SE210: Software Requirements Specification
for
Better Checkers

Team 4

By Yegeon Seo, Jakob Au,
Jared Balakrishnan, Matthew D'Ulisse

8 December 2019

Table of Contents

1 Introduction

This document is a software requirements specification for Better Checkers, a three month software requirements project completed in the SE210 (Software Specification and Design) course at Drexel University. This document consists of the overall description of the project and system features, including functional and non-functional requirements. We aim to give the users better understanding and details of the Better Checkers application, including but not limited to: product features, hardware and software requirements, and detailed system features.

1.1 Purpose

The purpose of this document is to specify the requirements and preview some elements of the Better Checkers application. It is the user's assurance that the developers understand the required software behavior and can address issues or problems found.

Better Checkers consists of the Client and the Server.

> **Server**: a database of players and their information including IDs, usernames, wins, and losses, number of games played, rank number
> Server shall also save the current state of the board game for the client to access and display to the user, especially in the case of a temporary disconnection
>
> **Client**: is a system that connects to the server. It displays information stored by the server to the user and transfers information from the user's actions to the server.

1.2 Project Scope

Better Checkers is a web browser application that simulates playing a game of checkers between two players.

Better Checkers aims to combine the advantages of the internet with the ease-of-use of real-life checkers. Being able to simply bring out a board and play is essential to the checkers' experience, and being able to do that online with your friends or anyone in the world is an advantage the internet gives.

The goal is to make playing checkers on the Better Checkers application the same as playing on the streets against all kinds of people, preserving the competitive and challenging spirit while taking the advantages of the internet.

Below are some screenshots of the Better Checkers wireframe.

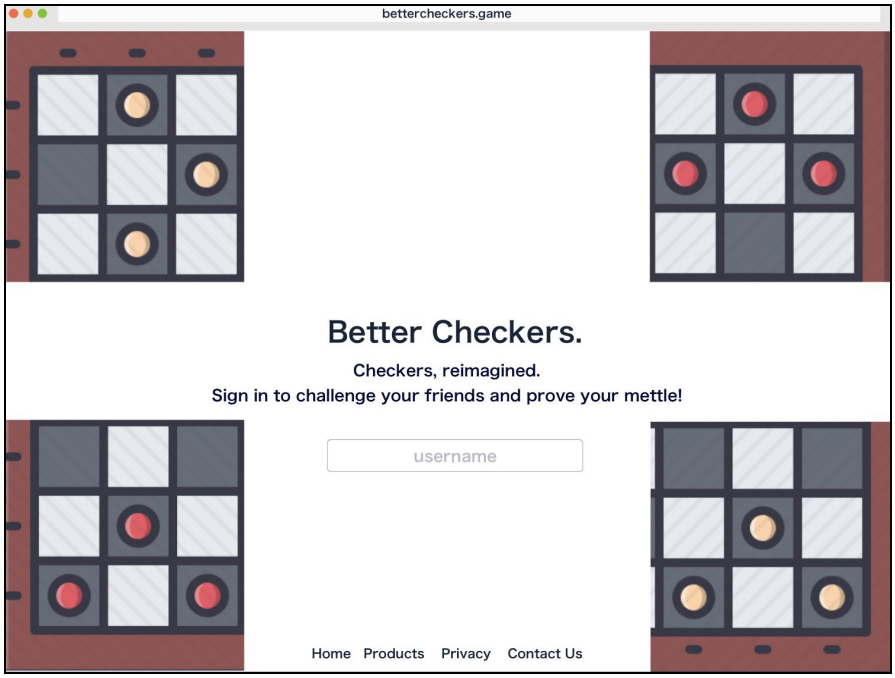

Figure 1.1 -Sample Login

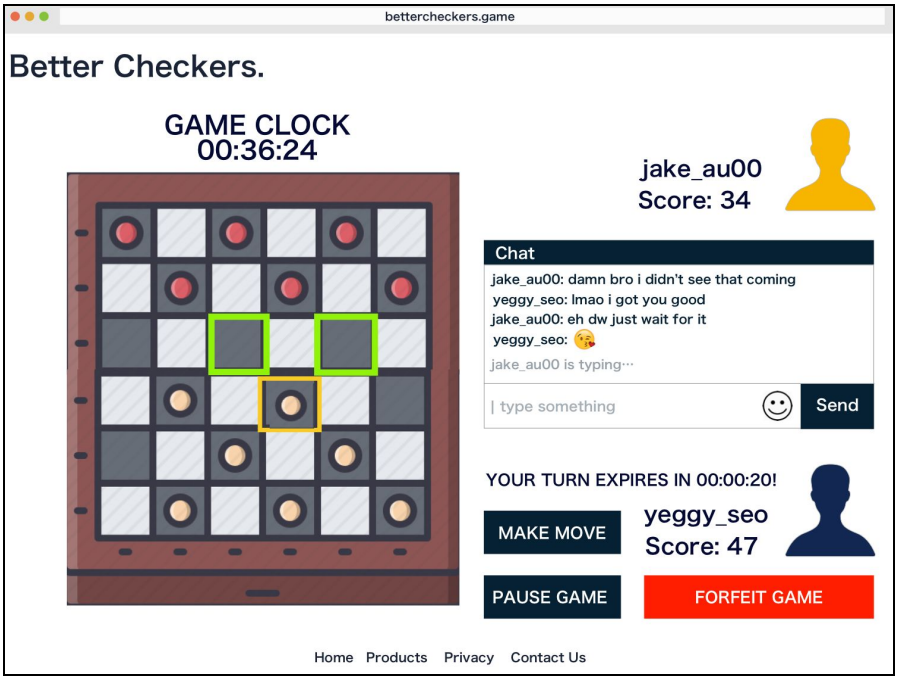

Figure 1.2 -Sample Gameplay

1.3 Intended Audience

This document is intended for a software developer, programmer, user, and tester that needs information about the specific requirements and system architecture of Better Checkers.

Developer: who wants to create the Better Checkers program.

System analysts, Requirements analysts: maintain the system, maintain and review the requirements

User: who wants to review the diagrams and requirements to better understand the inside structure.

Tester: who needs this document to compare to the existing program the specifications and requirements.

Project Managers: manage the development of the system and ensure work is adhering to the requirements

1.4 References

This document contains the necessary requirements based on the IEEE Standard for Software Requirements Specification (IEEE 830-1998) and based on the framework and works by the SRS-iTest document.

IEEE 830-1998:
http://www.ieee.org/publications_standards/index.html

2 Overall Description

Better Checkers is a free web browser application for checkers players of all skill levels. Beginners are provided with rules and a ranked system that matches them with similarly ranked players. Experts avoid reading the how-to guides and get right into ranked play against other players. Players can also play against a computer, where there are three levels of difficulty to choose from.

2.1 Product Perspective

The goal of the “Better Checkers” system is to allow a user to play checkers against a computer, a specific opponent, or a random opponent. The user does this without creating an account, instead, a unique ID is created for the user each time they connect to the system. The user is still able to enter a display name that gets stored in the database of players and shown to other users. The database of players also stores the number of users currently online so the info can be displayed to the users.

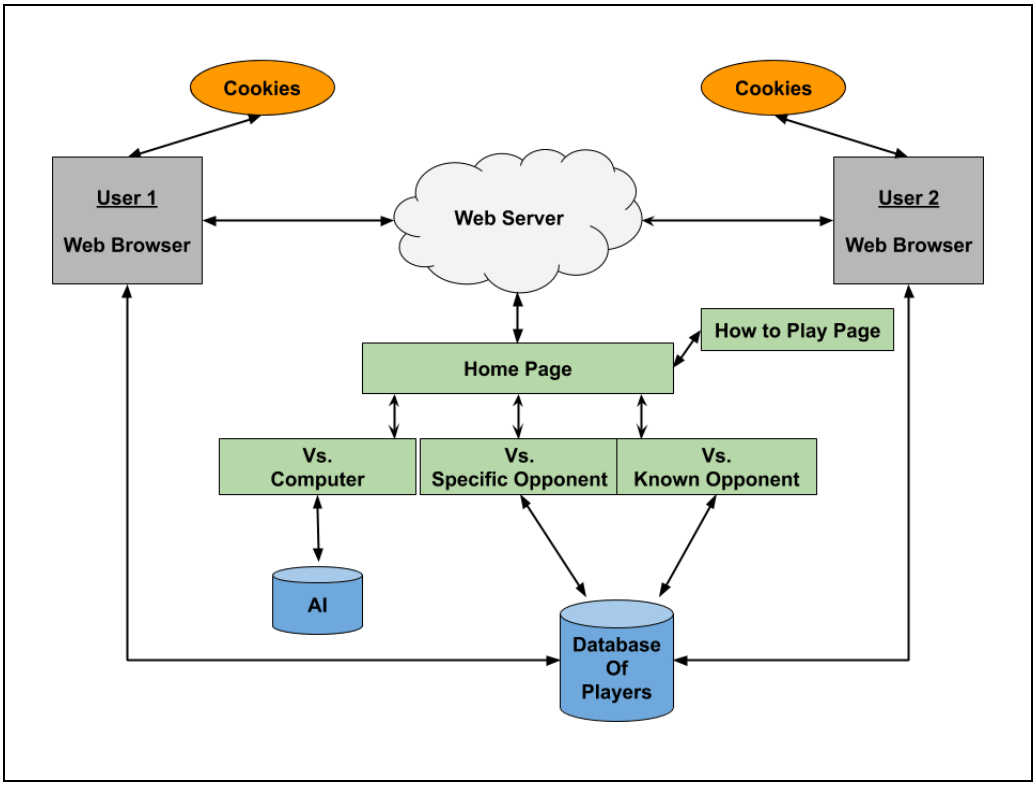

Figure 2.1 -High-Level Architecture Diagram

2.2 Product Features

The major features of this program, some also shown in Figure 2.1, are below:

- **One-time login**: After each session, the user's username and a unique ID is deleted from the database, forever.

- **Ranking**: wins and losses of each session are stored and determines a rank for each User. Ranks also determine a match against the next ranked opponents.

- **Chat**: Offers the option to send messages to your opponent in gameplay against another User.

- **Gameplay**: In each game, a User is put into a fair online gameplay experience offering a competitive game of checkers.

- **How to Play Page**: When a user connects to the webserver they are brought to the home page. This is the main page where the user interacts with the system. One option for the user is to go to a page that explains the rules of checkers. This is a static page for the user to read. The user is able to return to the home page from this screen.

- **Versus a Computer**: On the home page, the 3 major options for a user to take are the 3 game modes of Better Checkers: versing a computer, versing a specific opponent, and versing a random opponent. If the user chooses to verse the computer, then they are put up against an AI. The user also inputs a difficulty for the AI: easy, medium, or hard. The AI takes in the current game state and outputs a move to the board.

- **Versus a Specific Opponent**: If the user chooses to verse a specific opponent, they must input that opponent's unique ID. The ID is checked against the database to find the player with that ID, who then receives a notification. If the challenged player accepts, both are paired together and a game can begin. Pre-made messages can be sent between the players during the game.

- **Versus a Random Opponent**: If the user chooses to verse a random opponent, the player's rank in the database of players is used to find a similarly ranked player who is also searching for an opponent. The user is able to verse the computer while a match is being found. Once a suitable match is found, both users are asked to accept and are then paired together. Pre-made messages can be sent between the players during the game.

- **Database of Players**: Games between online players are stored in the database of players. The game state is kept track of, as well as the IDs of the players in the game. If one or both players lose connection, they are able to connect back into the game by pulling the ID from the cookies. Once the player spends a certain amount of time off the website, this ID is deleted so a new one can be generated upon the next login.

2.3 Definitions

These definitions are used throughout the document.

User: The primary actor in the application. Physical Actor.

Client: The client connects to the server, takes input from the user, and outputs information to the User. The secondary actor in the application. System Actor.

Server: The server interacts with the client and stores data in a database. System Actor.

2.4 Operating Environment

Better Checkers shall be designed to be a very minimal web application that is capable of running on any web browser.

Better Checkers aims to embody the phrase ”Lock and Load”. That is, players that wish to make use of the application will not be required to install any piece of software on their computer; they will be able to play a game of checkers online simply by proceeding to the website.

The application is designed to be responsive, which makes players that use mobile devices, tablet computers, or desktop browsers possible to enjoy the game without facing any problems regarding the responsiveness of the web application (such as resolution breaking, choppy graphics, etc).

2.5 Development Environment

As mentioned in section 2.4, the Better Checkers system is designed to be a responsive web application. Thus, the MERN (MongoDB, Express.js, ReactJS, Node.js) stack shall be suitable as a main development tool for the application. Note that all the tools used in the MERN stack require a strong foundation in JavaScript, which means proficiency in JavaScript is required for all developers.

The frontend of the website shall be constructed using ReactJS, which provides strong advantages in data flow control, performance, testability, and more. In addition, SASS (or CSS) shall be the main styling language for this application. SASS shall play the main role in making the website responsive, as well as constructing the general style of the entire website.

The backend of the application shall be constructed with the rest of the MERN stack. MongoDB (with Mongoose) shall be the main database, and Express.js and Node.js shall be the main backend tools. Express.js shall be used to process most of the HTTP related issues, such as parsing the payload, cookies, or storing sessions. Other server-side issues that are not mentioned here shall also be handled by Node.js and Express.js.

2.6 Design and Implementation Constraints

The challenges in developing the product include setting up the server using the stack. Due to the nature of web applications, the development can be affected by web server responsiveness or load, geographic location, or expected network delay.

2.7 Assumptions and Dependencies

The product would build on a preexisting create-react-app boilerplate. In this regard, some necessary tools can be node.js and node package manager (npm). Sufficient knowledge of full-stack development is required and assumed, as well as knowledge of javascript. Knowledge of the backend process, such as handling synchronous/asynchronous and HTTP calls will be helpful for ease of development. Lastly, since create-react-app supports freedom of code design, developer’s sufficient knowledge of programming design patterns is assumed.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 Login

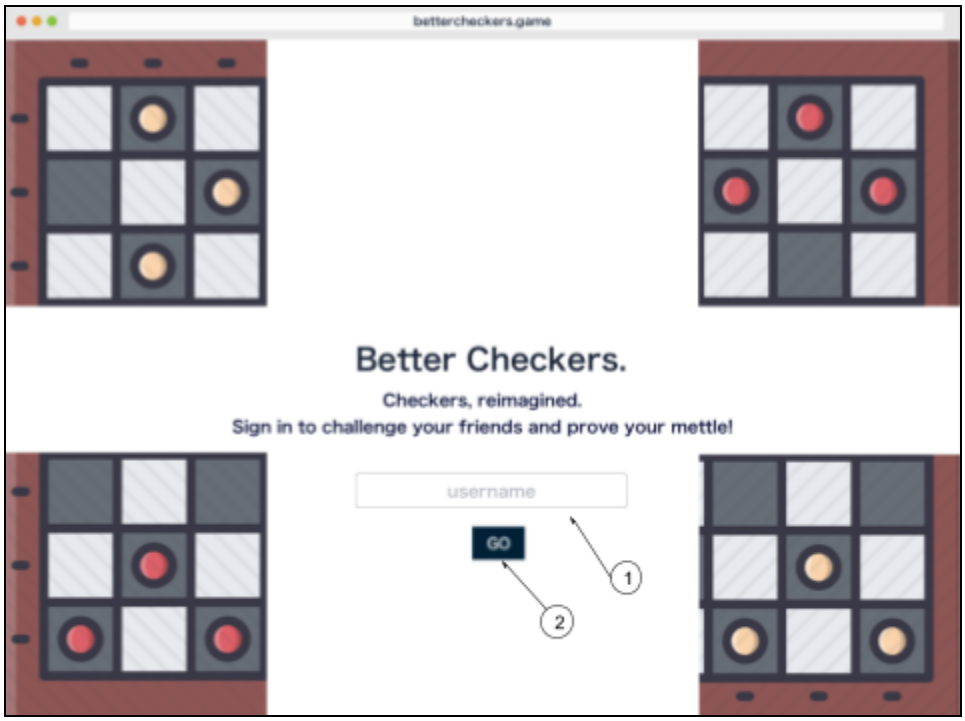

Figure 3.1 Login Page

Option 1 Username input window.

Option 2 goes to the landing page and starts a game with a computer on easy difficulty.

3.1.1.2 Landing Page

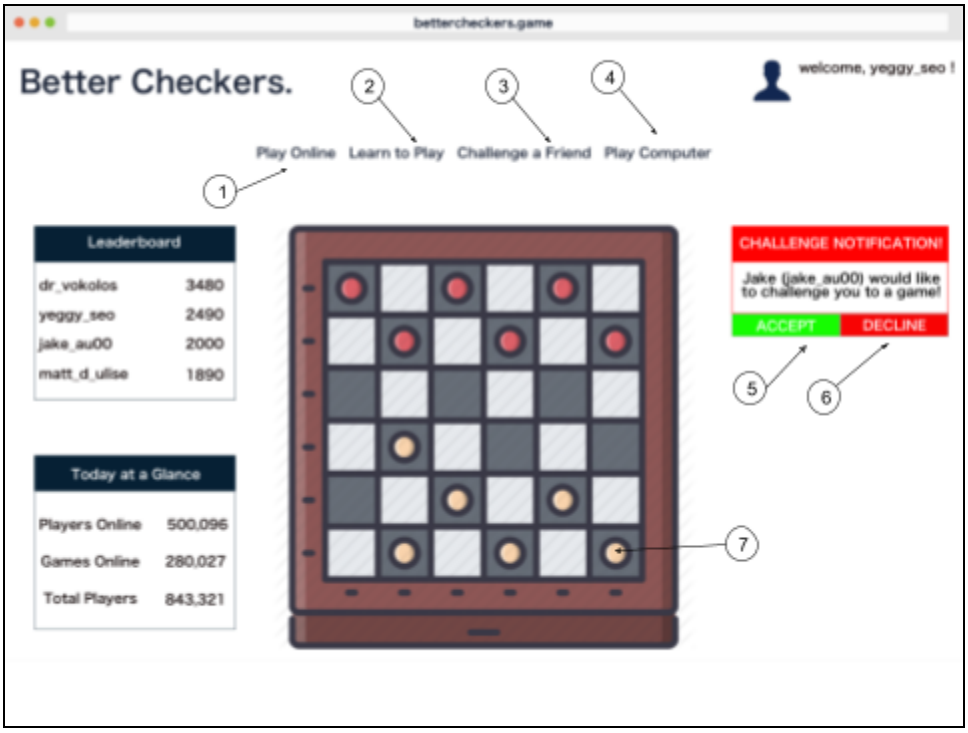

Figure 3.2 Landing Page, Playing Against a Computer Set to Easy Difficulty

Option 1 displays “in a queue” mode, shown in Figure 3.2.1.

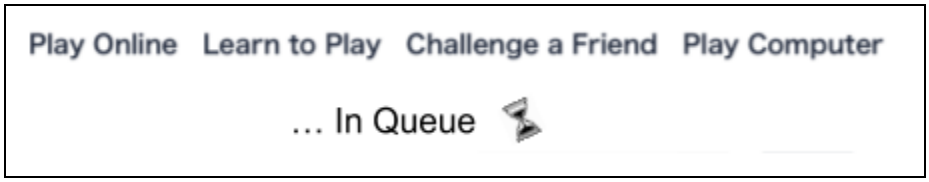

Figure 3.2.1 In Queue

Option 2 opens a pop-up of the rules and the board is dimmed out shown in Figure 3.2.2. Clicking the “X” on the top right of the pop-up will return to the original page.

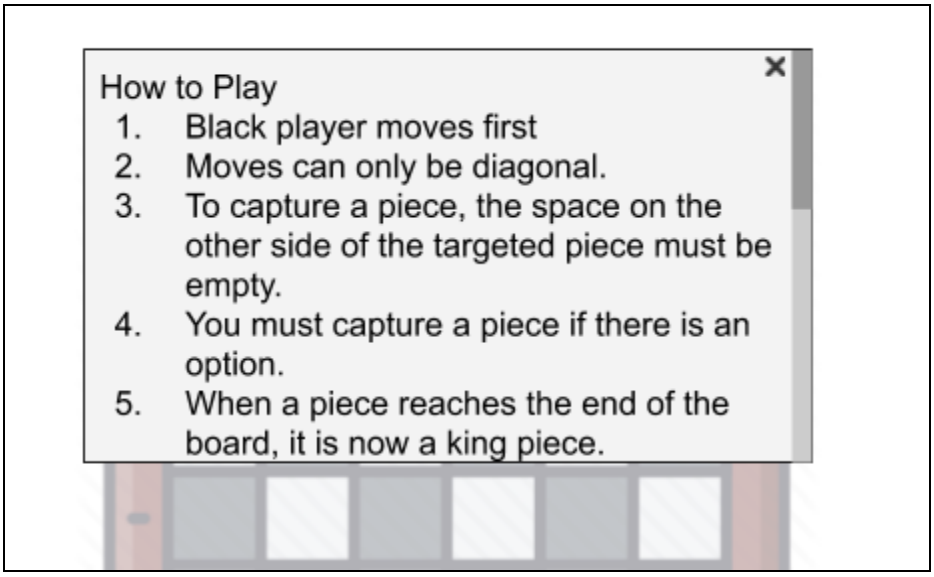

Figure 3.2.2 How to Play Pop-Up

Option 3 displays an input for an ID and the User's ID, shown in Figure 3.2.3. Selecting "GO" goes a game with the validated friend's ID.

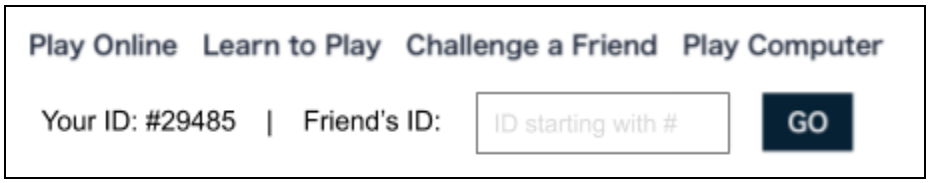

Figure 3.2.3 Challenging a Friend

Option 4 displays an input for selecting a difficulty, shown in Figure 3.2.4. Initially set to "Easy". Selecting "GO" goes to a new game against a computer with the selected difficulty.

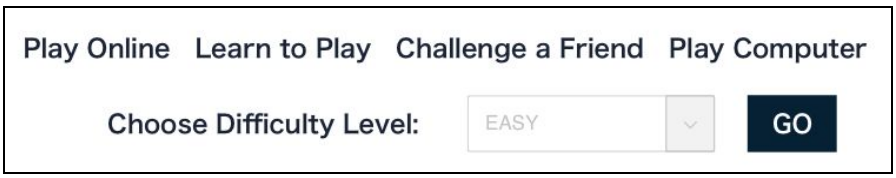

Figure 3.2.4 Selecting a Computer Difficulty

Option 5 goes to a new game against the challenger and the user leaves the queue mode from Figure 3.2.1.

Option 6 declines the challenge and the user leaves the queue mode from Figure 3.2.1.

Option 7 represents each piece in a game. More information in Figure 3.3 and Figure 3.4.

3.1.1.3 Gameplay

Figure 3.3 Gameplay Player 1 Perspective

Figure 3.4 Gameplay Player 2 Perspective

Option 1 is an example of a selected piece. Green boxes shall indicate valid moves and yellow boxes indicate the selected piece shown in Figure 3.3. It cannot be selected when the board is grey, shown in Figure 3.4.

Option 2 is an input box for text and selecting “Send” will update the chatbox shown in both Figure 3.3 and Figure 3.4.

Option 3 validates and confirms the selected move. If an invalid move, a message will appear shown in Figure 3.3.1.

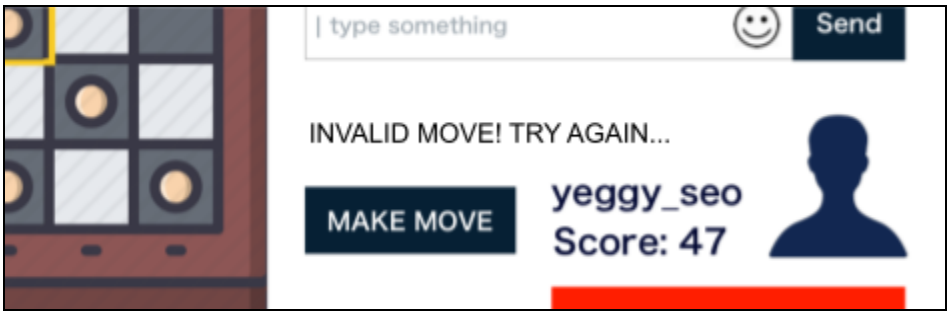

Figure 3.3.1 Invalid Move

Option 4 goes to the End Game Page shown in Figure 3.5 with the results for both players. The other player is notified that their opponent has forfeited.

3.1.1.4 End Game

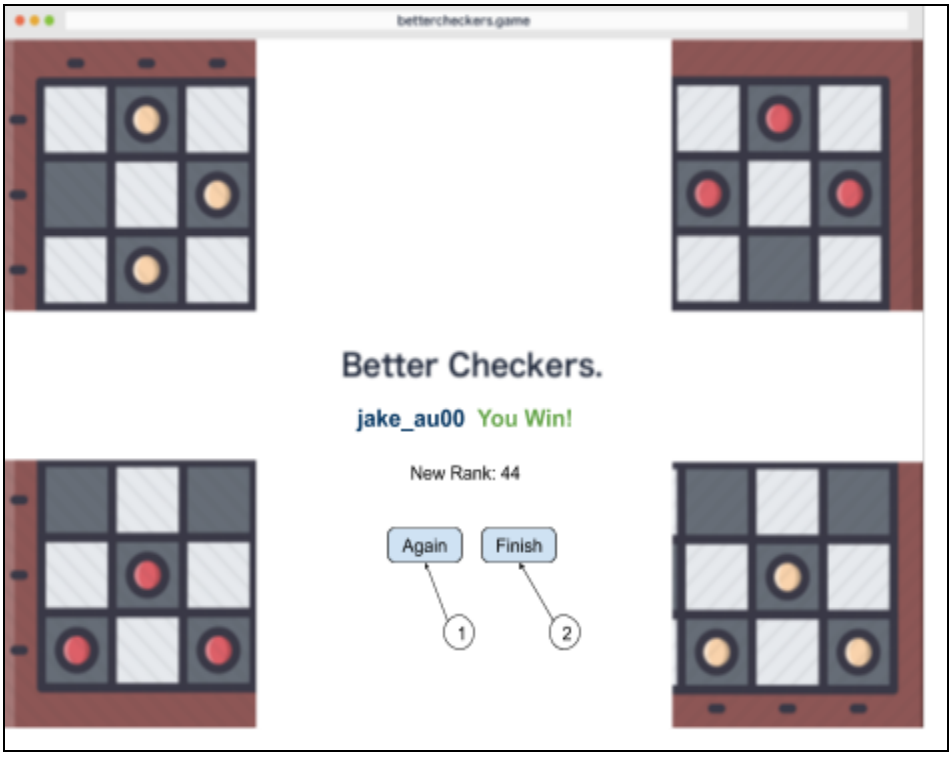

Figure 3.5 Ended Game

Option 1 goes to the landing page starting a game against an easy computer and into queue mode shown in Figure 3.2.1.

Option 2 goes to landing page starting a game against an easy computer.

3.1.2 Hardware Interfaces

The system shall allow users to play the game using a keyboard, a touch screen, or a mouse.

In regards to using a keyboard to play the game, players will be able to use the arrow keys to choose between squares, another key to select pieces and their desired moves, and another key to put pieces down.

In regards to using the mouse to move to play the game, the system will support both a “drag and drop” and “click to move” functionality when moving pieces. With “click to move”, a user can click on the piece then the desired location. With “drag and drop” the user can drag the piece to where they want to go.

The touch screen should offer the same options as the mouse.

3.2 Functional Requirements

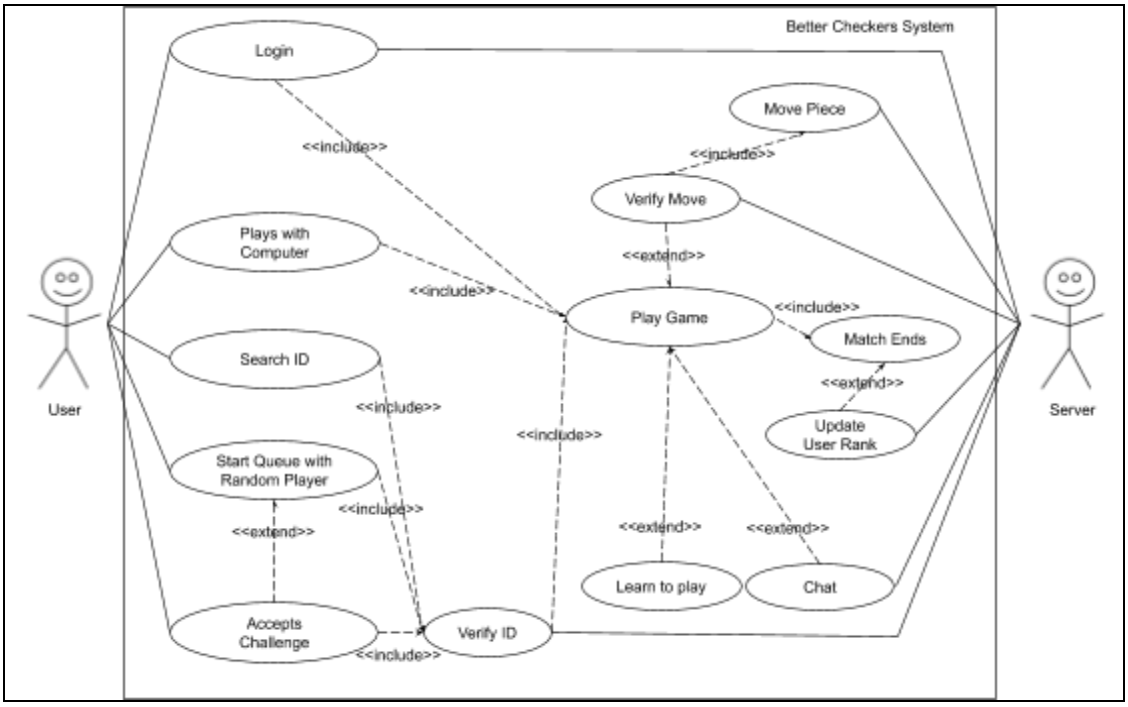

Figure 3.6 Function Diagram

The use-case diagram in Figure 3.6 shows the relationship between User and Server when each case is executed. The use-cases set the framework of the requirements.

Error Cases at any given time:

Error Case 1: User Exits Browser
The abrupt ending of the session by User closing their browser at any point causes a break out of the normal flow and delete the Unique ID as well as a username if it was provided after 5 minutes.

Error Case 2: Server is On
All these features require that the server is online and working. If the server is offline, display a “Server Not Online” page.

3.2.1 System Feature Login :: REQ1

3.2.1.1 Description and Pre-Conditions

The User wants to log in.
Pre-conditions: User is on the login page.

3.2.1.2 Normal Flow

1. System asks User for a username that cannot exceed 10 characters.
2. User inputs their desired username.
3. System checks input.
4. Server creates a unique ID and stores it along with the username in the database.
5. System notifies User that their username was created successfully.
6. System displays a message to User: “This username is only temporary. Once you exit your browser, the username and your progress will be deleted.”
7. System starts REQ7 Play Game with the match set to vs Computer on Easy mode.

3.2.1.3 Post-Conditions

The username and ID are stored in the database. User is notified that their username was created successfully. User is then assigned a game versus a computer with the game difficulty set to Easy.

3.2.1.4 Error Cases

Error Case 1: Invalid Username
If in case the username entered in Step 2 above turns out to be invalid, System shall display the message: “Username Invalid: Please Try Again” after which the normal flow is carried out from Step 1 again.

3.2.2 System Feature Play with Computer :: REQ2

3.2.2.1 Description and Pre-Conditions

User wants to play against a computer.
Pre-conditions: User has a valid ID.

3.2.2.2 Normal Flow

1. User selects the option “Play with Computer”.
2. System asks User to pick a level of difficulty with the options being Easy, Medium and Hard.
3. User selects the desired difficulty.
4. System starts Use REQ7 Play Game with the match set to vs Computer at the selected difficulty level.

3.2.2.3 Post-Conditions

User is assigned a game with the level of difficulty that was selected by them.

3.2.3 System Feature Search ID :: REQ3

3.2.3.1 Description and Pre-Conditions

User wants to play against a friend.
Pre-Conditions: User has a valid ID.

3.2.3.2 Normal Flow

1. User selects Play with Friend.
2. System displays User's ID.
3. System asks User for their friend's ID.
4. User inputs a friend's ID.
5. System starts REQ6 Verify ID with the match to: vs Friend.

3.2.3.3 Post-Conditions

User provides System with their friend's ID.

3.2.4 System Feature Start Queue with Random Player :: REQ4

3.2.4.1 Description and Pre-Conditions

User wants to play against a random player.
Pre-Conditions: User has a valid ID.

3.2.4.2 Normal Flow

1. User selects Play with Random Player.
2. System adds User's ID to a queue.
3. System searches for the proper ID, based on the time it spent in the queue and the similarity of the ranks between IDs.
4. System starts REQ6 Verify ID with the match set to vs Ranked.

3.2.4.3 Post-Conditions

User's ID is added to the queue. Searches for similar ID based on time in queue and rank. User is then assigned a game vs the random ID.

3.2.4.4 Error Cases

Error Case 1: Opponent Not Found
If in the case a match is not found within 30 minutes in Step 3, System shall remove User's ID from the queue and exits the use case.

3.2.5 System Feature Accepting Challenges :: REQ5

3.2.5.1 Description and Pre-Conditions

User wants to accept a challenge from another user.
Pre-Conditions: User has a valid ID. User is in a game vs the Computer. The match is set to vs Friend.

3.2.5.2 Normal Flow

1. System notifies User that they have been challenged by a friend.
2. User selects accepts the challenge.
3. System starts REQ6 Verify ID with the challenger's ID.

3.2.5.3 Post-Conditions

User is assigned a game vs the challenger's ID.

3.2.5.4 Alternate Flow

Alternate Flow 1: vs Ranked
If the match is set to vs Ranked, in Step 1 System shall notify User that they have been challenged by a random player. Continue to Step 2 and proceed as normal flow.

3.2.6 System Feature Verify ID :: REQ6

3.2.6.1 Description and Pre-Conditions

System wants to verify an ID.
Pre-Conditions: User has typed in an ID in an attempt to challenge another User, or User has accepted a challenge from another User.

3.2.6.2 Normal Flow

1. System sends the given ID to Server.
2. Server finds the given ID in the database and returns ID to System.

3. System starts REQ7 Play Game with the given ID and User ID.

3.2.6.3 Post-Conditions

System has a verified ID. User is assigned a game vs the other User's validated ID.

3.2.6.4 Error Cases

Error Case 1: Invalid ID
If in the case in Step 2 Server fails to find the given ID in the database, System shall display the message to User: "Invalid ID, player not found".

3.2.7 System Feature Play Game :: REQ7

3.2.7.1 Description and Pre-Conditions

System loads a new game for User.
Pre-Conditions: System has set the match to vs Friend, vs Ranked, or vs Computer at a certain difficulty level. Successfully completed REQ6 Verify ID, or REQ2 Play with Computer.

3.2.7.2 Normal Flow

1. System displays game board, chatbox, and timer countdown to User.
2. System outputs in the chatbox: "$Username1 vs $Username2 in a ranked match".
3. System randomly chooses which player goes first.
4. System sets Player 1 to User that goes first and Player 2 to the other User.
5. System indicates it is Player 1's turn and not Player 2's turn.
6. System starts Player 1's timer countdown.
7. System asks Player 1 to make a move.
8. Player 1 selects a piece.
9. Player 1 chooses a move.
10. System starts REQ8 Verify Move with Player 1's chosen move.
11. System repeats Steps 5-10 with the other player until pre-conditions for REQ12 Match Ends are met.

3.2.7.3 Post-Conditions

Both Users end a match and the winner and loser are decided. System knows the winner and loser's IDs.

3.2.7.4 Alternative Flows

Alternate Flow 1: vs Computer
If the match is set to vs Computer, Step 2 is replaced with: System gives User the option of going first, computer going first, or making who goes first random. Continue to Step 3 and proceed as normal flow.

Alternate Flow 2: vs Ranked or vs Friend
If User is playing against computer and match is set to vs Ranked or vs Friend then start REQ6 Verify ID with the given ID and match set to vs Ranked or vs Friend, respectively.

3.2.7.5 Error Cases

Error Case 1: User Exits Browser While in a Game vs Ranked
If match is set to vs Ranked and a User exits the browser. User has 5 minutes to reconnect. If they do not reconnect, Server deletes User's username and ID from the database and the other player is the winner. The opponent then receives a notification that the other player has disconnected.

3.2.8 System Feature Verify Move :: REQ8

3.2.8.1 Description and Pre-Conditions

System wants to verify a move taken by User.
Pre-Conditions: User selects a piece and makes move.

3.2.8.2 Normal Flow

1. Using the current state of the board and given piece, Server compares a list of possible moves with the given move.
2. Server verifies the given move.
3. Server returns 'change' and/or 'capture', depending on whether the move is changing their piece to a king piece or capturing opponent's pieces.
4. System starts REQ9 Move Piece with what was returned by Server in Step 3.

3.2.8.3 Post-Conditions

Move is determined to be valid or not. If it is, the appropriate action takes place. If it isn't, User must make another move.

3.2.8.4 Alternate Flows

Alternate Flow 1: Invalid Move

If, after Step 2, Server finds the given move invalid, System indicates to User that the given move is invalid. System then reverts back to REQ7 Play Game Step 7.

3.2.9 System Feature Move Piece :: REQ9

3.2.9.1 Description and Pre-Conditions

User wants to move a selected piece.
Pre-Conditions: Move action is set to ‘move’.

3.2.9.2 Normal Flow

1. System displays to User their move.
2. Server updates the current state of the board to the board after the move is evaluated.

3.2.9.3 Post-Conditions

User’s move has been executed. The board state has been changed in the database.

3.2.9.4 Alternative Flow

Alternate Flow 1: King Piece
If ‘change’ was returned, include in Step 1 the animation for replacing the given piece with a king piece. Continue to Step 1 and proceed as normal flow.

Alternate Flow 2: Capture
If ‘capture’ was returned, include in Step 1 the animation removing the pieces that were captured and User’s piece jumping over them. Continue to Step 1 and proceed as normal flow.

3.2.9.5 Error Cases

Error Case 1: King Piece to King Piece
If the given piece is already a king piece and move action is set to ‘change’, do nothing and continue.

3.2.10 System Feature Learning to Play :: REQ10

3.2.10.1 Description and Pre-Conditions

User wants to learn the rules and how to play checkers.
Pre-Conditions: User has a valid ID.

3.2.10.2 Normal Flow

1. User selects learn to play.
2. System opens a pop-up of a Learn to Play guide and User is unable to select anything else but to exit the pop-up.
3. User exits pop-up by clicking on the “X” or select around the pop-up.
4. System removes pop-up and gameplay is continued as normal as before.

3.2.10.3 Post-Conditions

User learned to play.

3.2.11 System Feature Chatting :: REQ11

3.2.11.1 Description and Pre-Conditions

User wants to send messages to the other User.
Pre-Conditions: Match is set to vs ranked or vs friend.

3.2.11.2 Normal Flow

1. System gives User the option to select a variety of preset phrases and emojis.
2. User selects a pre-set sentence or emoji.
3. Server updates User and opponent’s chatbox with the most recent at the bottom.
4. System displays User’s message with a timestamp.

3.2.11.3 Post-Conditions

User has sent a message. Chatbox for both Users is updated.

3.2.12 System Feature Match Ends :: REQ12

3.2.12.1 Description and Pre-Conditions

User’s match has come to an end.
Pre-Conditions: User concedes, the timer runs out, or User runs out of pieces.
Successfully completed REQ7 Play Game.

3.2.12.2 Normal Flow

1. System displays to User and opponent the match outcome, the loser gets a “you lose” message and the winner gets a “you win” message.

2. System starts REQ13 Update User Rank with winner and loser's ID.
3. System gives User the option to play again.
4. User selects play again.
5. System starts REQ4 Start Queue with Random Player.

3.2.12.3 Post-Conditions

System notifies User that the match is over and updates the rank of the winners and losers of the match. User is asked if they want to play again.

3.2.12.4 Alternative Flows

Alternative Flow 1: vs Friend
If the match is set to vs friend, Step 2 will not occur and in Step 5 System shall start REQ7 Play Game with winner and loser's ID. The rest proceeds as normal.

Alternate Flow 2: vs Computer
If the match is set to vs computer, System in Step 3 shall give User the option to select difficulty again and start REQ7 Play Game. The rest proceeds as normal.

Alternate Flow 3: Finish
If User selects not to play again, start REQ7 Play Game with a computer on easy difficulty.

3.2.13 System Feature Update User Rank :: REQ13

3.2.13.1 Description and Pre-Conditions

System wants to update winners and losers rank in the database.
Pre-Conditions: Successfully completed Step 2 of REQ12 Match Ends. The match is set to vs Ranked. The ID of the Winner and Loser are given.

3.2.13.2 Normal Flow

1. Server updates the Winner ID's rank status.
2. Server updates the Loser ID's rank status.
3. System displays their new ranked status to each User.

3.2.13.3 Post-Conditions

Loser and winner ID's rank status is updated and displayed.

3.3 Sequence Diagrams

For each requirement, a sequence diagram follows. Below are the corresponding diagrams that explain in detail the relationships between the actors.

3.3.1 REQ1

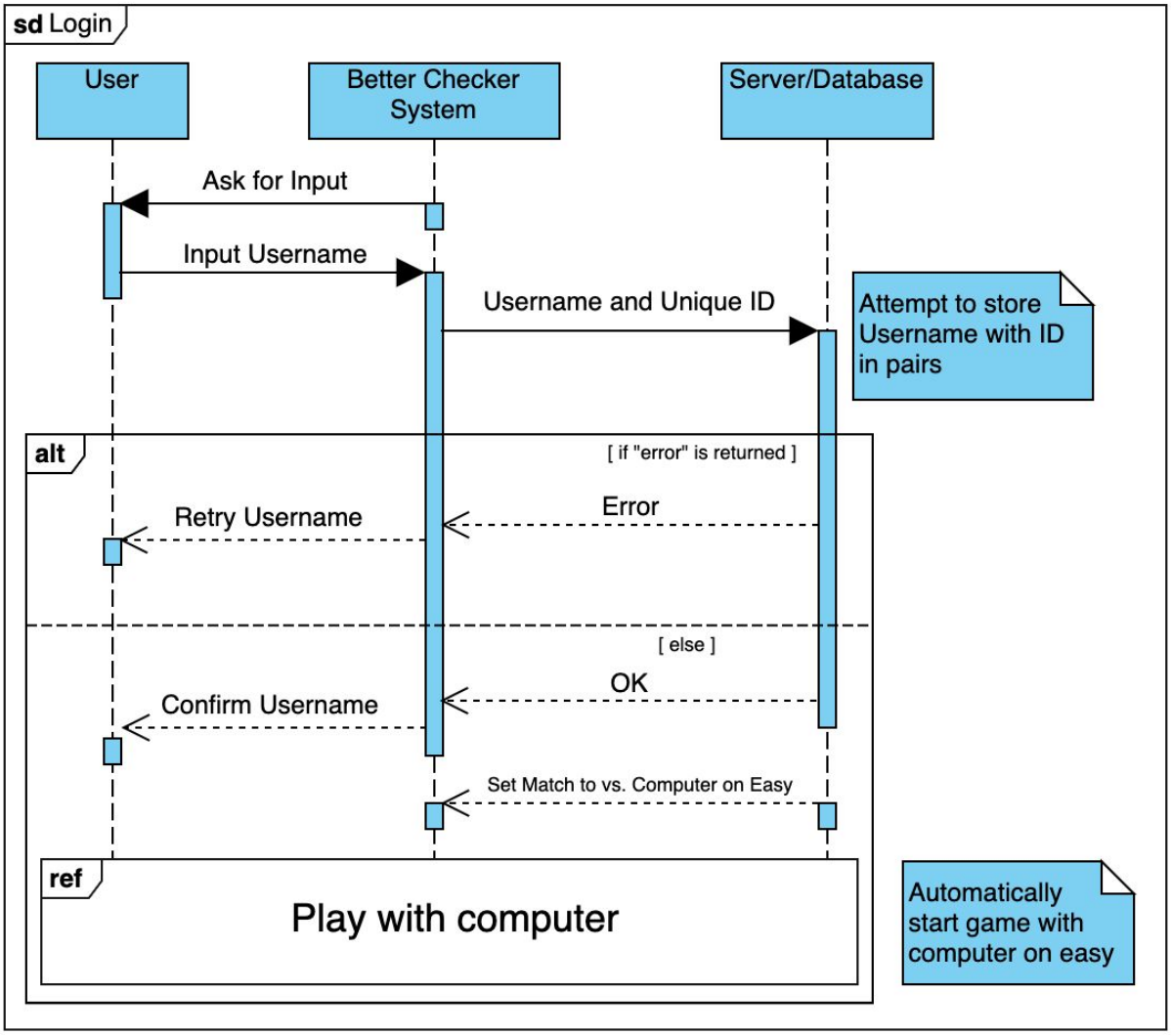

Figure 3.7 Login

3.3.2 REQ2

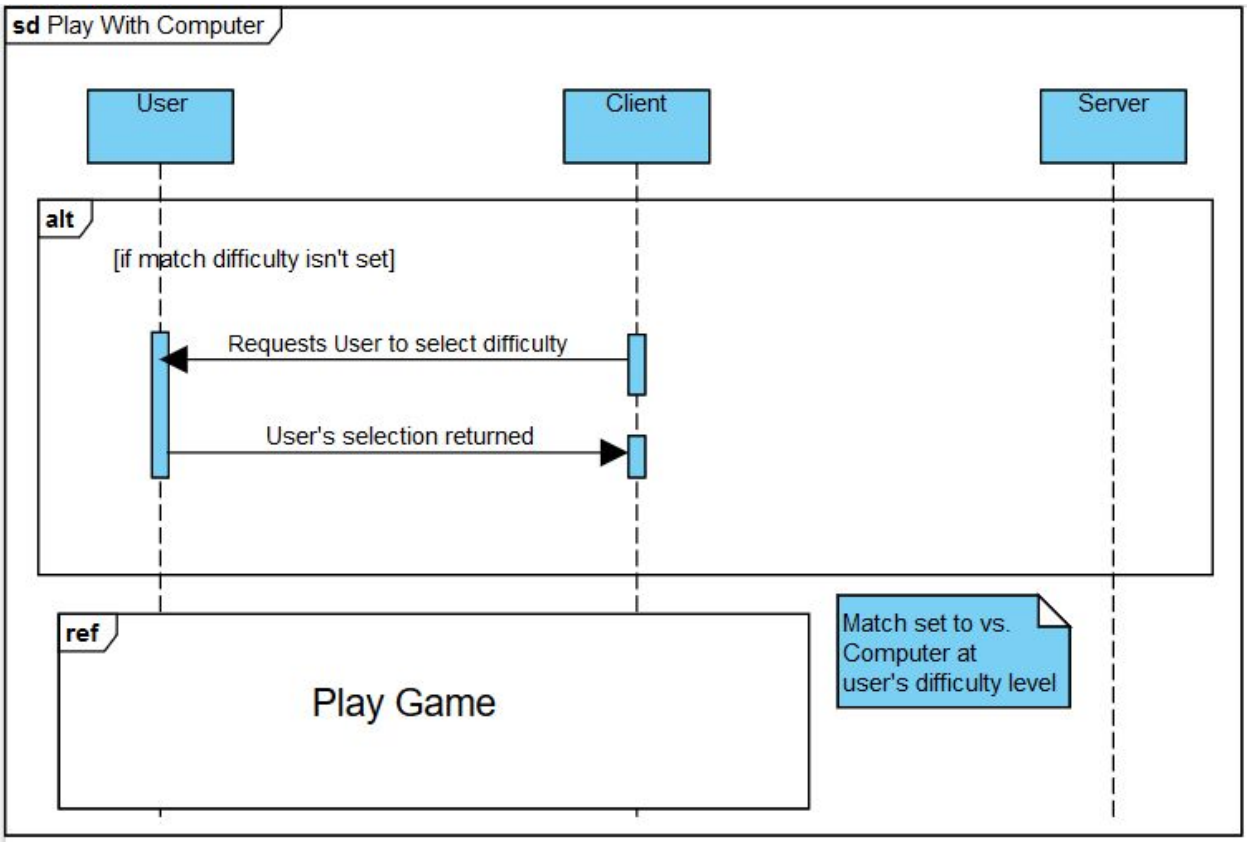

Figure 3.8 Play With Computer

3.3.3 REQ3

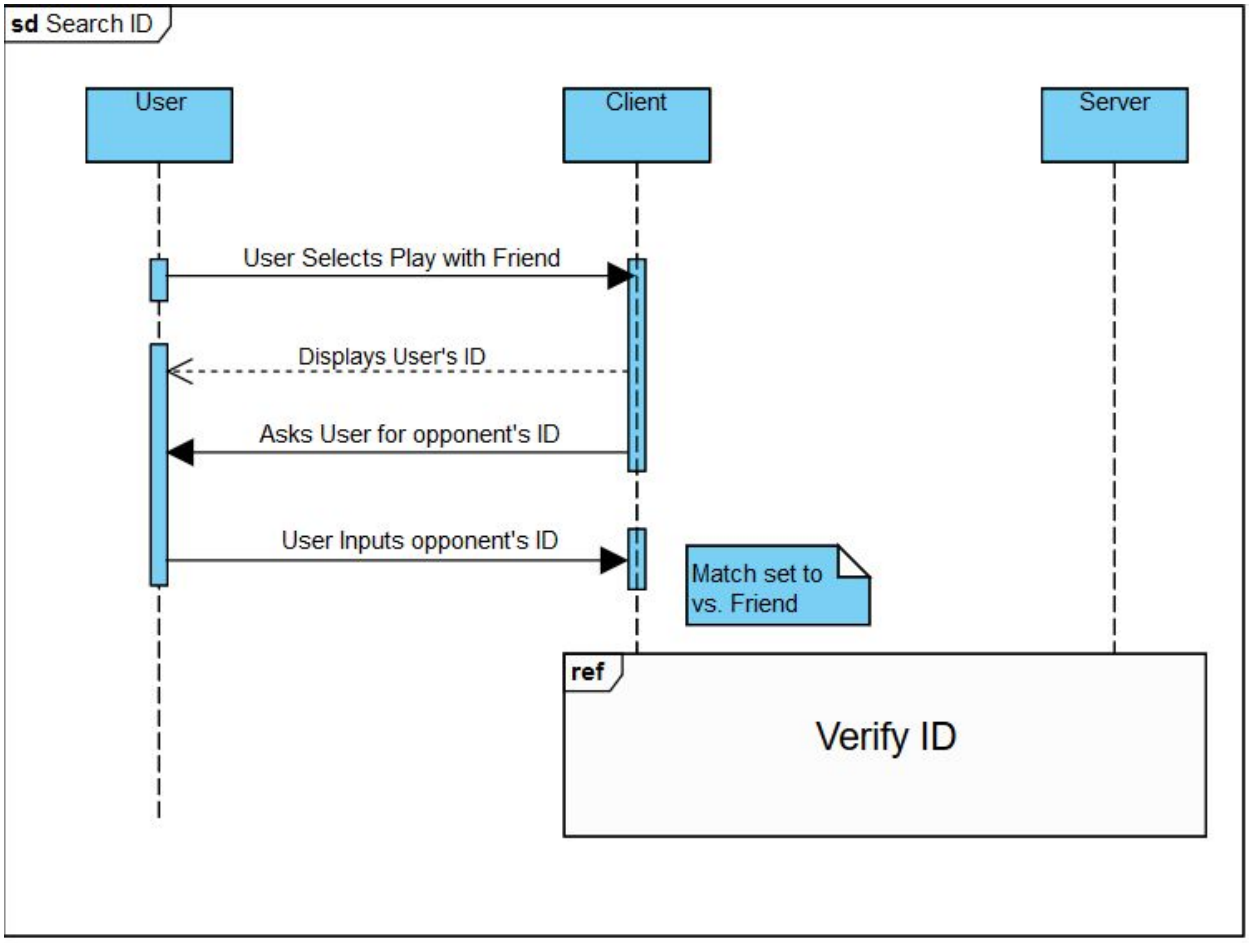

Figure 3.9 Search ID

3.3.4 REQ4

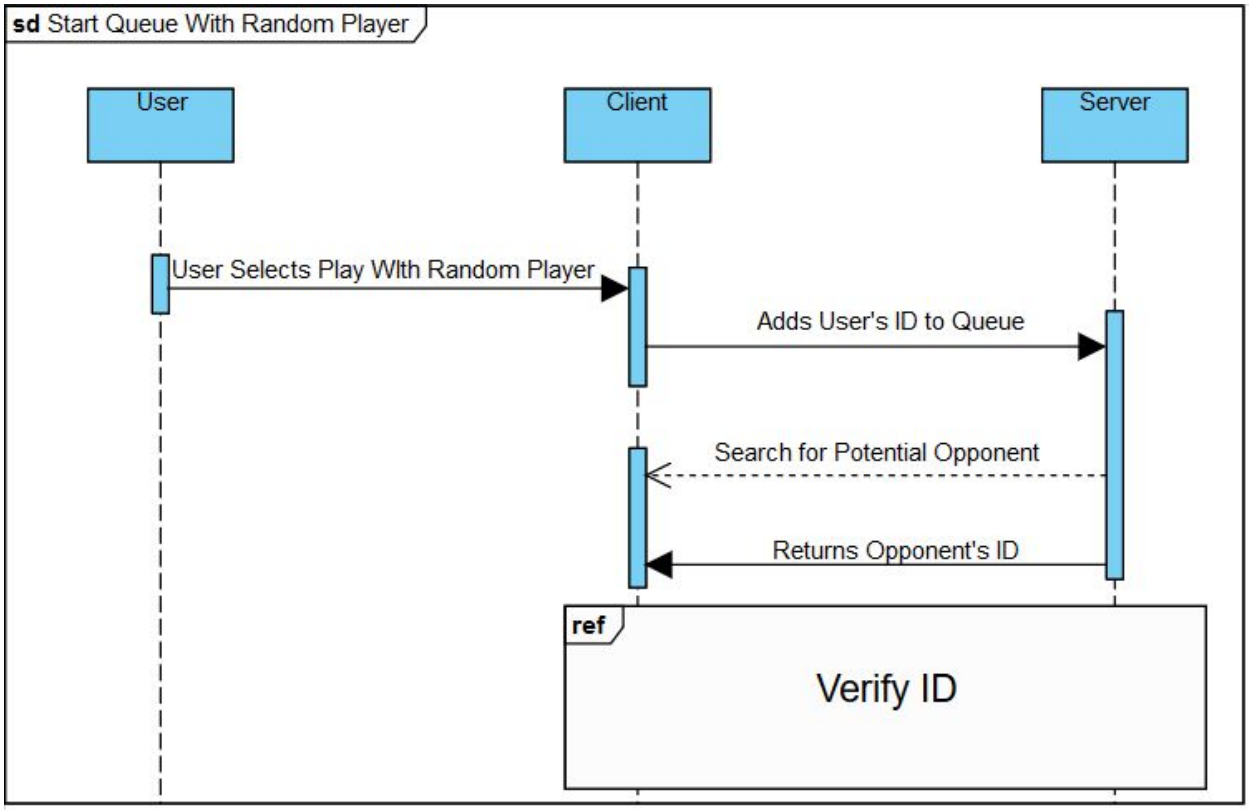

Figure 3.10 Start Queue with Random Player

3.3.5 REQ5

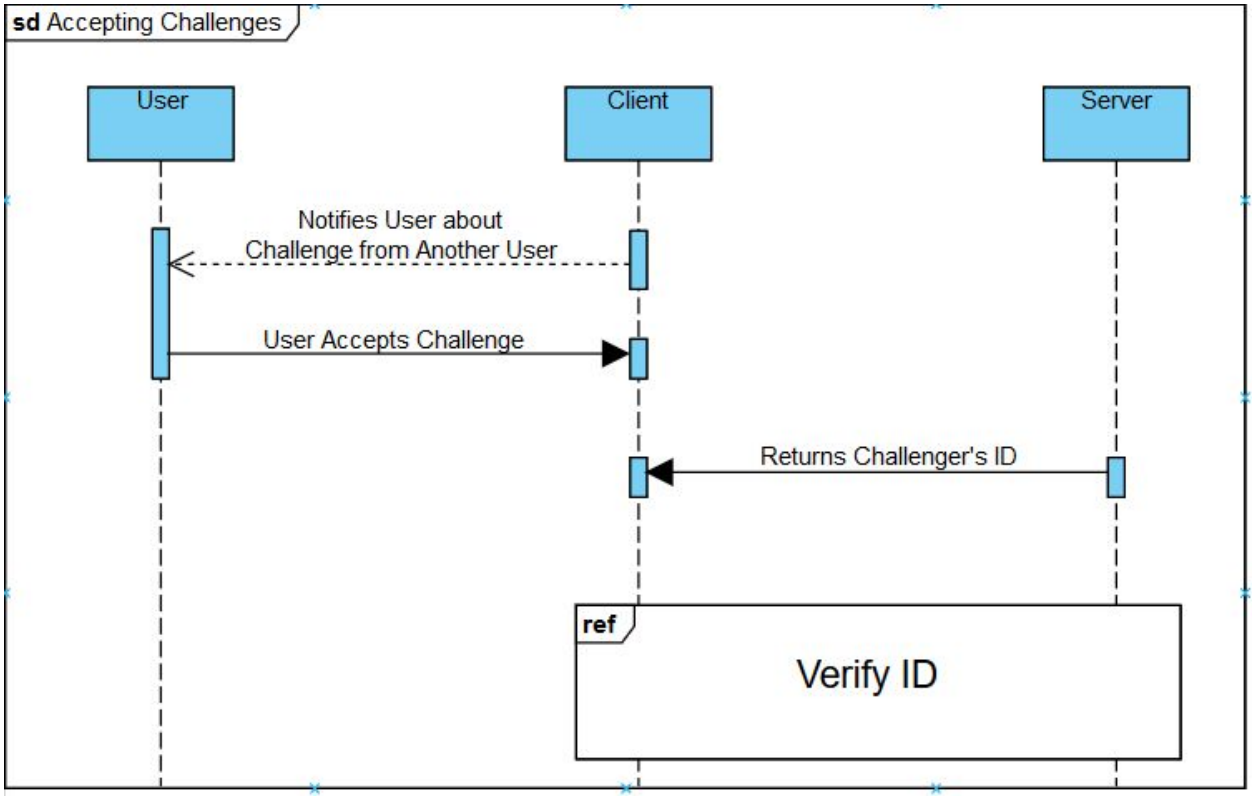

Figure 3.11 Accepting Challenges

3.3.6 REQ6

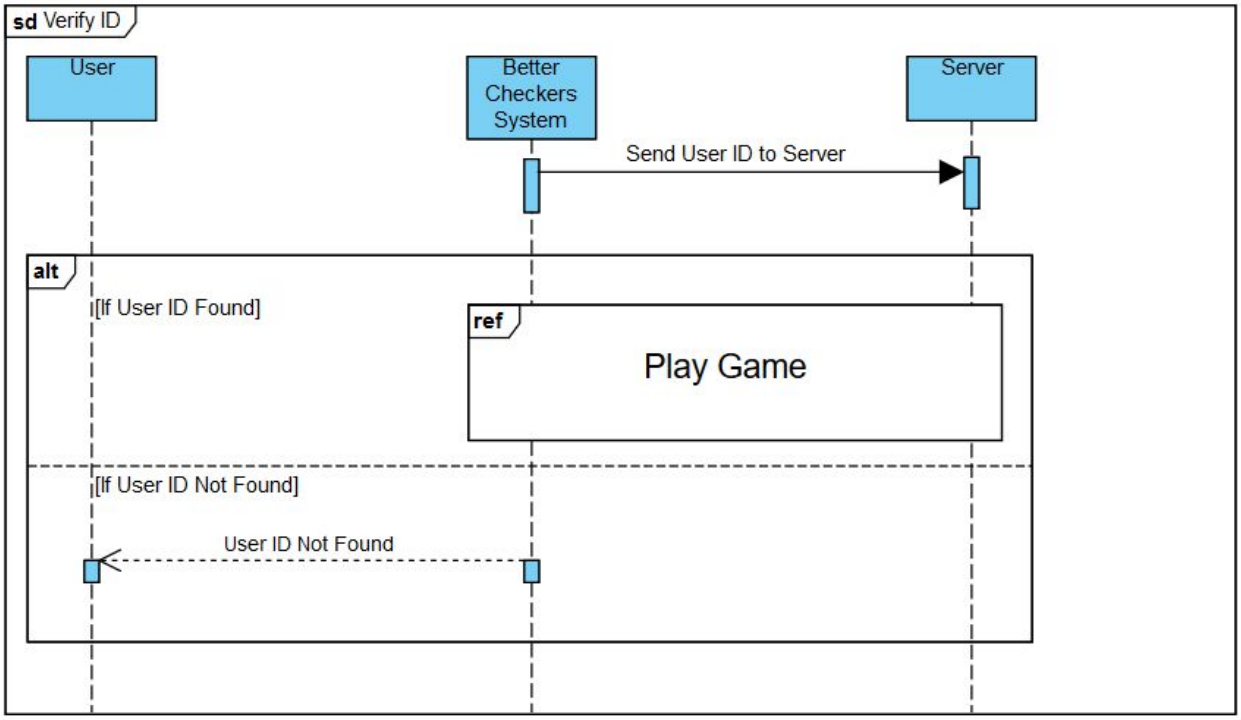

Figure 3.12 Verify ID

3.3.7 REQ7

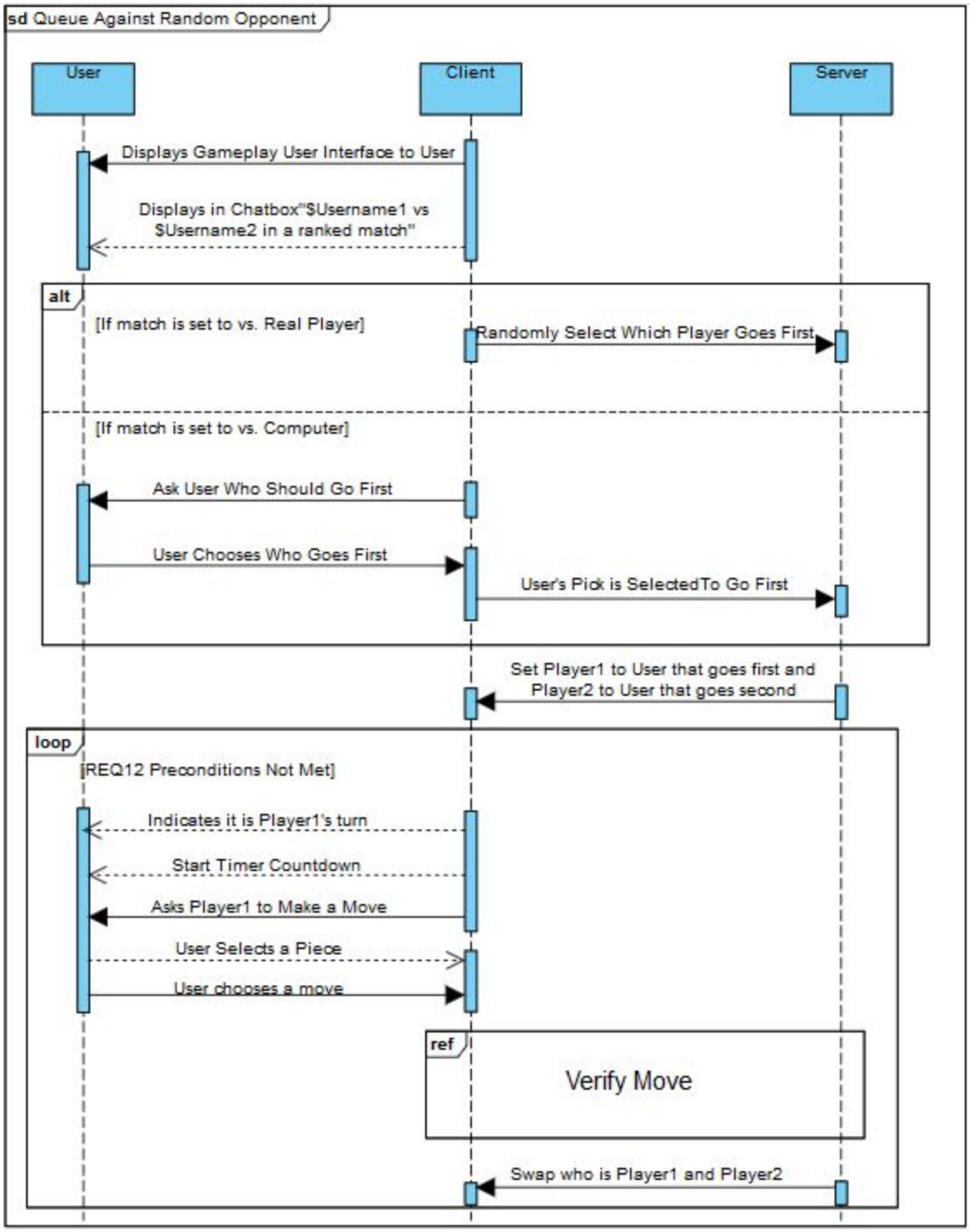

Figure 3.13 Queue Against Random Opponent

3.3.8 REQ8

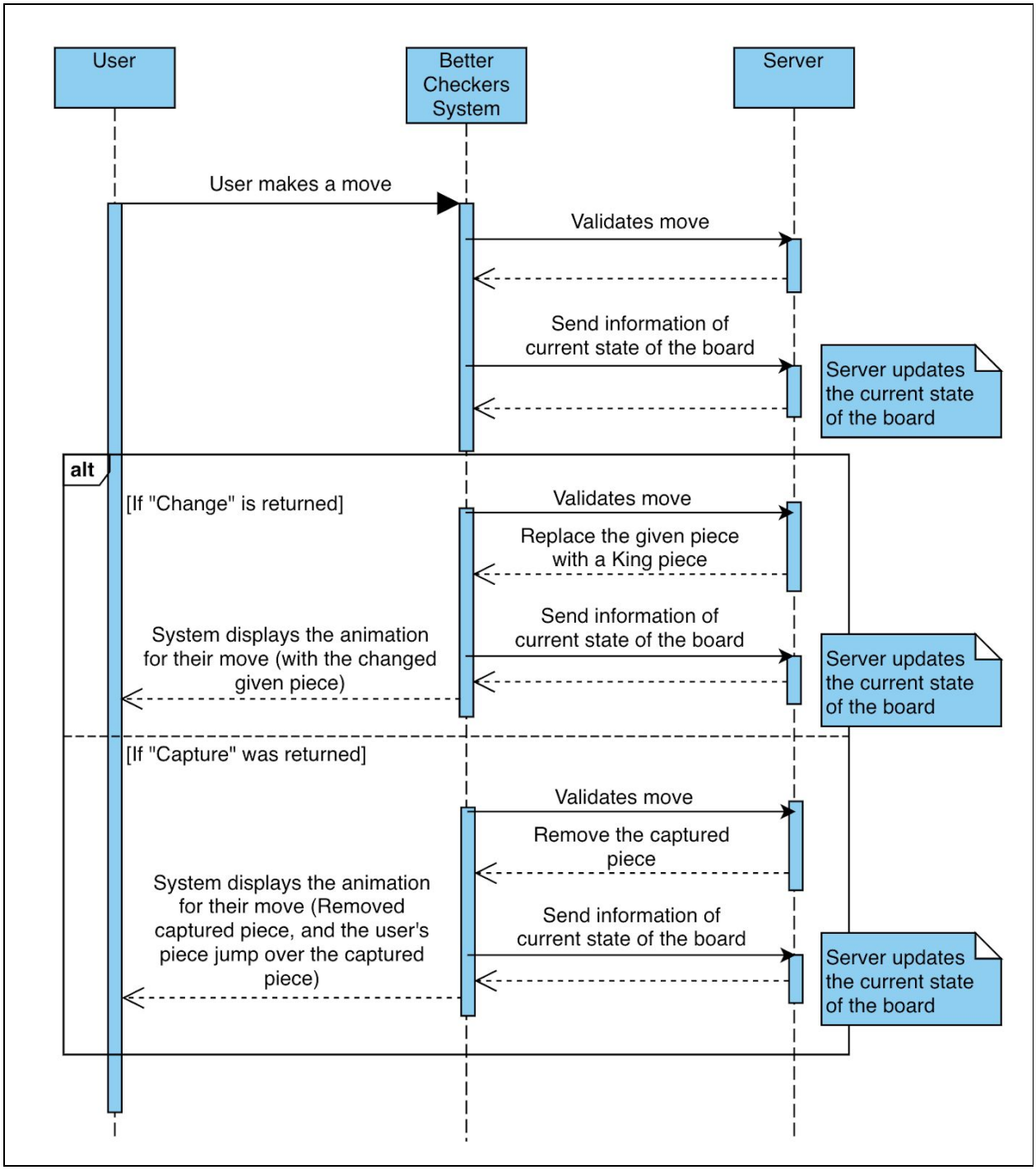

Figure 3.14 Verify Move

3.3.9 REQ9

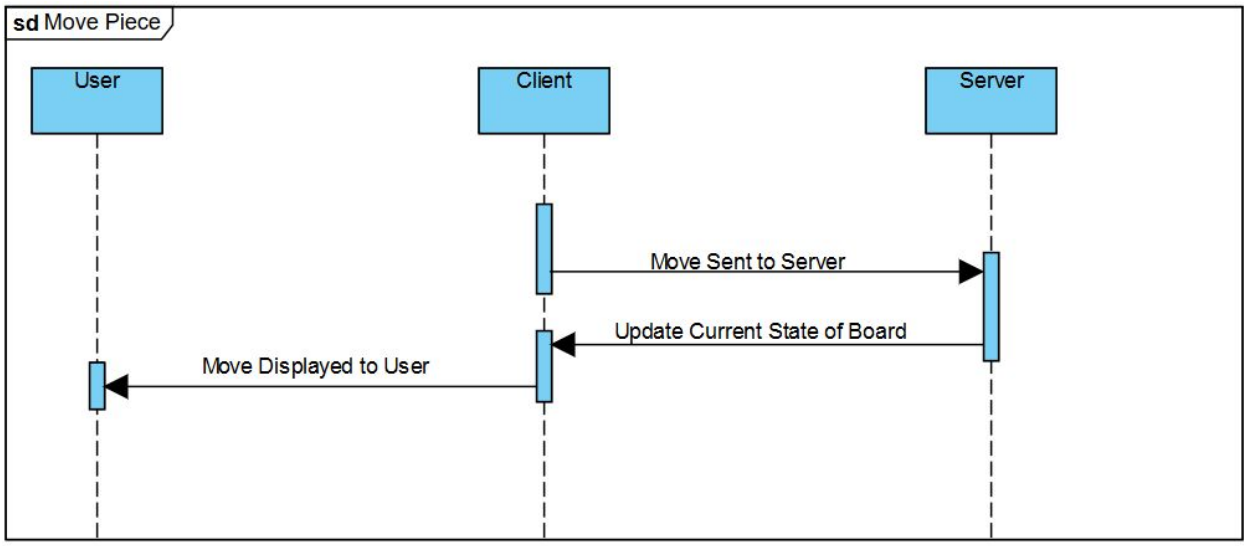

Figure 3.15 Move Piece

3.3.10 REQ10

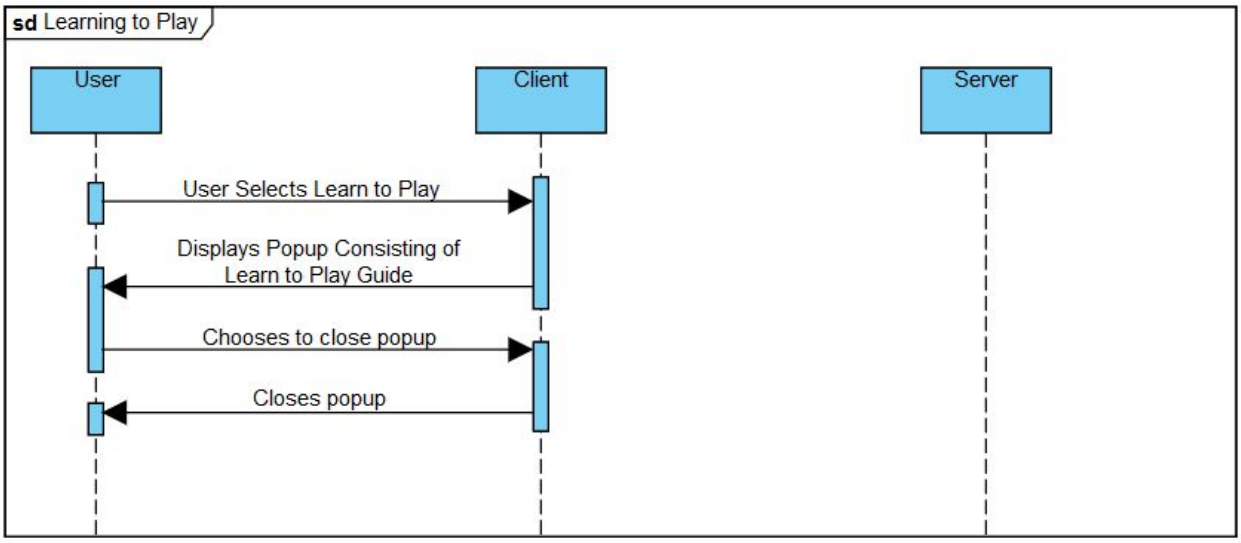

Figure 3.16 Learning to Play

3.3.11 REQ11

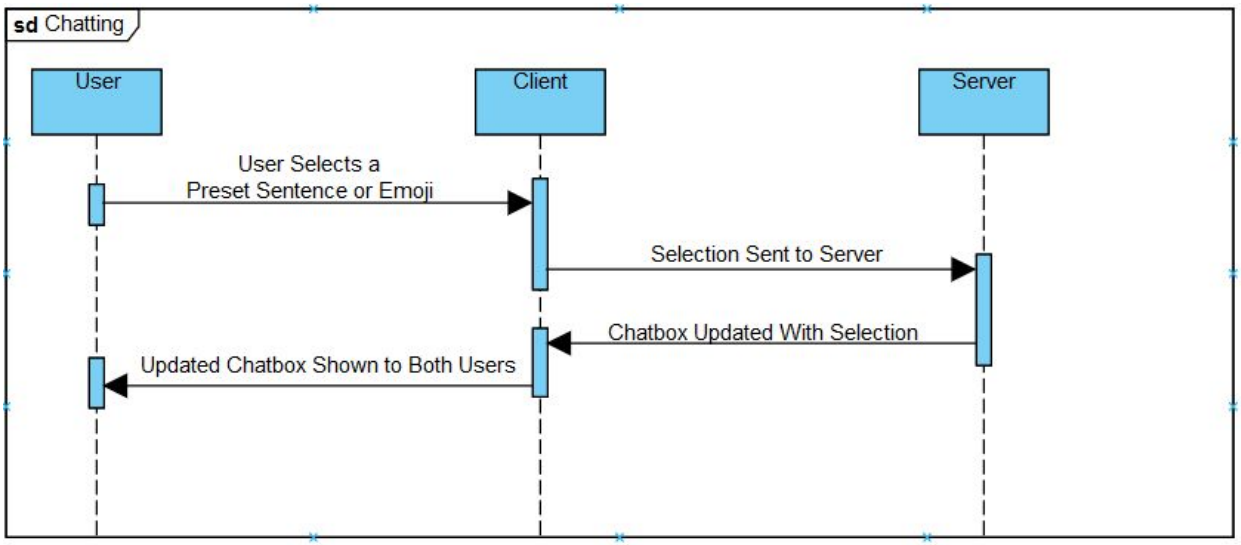

Figure 3.17 Chatting

3.3.12 REQ12

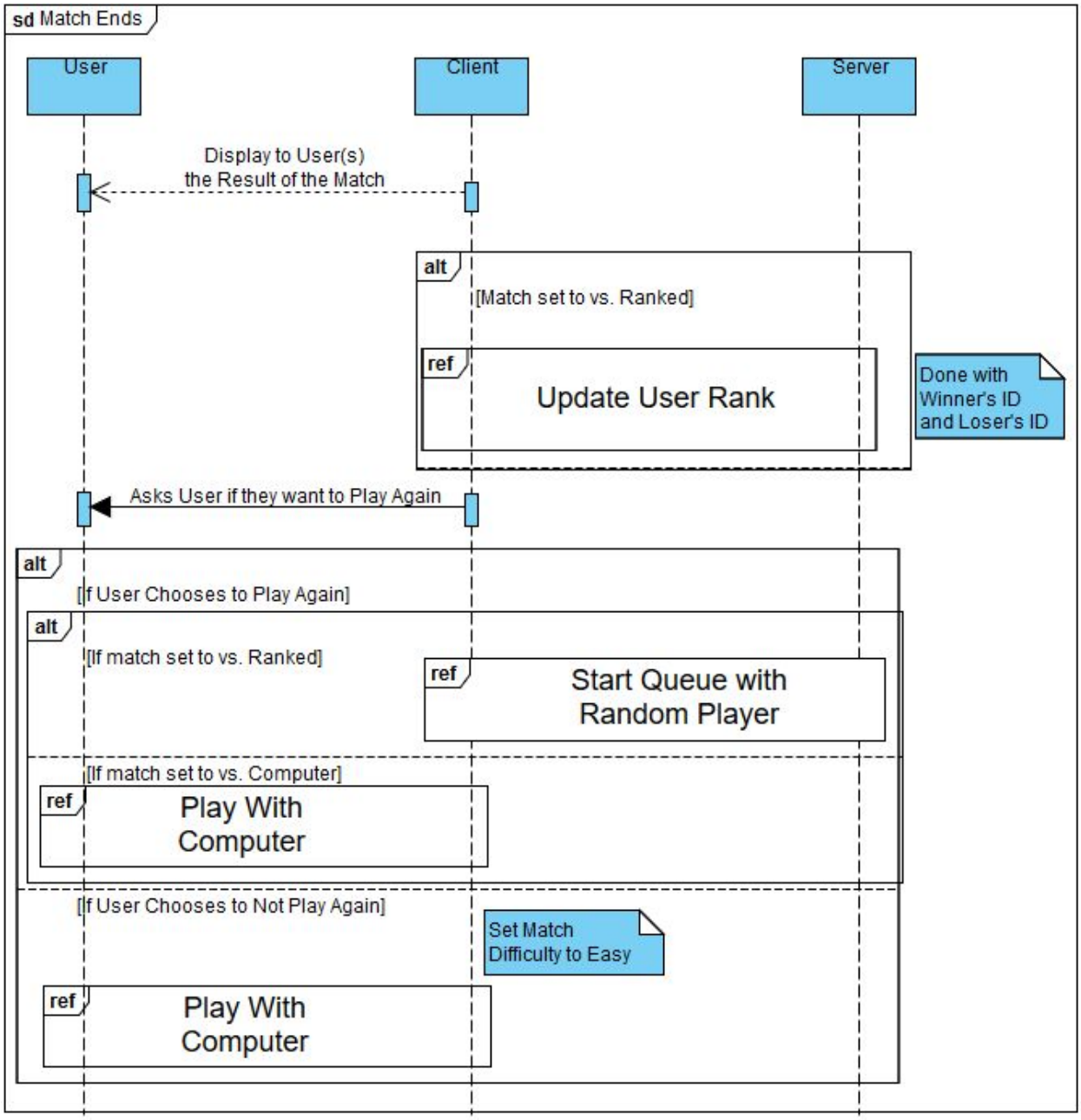

Figure 3.18 Match Ends

3.3.13 REQ13

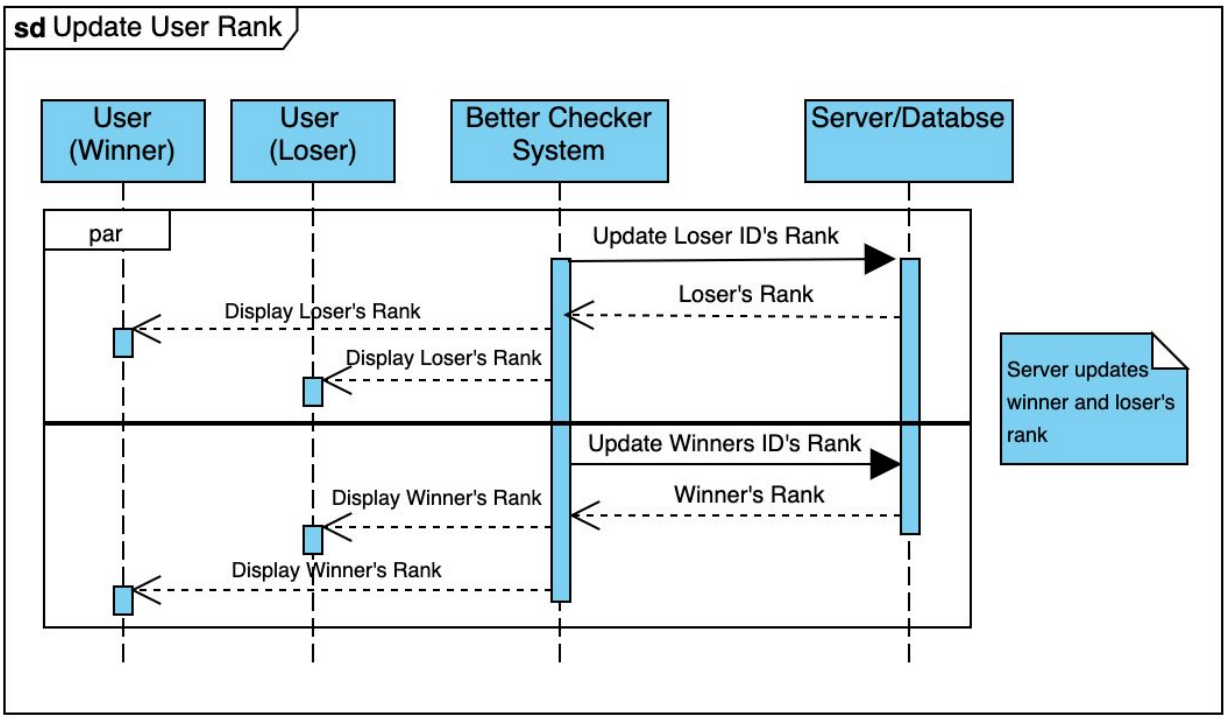

Figure 3.20 Update User Rank

3.4 Nonfunctional Requirements

3.4.1 Performance

The Better Checkers application will be capable of providing sub-two second response times for approximately 90% of the live users.

3.4.2 Scalability

The Better Checkers application is designed in a way such as to handle a maximum of 50,000 users at any given moment that the application is live. Future improvements regarding the expansion of this threshold will be updated in the next functional specification for the product.

3.4.3 Availability

The application must run for 363 days out of the 365 days of a year, with an availability of 0.994. The 2 days that the system is projected to be down will be used for the purposes of maintenance and updates.

3.4.4 Reliability

The Better Checkers application mandates that there should not be any message loss in order to not compromise the performance of the application. This, in turn, implies that the threshold of response times with regards to the outcomes of message and packet deliveries is restricted to 30 seconds.

3.4.5 Usability

Considering the fact that Better Checkers is designed to be a responsive web application, it is designed to run on an Internet browser supporting JavaScript. It is also designed to be cross-functional and run on mobile phones, laptops, tablet computers, and desktop computers.

3.4.6 Security

The privacy of users, as well as the integrity of the database, is paramount to the working of Better Checkers. This, in turn, implies that all communications made will undergo a stringent authentication process. In addition, these communications will be encrypted using the appropriate certificates.

It should be noted that since the Better Checkers application allows users to chat with each other or make profiles, questions might arise regarding the harvesting of user data or the vulnerability of the application towards cyber-attacks. Privacy is paramount and therefore, none of the chat data between users will be collected. Usernames will be encrypted using a unique key for each user.