flows:

- 2.1 The system detects invalid credentials and allows the user to try logging in again.

Exceptions:

1. The log in function in the system is not available due to technical issues.

Related Use Cases: UC-03, UC-04, UC-05

Use Case ID: UC-03

Use Case Name: Player looks at their profile

Relevant Requirements: 3.1.2, 3.1.6, 3.2

Primary Actor: Player

Pre-conditions:

1. The player must have logged in successfully
2. Not currently within a game of blackjack

Post-conditions:

1. The profile UI will be displayed, allowing the user to see their information

Basic Flow or Main Scenario:

1. On the main menu, User clicks on user profile
2. The system responds, calling the appropriate module
3. The user's profile is displayed via the GUI, showcasing their information such as funds, game history, username, and password.
4. From here, the user can also "cash out" their account.

Extensions or Alternate Flows:

None

Exceptions:

1. The profile function is not available due to technical issues within the system.

Related Use Cases: UC-02, UC-04

Use Case ID: UC-04

Use Case Name: Player joins a blackjack game

Relevant Requirements: 3.1.2, 3.1.5, and 3.1.6

Primary Actor: Player

Pre-conditions:

1. Player successfully logged in
2. No active game session
3. Player has enough funds to match the minimum bet
4. Game table has at least one seat open

Post-conditions:

1. The player is at a blackjack table and can play a game once the round begins.
2. The number of players at the table is incremented by 1.
3. The player's funds will be updated according to whether they win the round or not.

Basic Flow or Main Scenario:

1. The player selects the game table overview, gaining access to all the blackjack tables currently open and the number of players within them.
2. The player chooses a table of their liking and joins it.
3. After the brief intermission between rounds ends, the player will then be entered into a game of blackjack.
4. The player selects the amount they would like to bet.
5. The dealer deals the cards to all players at the table and themselves.
6. The player chooses whatever action they deem best for their hand until the round is over.
7. Once the round is over, everyone's hands are revealed and the total of each person's hand is calculated. Between the player and the dealer, whoever is closest to 21 will win.
8. Depending on the outcome, the player's funds will be adjusted accordingly.

Extensions or Alternate Flows:

2.1 If the player tries joining a table that is full, the system will let them know that is the case and they will have to select another table.

4.1 If the player tries betting more than what they have, the system will let them know that they cannot and they will have to reenter the amount they'd like to wager.

Exceptions:

1. Technical issues prevent the user from accessing the blackjack table
2. The user disconnects in the middle of the blackjack game

Related Use Cases: UC-01, UC-02

Use Case ID: UC-05

Use Case Name: Exit from Blackjack Table

Relevant Requirements: 3.

Primary Actor: Player

Pre-conditions:

1. The player is successfully logged into the system.
2. The player is currently seated at a blackjack table that has a valid table ID.
3. The client application is connected to the server.

Post-conditions:

1. The player is at the table overview and can see all the joinable tables.
2. The user state is updated to reflect that the player is no longer within an active blackjack game.

Basic Flow or Main Scenario:

1. While at a blackjack table during the intermission window/period, the player clicks on the “Exit Table” button.
2. The system confirms if the player wants to leave with a message: “Are you sure you want to leave the table?”
3. The player confirms their choice, and the system removes them from the table.
4. The system frees up a seat once the player has been removed from the table. An updated player array is then displayed for other players to view
5. The player is then returned to the main menu.

Extensions or Alternate Flows:

2.1 If the player decides to cancel their choice to exit, the player remains at the table.

2.2 If the player attempts to exit during an active round, the system saves their updated balance to the player’s text file.

Exceptions:

1. The exit button is not functioning because of a network error.

Related Use Cases: UC-02

Use Case ID: UC-06

Use Case Name: Exit from Main Menu

Relevant Requirements: 3.1.6, 3.2

Primary Actor: Player

Pre-conditions:

1. The player is logged into the system or at the main menu.
2. The client application is running.

Post-conditions:

1. If the player exits during an active game, their updated balance is stored in the players’ text file.
2. If the player is in an active blackjack game, the system saves their updated balance to players’ text file.
3. The system closes the player’s connection to the server.
4. The client application shuts down.

Basic Flow or Main Scenario:

1. The player selects Exit from the main menu or closes the client application window.
2. If the player is in an active blackgame game and selects Exit, the system has the player’s hand to stand and update their player balance at the end.
3. The system closes the player’s connection to the server.
4. Client application shuts down.

Extensions or Alternate Flows:

2.1 If the player is not in an active game, the system immediately closes without needing to update balance.

2.2 If the player selects Exit on accident or mistake, the system will always prompt in every situation of a player selecting exit with a confirmation message displaying “Are you sure you want to exit?”.

Exceptions:

1. The exit button is not functioning because of a network error.

Related Use Cases: UC-02

Use Case ID: UC-07

Use Case Name: Place Bet

Relevant Requirements: 3.1.2.9, 3.1.5.6, 3.1.5.7

Primary Actor: Player

Pre-conditions:

1. The player must be logged into the system
2. The player must have joined an active blackjack table
3. The game’s round must be in the betting phase
4. The player must have sufficient balance to cover their bet amount
5. The bet amount must be within the table’s minimum (\$5) and maximum (\$50,000)

Post-conditions:

1. The player’s bet is set to the specified amount.
2. The bet amount must be deducted from the player’s account balance.
3. The player’s state is updated to “Bet Placed”.
4. The player’s bet amount is added to the current round’s total pot.

Basic Flow or Main Scenario:

1. Player selects a bet amount.
2. System validates the bet is within limits and player has sufficient funds.
3. System deducts the amount from the player's balance and sets it as their current bet.
4. System confirms the bet is placed.

Extensions or Alternate Flows:

2.1 If the player bets an amount higher than their current balance, the system displays an “Insufficient Funds” error and prompts them for a lower amount.

2.2 If the player bets outside of the \$5-\$50,000 range, the system displays an “Invalid Bet Amount” error and prompts them to enter an amount within the acceptable range.

Exceptions:

1. The system’s betting becomes unresponsive due to a network error
2. The game state changes before the bet is confirmed.

Related Use Cases: UC-04, UC-08, UC-09

Use Case ID: UC-08

Use Case Name: Hit

Relevant Requirements: 3.1.2.8

Primary Actor: Player

Pre-conditions:

1. The player is in an active game round.
2. The player's hand value is less than 21.
3. It is the player's turn.
4. The player has not already stood or busted.

Post-conditions:

1. The player receives an additional card
2. The player's current hand value is recalculated
3. If the updated hand value is above 21, the player busts and loses their bet

Basic Flow or Main Scenario:

1. Player selects the "Hit" action
2. The system deals one card to the player
3. The system updates the player's hand, and the total value is recalculated
4. If the hand value is less than 21, the player's turn continues.

Extensions or Alternate Flows:

- 4.1 If the hand value exceeds 21, the player busts and their turn ends.

Exceptions:

1. A network error occurs during the player's turn.
2. The system fails to draw a card from the shoe.

Related Use Cases: UC-07, UC-09, UC-12

Use Case ID: UC-09

Use Case Name: Stand

Relevant Requirements: 3.1.2.8

Primary Actor: Player

Pre-conditions:

1. The player is in an active game round.
2. The player's hand value is less than 21.
3. It is the player's turn.

Post-conditions:

1. The player stays with the cards that have, no additional cards are given

Basic Flow or Main Scenario:

1. Player selects the "Stand" action
2. The player's turn is ended, the system goes to the next player.

Extensions or Alternate Flows:

1. No external or alternate flows

Exceptions:

1. The system experiences issues with the network, causing the player actions to not function properly or not register with the server.

Related Use Cases: UC-04

Use Case ID: UC-10

Use Case Name: Double down

Relevant Requirements: 3.1.2.8

Primary Actor: Player

Pre-conditions:

1. The player is logged into the system.
2. The player has joined a Blackjack table.
3. The player is in an active round of Blackjack.

Post-conditions:

1. The player's turn status will be shown as Double Down.
2. The player's bet is updated to be double their initial wager.
3. The dealer gives the player ONLY one card because they have chosen to double down.
4. The player's hand is updated to reflect the value including their third card.

Basic Flow or Main Scenario:

1. The player joins a blackjack table and waits until the round begins.
2. Once the round begins and the dealer has dealt everyone their cards, the player can then select "Double Down" if they are confident that they can win with the next card.
3. The dealer gives the player their only additional card
4. The player waits for everyone's turns to end
5. Everyone reveals their hands and the outcome of the round is decided.

Extensions or Alternate Flows:

1.2 If the player does not have enough funds to match double their initial wage, then they will not be able to select double down

5.1 If the player wins the round, the payout from their bet will be added to their funds. Otherwise, they have lost the round and the funds are removed from their account.

Exceptions:

1. The system experiences issues with the network, causing the player actions to not function properly or not register with the server.

Related Use Cases: UC-04, UC-07

Use Case ID: UC-11

Use Case Name: Split

Relevant Requirements:

Primary Actor: Player

Pre-conditions:

1. The player has logged into the system successfully.
2. The player has joined a blackjack table.
3. The player is in an active round of Blackjack.
4. The player's initial two cards are of equal value

Post-conditions:

1. The player is now managing two different hands.
2. The second hand's wager is the same as the first wager, effectively doubling the initial wager.

Basic Flow or Main Scenario:

1. The player joins a blackjack table and waits until the round begins.
2. The round begins and the dealer gives everyone their cards.
3. The player receives a pair of cards with equal value
4. The player chooses to "Split" which doubles their wager in order to split this pair of equal cards into two separate hands.
5. The two hands (each with 1 card at the moment) are then dealt their second card by the dealer.
6. The Blackjack round proceeds as normal with the player that manages the split hands having two turns, one for each hand that they have.
7. At the end of the round, everyone's cards are revealed and the outcome for each player is decided.

Extensions or Alternate Flows:

- 4.1 If the player does not have enough funds to match double their initial wager, then they cannot split even if they have a pair of cards with the same value.

Exceptions:

1. The system experiences issues with the network, causing the player actions to not function properly or not register with the server.

Related Use Cases: UC-04, UC-07

Use Case ID: UC-12

Use Case Name: Dealer Plays Hand

Relevant Requirements: 3.1.2.8

Primary Actor: Dealer

Pre-conditions:

1. The dealer plays with at least 1 player.
2. The dealer is in an active round of Blackjack.

Post-conditions:

1. The player will win or lose the round.
2. The dealer will win or lose the round.
3. The dealer's hand is updated to reflect the value of it.

Basic Flow or Main Scenario:

1. The player joins a blackjack table and waits until the round begins.
2. Once the round begins and the dealer has dealt everyone their cards, the player selects the actions of its hand.
3. The dealer deals and the cards of the players.
4. When all players have stood or busted, the dealer opens up his second card.
5. If the dealer busts, all players who chose to stand automatically win the round. If the dealer reaches 21, he wins. If the dealer's hand is less than 21, the game compares his total to each player's hand. Any player with a higher total than the dealer wins and receives the corresponding payout, while players with lower totals lose their bets. If a player and the dealer have the same total, the round is a push, and the player's bet is returned.
6. The round ends and the dealer clears all the cards.

Extensions or Alternate Flows:

1. If the dealer's first card is an Ace, players are offered insurance before the dealer checks for blackjack.
2. The player places a side bet.

Exceptions:

1. The system experiences issues with the network, causing the player actions to not function properly or not register with the server.

Related Use Cases: UC-04, UC-07, UC-08, UC-09, UC-10, UC-11

Use Case ID: UC-13

Use Case Name: Resolve Round

Relevant Requirements: 3.1.3.2, 3.1.3.4

Primary Actor: Dealer

Pre-conditions:

1. An active round of Blackjack is underway.
2. All players at the blackjack table have finished their turns and are at the “Bust” or “Stand” status.

Post-conditions:

1. Each player’s hand is compared to the dealer’s and it is determined whether they have won or not.
2. The player’s funds are updated according to the outcome
3. The blackjack table enters the intermission period until the next round begins.

Basic Flow or Main Scenario:

1. The Blackjack table has at least one player and the round begins.
2. After the player(s)’ turns are over, their hands are revealed and the total value is counted.
3. The value of each player’s hand is compared to that of the dealer.
4. If the player’s hand value is closer to 21 than the dealer's or the dealer has “Bust” then the player wins.

Extensions or Alternate Flows:

1. If any player disconnects during the round and does not reconnect, the dealer will automatically stand for the player with the value that they had prior to the disconnect. The game proceeds as normal.

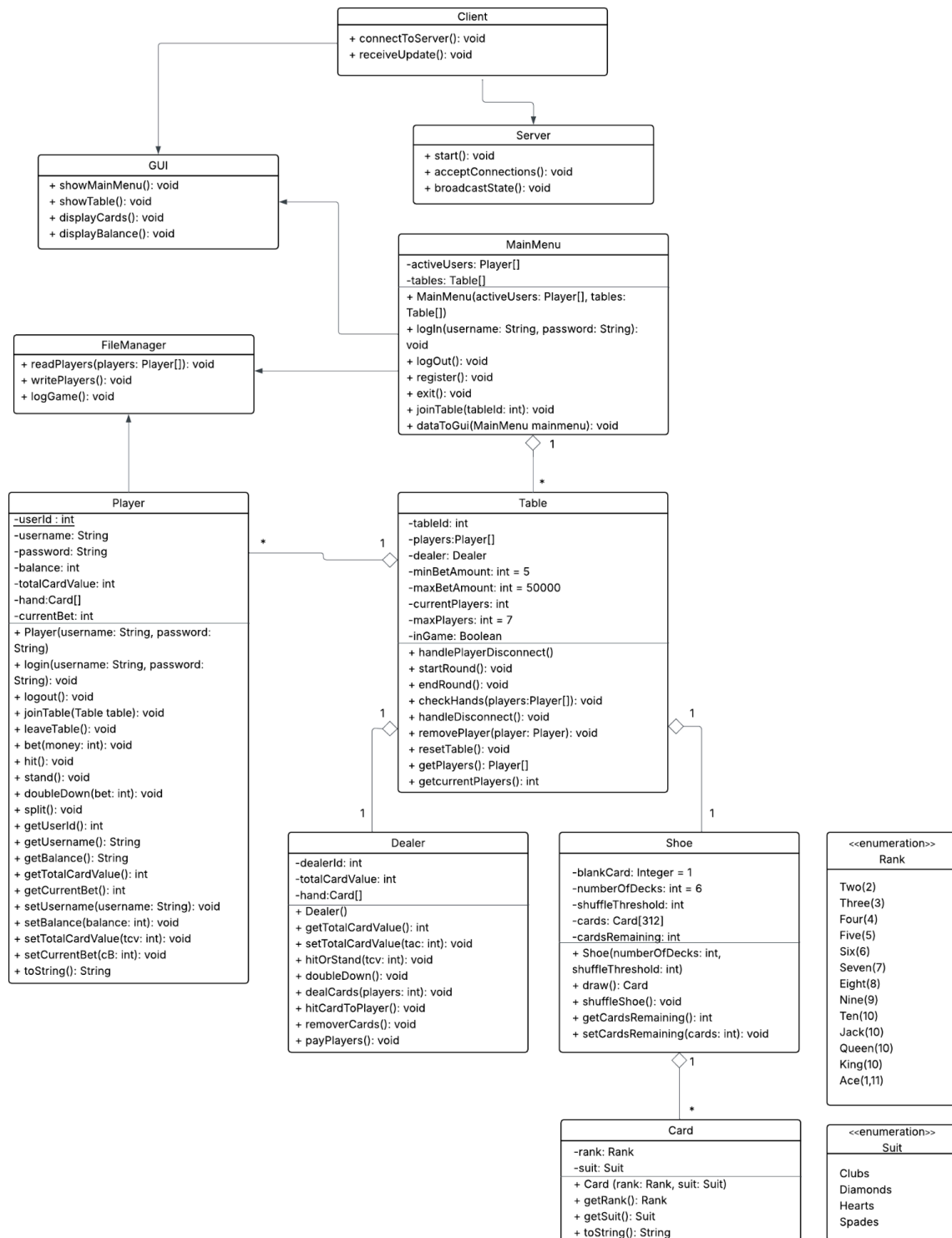
Exceptions:

1. The system experiences issues and thus cannot process the resolution of the Blackjack round.

Related Use Cases: UC-04, Use Cases 7-12

CS 401 Blackjack Game Project

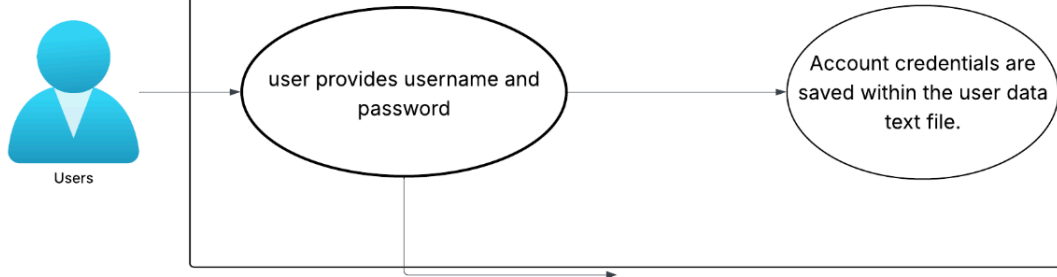
UML Class Diagrams Document



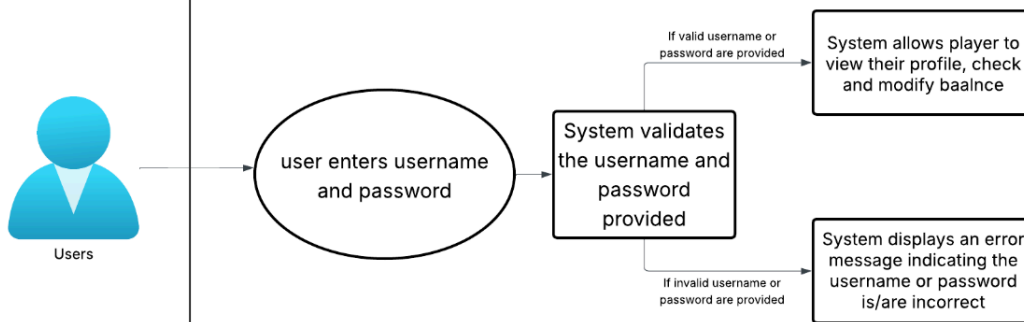
CS 401 Blackjack Game Project

UML Use Case Diagrams Document

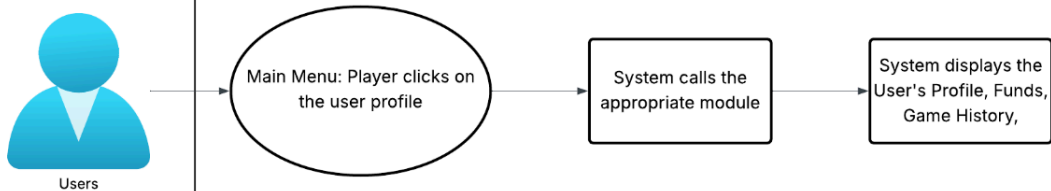
Use Case: Player registers a new account



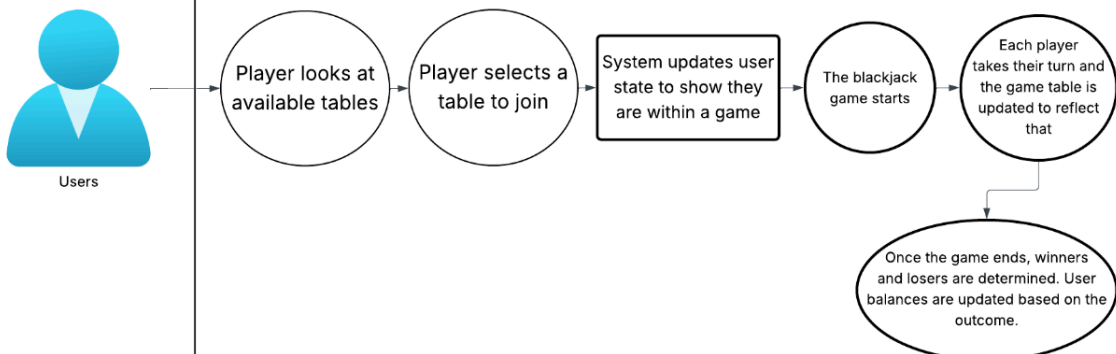
Use Case: Player logs in



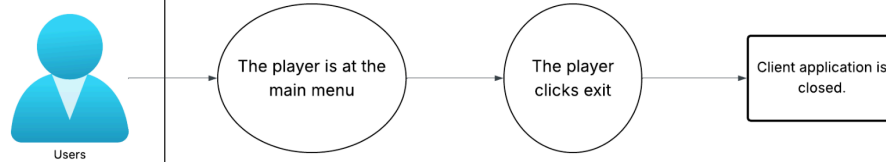
Use Case: Player checks profile



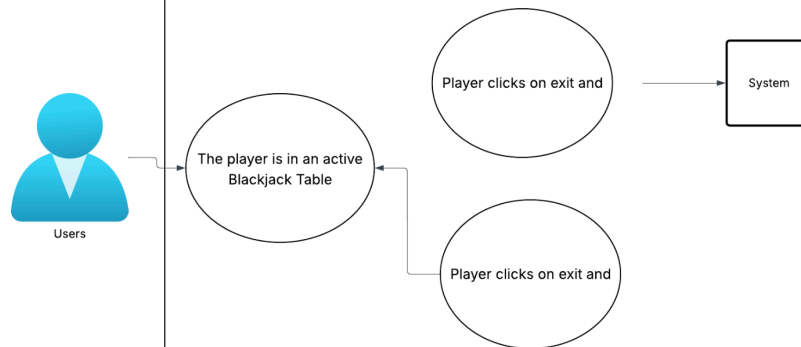
Use Case: Player joins a blackjack game



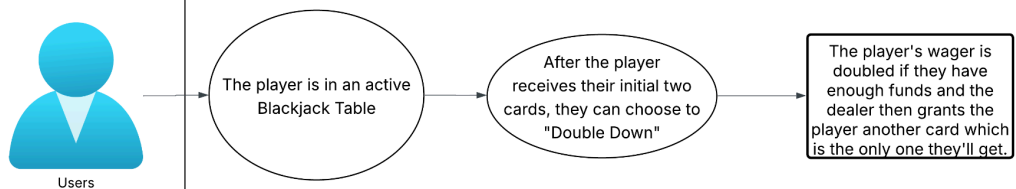
Use Case: Player exits the main menu of the Blackjack Program



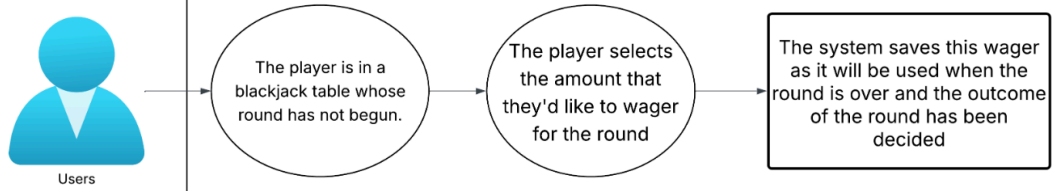
Use Case: Player exits a Blackjack table



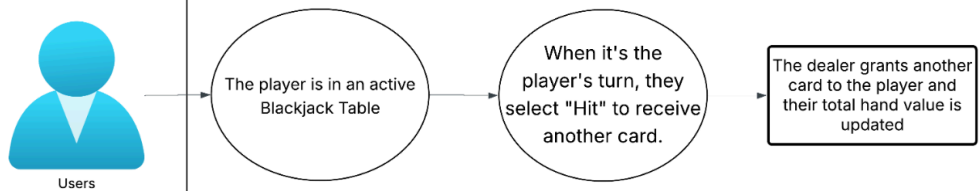
Use Case: Player chooses "Double Down"



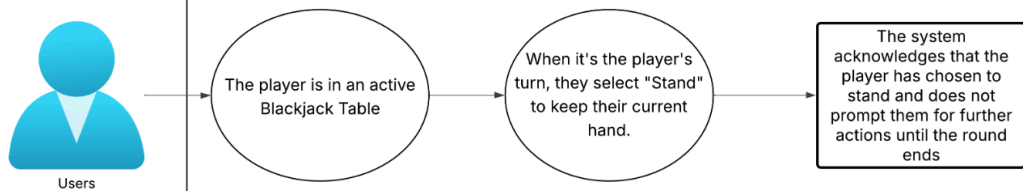
Use Case: Player places a Bet



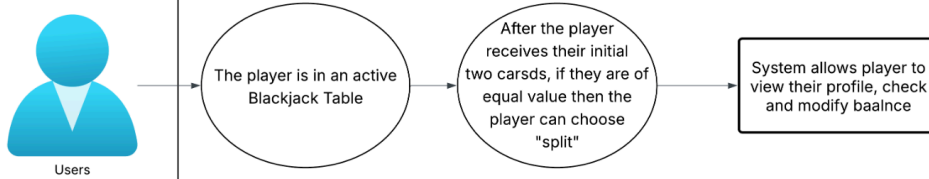
Use Case: Player chooses "Hit"



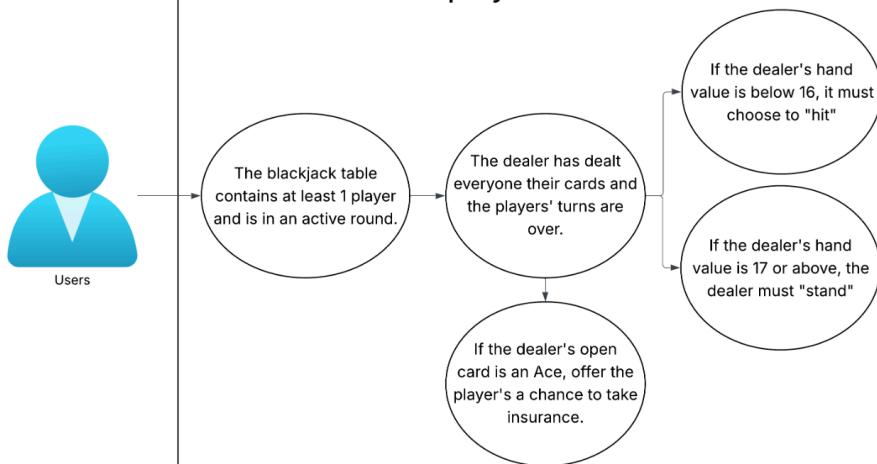
Use Case: Player chooses "Stand"



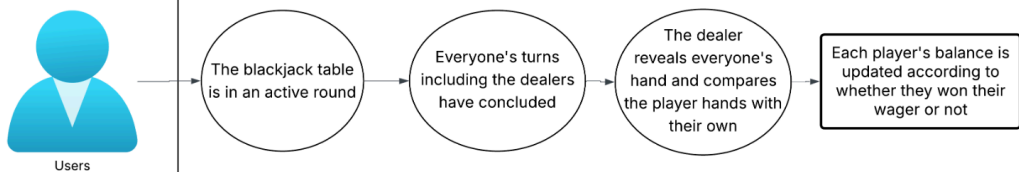
Use Case: Player chooses "Split"



Use Case: Dealer plays hand

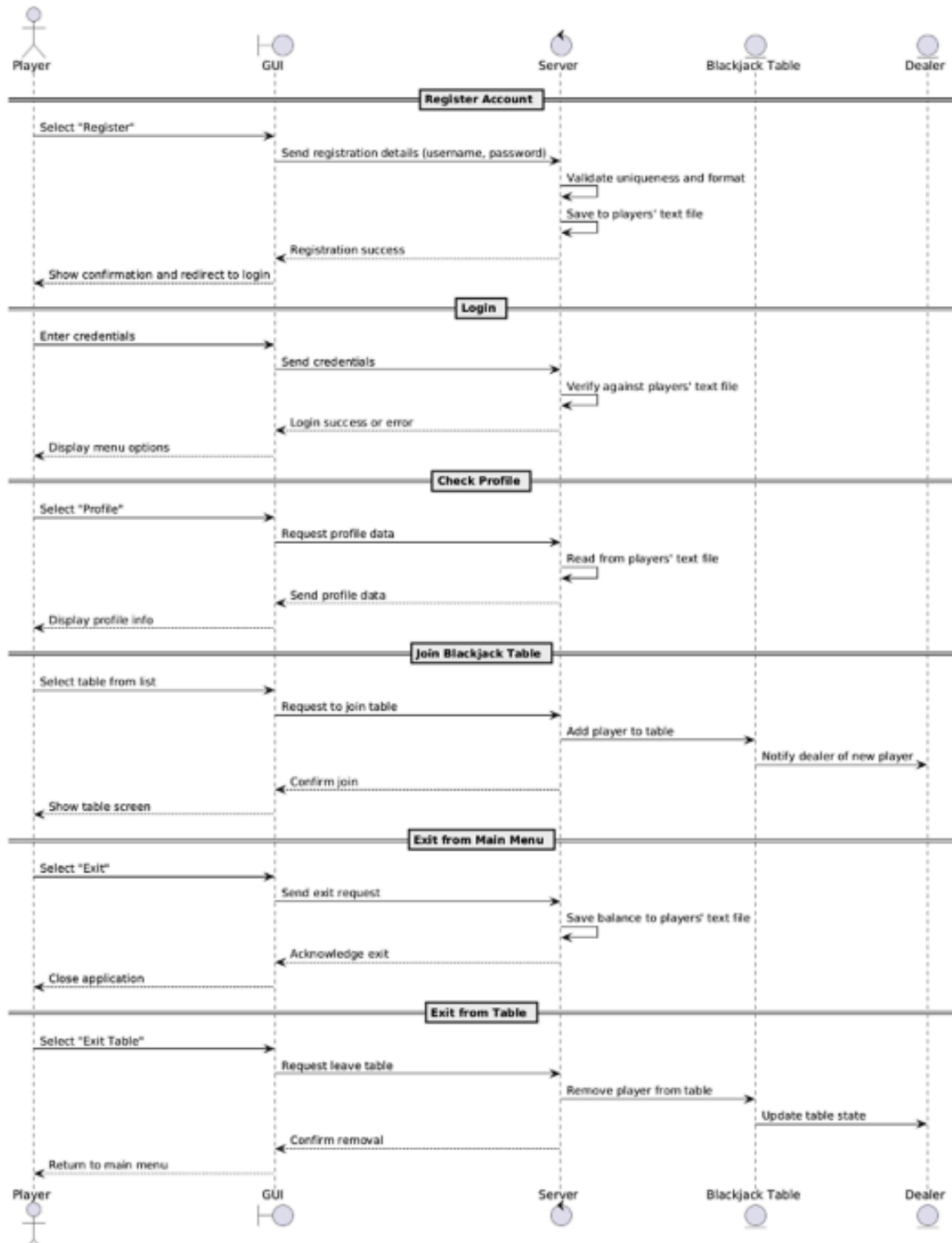


Use Case: Dealer resolves round



CS 401 Blackjack Game Project

UML Sequence Diagram Document



CS 401 Blackjack Game Project

Project Schedule Chart

Multiplayer Blackjack Game Project Timeline

Period Highlight: 16

Plan Duration Actual Start % Complete

