# Assignment #2 – Multi-Player BlackJack Game

Design Specification Document

Daniel Jensen

Chun Yin Lee

Aastha Sharma

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 07/14/2022 | 1.0.0 | Initial Version | Chun Yin Lee |
| 07/16/2022 | 1.0.1 | Updates to text, added 1.4 | Daniel Jensen |
| 07/17/2022 | 1.0.2 | Updates 3.2.3 | Daniel Jensen |
| 07/18/2022 | 1.1.0 | Formatting updates | Daniel Jensen |
| 07/18/2022 | 1.1.1 | Updated 3.1.1 | Daniel Jensen |
| 07/18/2022 | 1.1.2 | Added UML class design and flows | Aastha Sharma |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction [4](#__RefHeading___Toc19440719)

1.1. Goals and Objectives [4](#__RefHeading___Toc19440720)

1.2. Statement of Scope [4](#__RefHeading___Toc19440721)

1.3. Software Context [4](#__RefHeading___Toc19440722)

1.4. Major Constraints [4](#__RefHeading___Toc19440723)

2. Data Design [4](#__RefHeading___Toc19440724)

2.1. Client side [5](#__RefHeading___Toc19440725)

2.2. Server side [5](#__RefHeading___Toc19440726)

3. Architectural and Component-level Design [6](#__RefHeading___Toc19440730)

3.1. Program Structure [6](#__RefHeading___Toc19440731)

3.2. Description of Client [6](#__RefHeading___Toc19440736)

3.2.1 Client processing narrative…………………………………………………………………….……..6

3.2.2 Client interface description……………………………………………………..………….………..7

3.2.3 Client processing details………………………………………………………………….…….……..7

3.3 Description of Server………………………………………………………………….…………….……..7

3.3.1 Software Interface Description………………………………………………..…………….……..7

3.3.3 Internal Interfaces…………………………………………………………………..……….…………..7

3.3.4 Human Interfaces………………………………………………………..….…..…..…………………..7

**4. User Interface Design**………………………………………………………………..….…………………..8

**5. Restrictions, Limitations, and constraints**……………….…………………..………….………..8

**6. Testing Issues**………………………………………………….…………………………..………….………..8

**7. Appendices**……………………………………………………………….……………….…..….……………..8

**1. Introduction**

**1.1. Goals and Objectives**

This document describes important aspects of the implementation of the **#login**, **#card**, **#deck**, **#gameroom** and **#onlinebank** commands of the Multi-Player Blackjack.

**1.2. Statement of Scope**

Decisions in this document are made based on the following priorities (most important first): Maintainability, Usability, Portability, Efficiency

**1.3** **Software Context**

This Multi-Player Blackjack game combines several subprograms: login page, game lobby, game room and online bank. Login page is for the player and dealer to log in. After that, player can access their online bank to add/withdraw funds or they can get in the queue to join a game. Finally, they can play in the game room. Card and deck are the elements for the blackjack game.

**1.4.** **Major Constraints**

**Issue 1:** Where should account data be stored?

**Option 1.1**: Store the information in a file on the client side for the client application to validate against?

**Option 1.2**: Store the information in a file on the server side for the server application to validate against?

**Decision**: Per section 2.5.3 of the Requirements Specification document, the login data file will be stored on the server as it will be a function of the sever to limit logins to one client per user. Per section 3.2 of the Requirements Specification document, account balance and game play logs will be stored on the server as well.

**2. Data Design**

1. Login page – for the player and the dealer to log in
   1. It requests the user to enter a username and password. It compares against a comma separated text file which contains “Id”, “Name”, “Username”, “Password”, “User Type”.
2. Card – a single card represents for a rank and suit
   1. It combines from one of the thirteen ranks (from 2 to 10, Jack, Queen, King, and Ace), and one of the four suits (diamonds, clubs, hearts, and spades). Therefore, each item will contain one rank and one suit. No duplicates will occur.
3. Deck—a deck combines all 52 cards
   1. It is a list of all 52 cards in a particular (random) order, and the number of cards will decrease by dealing out cards to dealer and player hands. When the game is done, it will reset as a full deck and be shuffled.
4. Online Bank – for the players to add/withdraw the funds to their account
   1. It requests a text file to save the balance of their account every time while they add/ withdraw the funds from their account. Also, during the game, the system will load this record to let the player know what their current balance is and will update when the game ends.
5. Game room – for the player to play game
   1. Once the server will receive the signal from the player to hit the button of “play game”, it will create a game room and notify dealer to get in the game room. Once there has five people, the game will start instantly, or when the player has waited for more than three minutes, the game will also start. Those players will automatically enter to game room to play against the dealer.
6. Game - the game based on our rules
   1. It will receive the instruction from the player to hit, stand or leave game during the game play, and make the decision- play again or leave when the game end.

**2.1 Client side**

1. Once the player logs in to the game, it passes his username and password to the server to verify his identity.
2. When the player decides to add/withdraw funds, it will pass the instruction to the server and update the balance.
3. When the player clicks to play game, the server will reserve a place for him until the game starts.
4. When the game starts, the player will place a bet which is sent to the server.
5. When all bets are places, the dealer will tell the server to deal. This will give each user will a hand of cards from the server.
6. When the player receives cards from dealer, the player can decide to either hit or stand, and send the instruction back to server until they bust, score 21, or stand.
7. When the game ends, the player can decide to play again or leave, and the client can see the updated balance from the server.
8. The dealer will receive instructions from the server to join a game and click to join.
9. The dealer will give instructions to end the game and settle all bets.

**2.2 Server side**

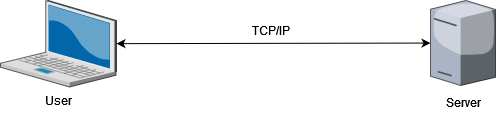
1. The server will verify the username and password. If it is real, it will allow the player to log in to play game, if not, decline it. If the player tries thirds time and still fail, the server banned this account until verify
2. When received the add funds or withdraw funds, the server will update the balance
3. When receive the “play game “instruction from the player, the system will setup a game room and arrange a dealer ( send a notification to the dealer that request him to get in a game to ready)
4. When the server receives the dealer signal to start the game, the server will give out card to the dealer and player hands, and ask for the instruction from each player
5. When all players done their decision for hit/stand, it will send card to the dealer until it hit or over 17.
6. When the server received the instruction from the dealer to end game and settle the bet, the server will automatically settle the bet and send notification to the player to check if they still continue or done
7. When the last player decides to leave, the server will send notification to the dealer that the game is done and ready for next game.

**3. Architectural and Component-level Design**

**3.1 Program Structure**

The Multi-player Blackjack runs as a client-server application and save all record into three different text files; one is for player and dealer login details; one is for the balance, and one is for the game record

**3.1.1 Architecture diagram**

****

**3.2 Description of Client**

**3.2.1 Client processing narrative** **(****functions and processes described)**

1. login() to log into the serve, sending username and password (string) to server
2. addFunds() to add money into the account, sending some amount (int) to server
3. withdrawFunds() to take out the money from the account, sending some amount (int) to server
4. playGame() to start the game, sending a signal (Boolean)to server
5. hit() to call for an extra card, sending a signal (Boolean)to server
6. stand() to refuse for an extra card, sending a signal (Boolean)to server
7. playAgain() to continuous a exiting game, sending a signal (Boolean)to server
8. leave() to leave a exiting game, sending a signal (Boolean)to server

**3.2.2 Client** **interface description** **(input and output interfaces described)**

Everything will use a graphical user interface (GUI), most of the choices are using button, and the client need to type letters and numbers only when login and add/withdraw funds

**3.2.3 Client processing details** **(algorithmic description)**

Client will take user input and send those to the server via message objects.

**3.3 Description of Server**

**3.3.1 Software Interface Description**

Everything is using graphical user interface (GUI), first, it will pop up a login page for the client to login, after login, the login page will close and pop up the main page for the client to either get into the game room or online bank to add/withdraw funds

**3.3.2 External Interfaces**

1. The system must provide an interface to our payment system so that players can add funds and withdraw funds as needed. The interface is to be in a comma-separated text file containing the following fields: player id, current balance, amount of money being moved, action. Where “action” is whether the user has added or withdrawn the funds. The file will be exported nightly and will contain new transactions only
2. The system must provide an interface to our logging system so that all game moves and cards are logged to prevent cheating. The interface is to be in a comma-separated text file containing the following fields: player id, current balance, amount bet, status in game (in or out), action (hit or stay), cards in hand. The file will be exported after every game session (table) closes.

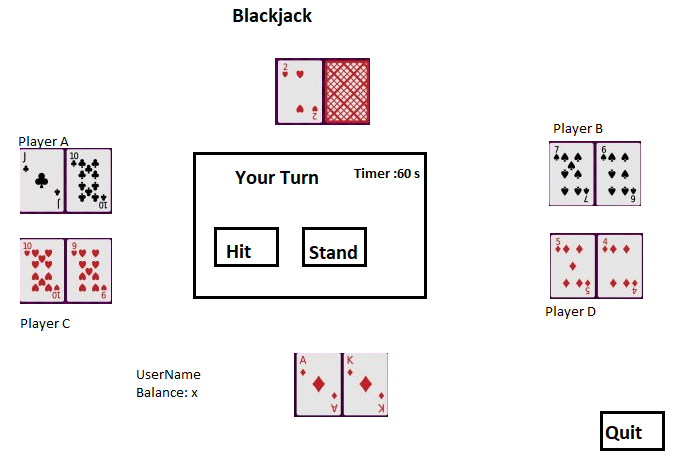
**3.3.3 Internal Interfaces**

The system must process a data-feed from the logging system so that all game moves and cards played are stored to prevent cheating. The data feed is to be in a comma-separated text file containing the following fields: player id, current balance, amount bet, status in game (in or out), action (hit or stay), cards in hand. The file will be exported after every game session (table) closes.

**3.3.4 Human Interfaces**

1. Must have a keyboard to input value.
2. Must have a mouse to choose option.
3. Must have a monitor to view the GUI.

**4. User Interface Design**



**5. Restrictions, Limitations, and constraints**

1. All players must register before playing.
2. The age of the player must be 21 or over.
3. If the player makes no action within 1 minute of being prompted to place a bet, the system will assume the player has left the table.
4. If the player makes no action within 1 minute of being prompted to hit or stand, the system will assume the player has forfeited their hand.
5. If the player makes no action within 1 minute after the game ends, the system will assume the player has decided to leave the table.

**6. Testing Issues**

**7. Appendices**

Diagram

Description automatically generatedChart

Description automatically generatedDiagram, schematic

Description automatically generatedDiagram

Description automatically generated