Software Requirements Specification

for

Checkers

Version 1.0 approved

Prepared by Lale Saparova and Dacota Carpenter

WIU-CS 491 Soft.Eng.1

09.21.2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirements Specification document is intended to demonstrate the (partial or whole) utter outline of the computer-based game project Checkers, including game mechanics, user interface, and user experience. The SRS list includes all the characteristics Checkers have and those that are crucial to their application.

1.2 Document Conventions

There is a disparity between player_1 and player_2. Player 2 can be the character that is being controlled by another person or a CPU player. Whereas player_1 is always an actual person engaging with the game. The SRS documentation printed in Times New Roman font with the normal text size 12 black.

1.3 Intended Audience and Reading Suggestions

The SRS paper also explains project managers and tests a mechanism to make sure that the game stays true to the original version. The text was prepared in segments and may therefore be viewed as such to fully grasp the project. The overall description for a summary of the document might be viewed in Section 2. The System features provide a thorough explanation of the gaming components and how they relate to each other and can be viewed in Section 3. The additional non-functional requirements outline the technical standards the team will adhere to the project in Section 5.

1.4 Project Scope

This system's scope is to create a checkers gaming platform using the NetBeans GUI interface. The system will support the game with multiple players including a CPU player. On opposite ends of the gaming board, two opponents play the game of checkers. Players 1 and 2 each have a set of dark and light pieces. Player’s switch turns. An opponent's pieces cannot be moved by a player. The support for the fundamental set of features (multiplayer, CPU), as well as the simplicity of developing game comprehension, are advantages of creating this game. This game is available for low-memory devices.

1.5 References

- Game Instructions: https://en.wikipedia.org/wiki/Checkers#General_rules

2. Overall Description

2.1 Product Perspective

This product is a replacement for certain existing systems. The consumer can alter the game's state by interacting with it while offline. The user's device must support Java 8 or later versions.

2.2 Product Features

This product will have a fully playable Checkers game. It will have a simple multiplayer feature that allows the user to play against another human or a CPU player. The program will also include an easily understandable user interface.

2.3 User Classes and Characteristics

We anticipate that this product will be primarily used by a younger audience with a basic level of literacy. Our product can also be used by people of older ages as well, but the simplicity may not be as appealing or necessary for that user class.

2.4 Operating Environment

Our software will be runnable on operating systems compatible with Java 8 and above.

2.5 Design and Implementation Constraints

Acquiring requirements and developing the game's software architecture make up the design phase. Throughout the implementation phase, the phase will be reviewed multiple times. The iterative process of developing the game is what makes up the implementation phase. The overall architecture should be put into practice before writing the actual code, tests, and comprehensive documentation. There will be several sprints during the implementation phase.

2.6 User Documentation

The user can learn how to play checkers using several online resources, such as https://en.wikipedia.org/wiki/Checkers#General_rules

2.7 Assumptions and Dependencies

- Users have a device capable of running java 8 or later versions
  - The user's device needs to be quick enough to handle java files if that isn't possible.
- Constraints - future game development
- Plan where and how to store data from the user

3. System Features

3.1. Multiplayer

3.1.1. Description and Priority

The user should be able to play against another person or a CPU player. This feature is of high priority.

3.1.2. Functional Requirements

REQ	**Reason**
REQ-1	Player selection

3.2. Stalemate/draw Detection

3.2.1. Description and Priority

The game should be able to detect if there are no possible moves remaining, then proceed by either declaring a winner or a draw. This feature is a high priority.

3.2.2. Functional Requirements

REQ	**Reason**
REQ-1.1	Check if any valid moves are remaining from either player.
REQ-2.1	Count the number of remaining pieces each player has on the board.

3.3. Confirm/undo Move

3.3.1. Description and Priority

The player should be able to confirm or undo their move before the next player's turn.

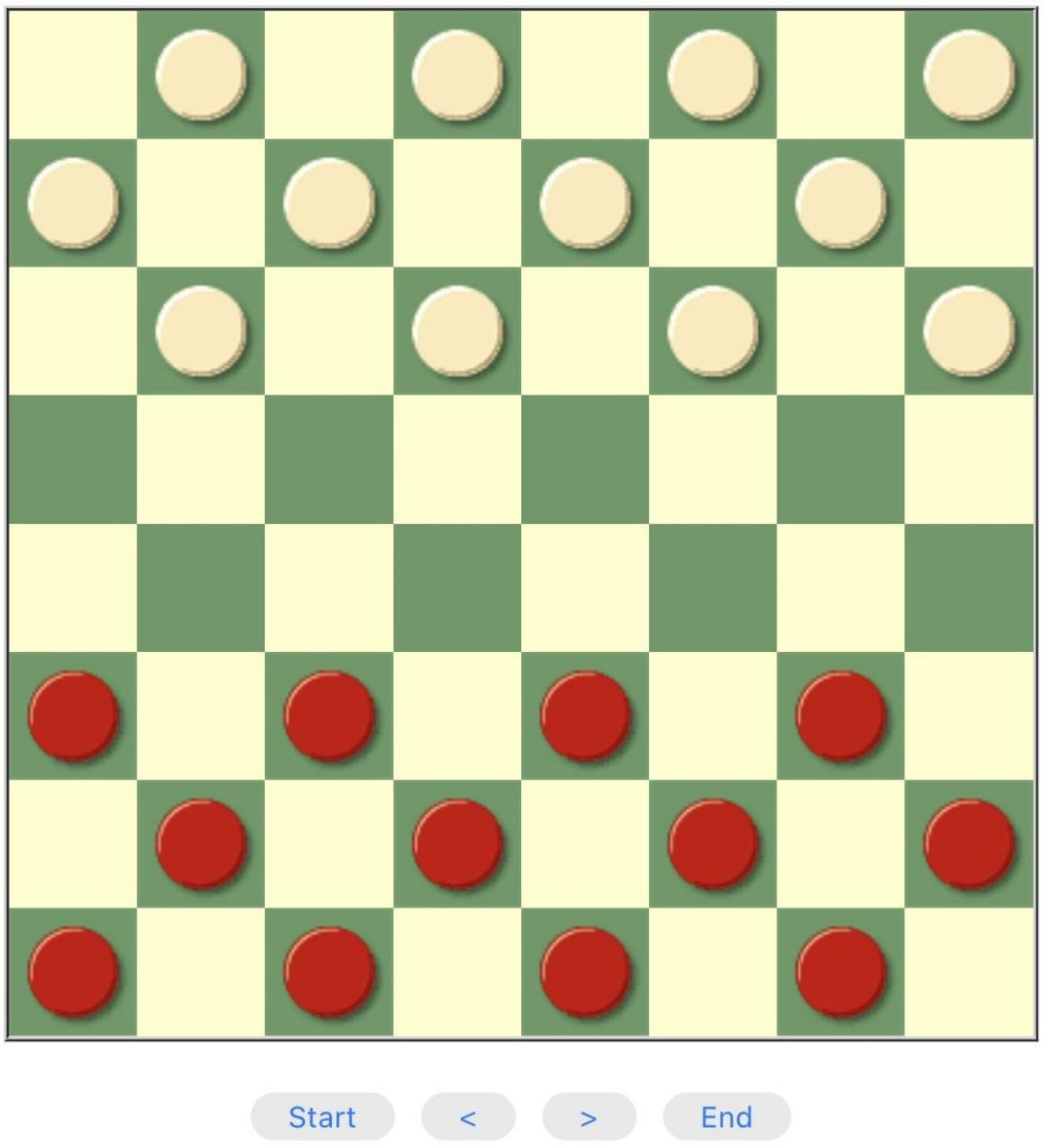
3.3.2. **Functional Requirements**

REQ	**Reason**
REQ-1.2	The player should be able to move their piece.
REQ-2.2	Store the last position of the pieces for the undo function.

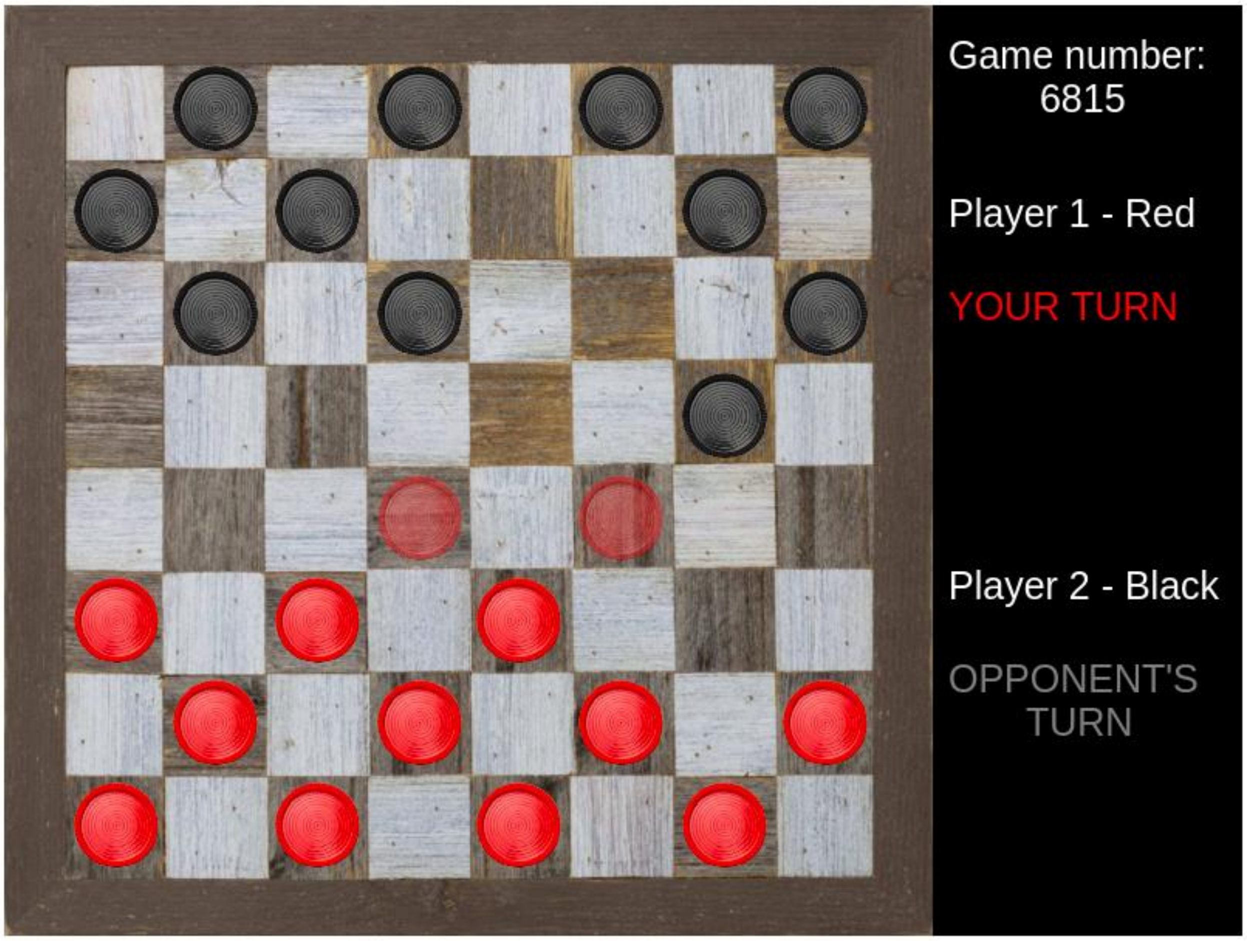
4. External Interface Requirements

4.1 User Interfaces

There will be a start game button, restart button, help button, and exit game button. There will also be a full-sized checkers board in the center of the application.



Additionally, Undo-Restart-Help-buttons will be implemented. These two boards are representations of how we plan our checkers game to look like.



4.2 Hardware Interfaces

Supported devices include desktop and laptop computers.

4.3 Software Interfaces

Checkers game is to be developed for devices that support Java 8 or later versions using a game development tool.

- NetBeans: primary game development library using GUI

4.4 Communications Interfaces

- HTTP/HTTPS

- xxx@checkers-website.com - domain-oriented email address.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Performance shouldn't be a problem given what the current operating systems are capable of. However, operating systems with less powerful hardware and older versions could have issues and possibly operate more slowly. The design will be adjusted to provide a pleasurable experience on the systems offered. The game's functioning will be simple enough to understand, and the graphics won't be extremely complex to avoid slowing down the system.

5.2 Safety Requirements

None of the other applications installed on the user's system will be impacted or damaged by the Checkers game. The user's operating system won't be harmed by the game because it will not cause overheating to the user's computer.

5.3 Security Requirements

The user will not be required to provide any personal information for the checkers game, making it impossible for it to jeopardize such information. Playing the game does not require user login. Simply downloading our application will allow the user to begin playing checkers. The user's operating system will be available to any illegal player who obtains it, giving them access to the Checkers game. The user is responsible for making sure that nobody else will be able to access their operating device.

5.4 Software Quality Attributes

Checkers will react to the user's commands promptly to ensure dependability and accuracy. After every game of Checkers, we'll strive to develop a feature that saves the user's progress for flexibility and adaptation. In this manner, the user can restart the game at a respectable beginning point in the event that the user's operating device experiences a power outage or other problems while the user is playing. The graphical user interface will be easy for the user to use in terms of usability. The user's screen will essentially show all play possibilities in our game, which does not favor one over the other in terms of usability or learning. In terms of availability, Checkers will be available to the user at any time. Moreover, for reliability our game should not cause any crashes to a user’s system.

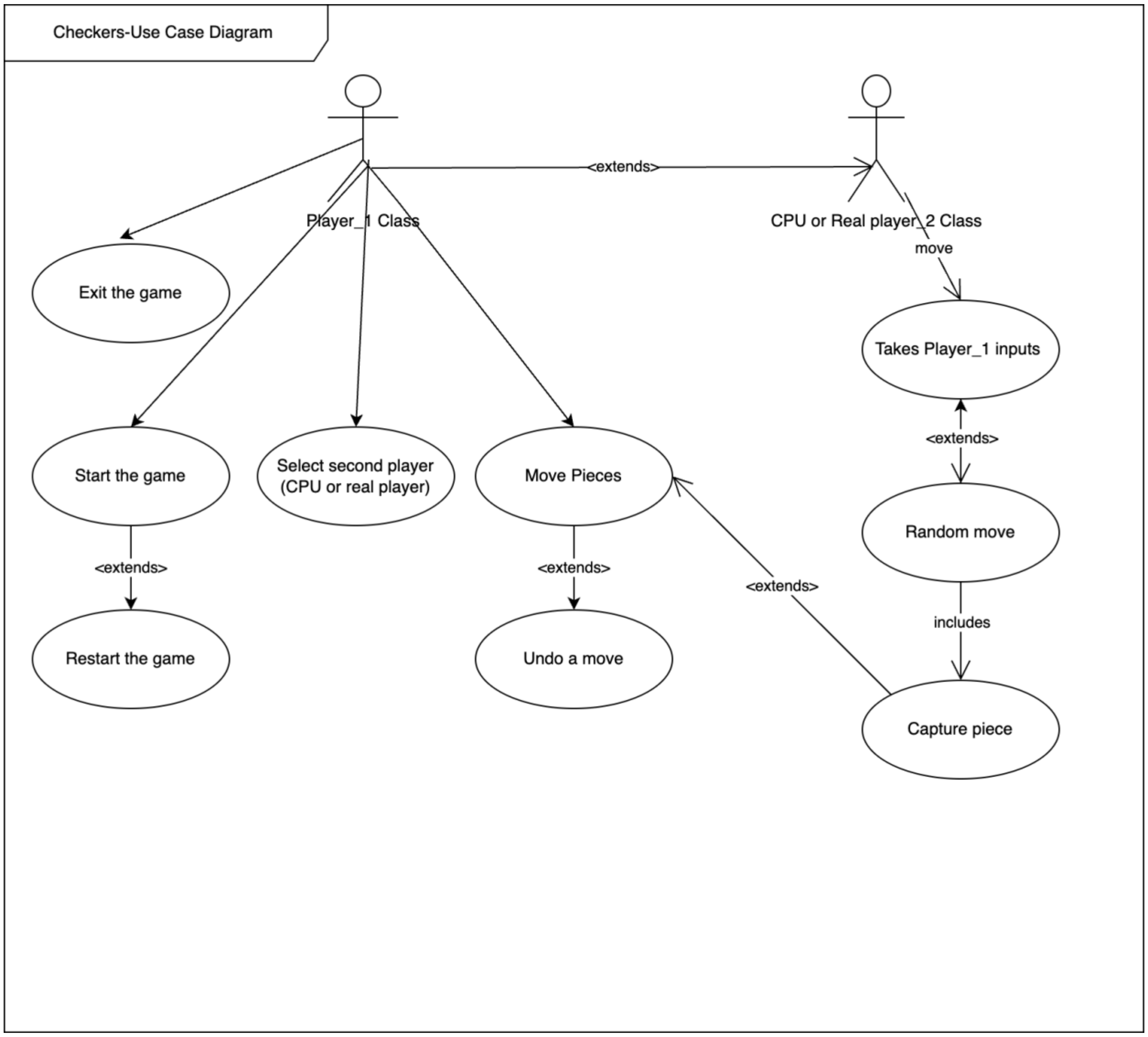
6. Other Requirements

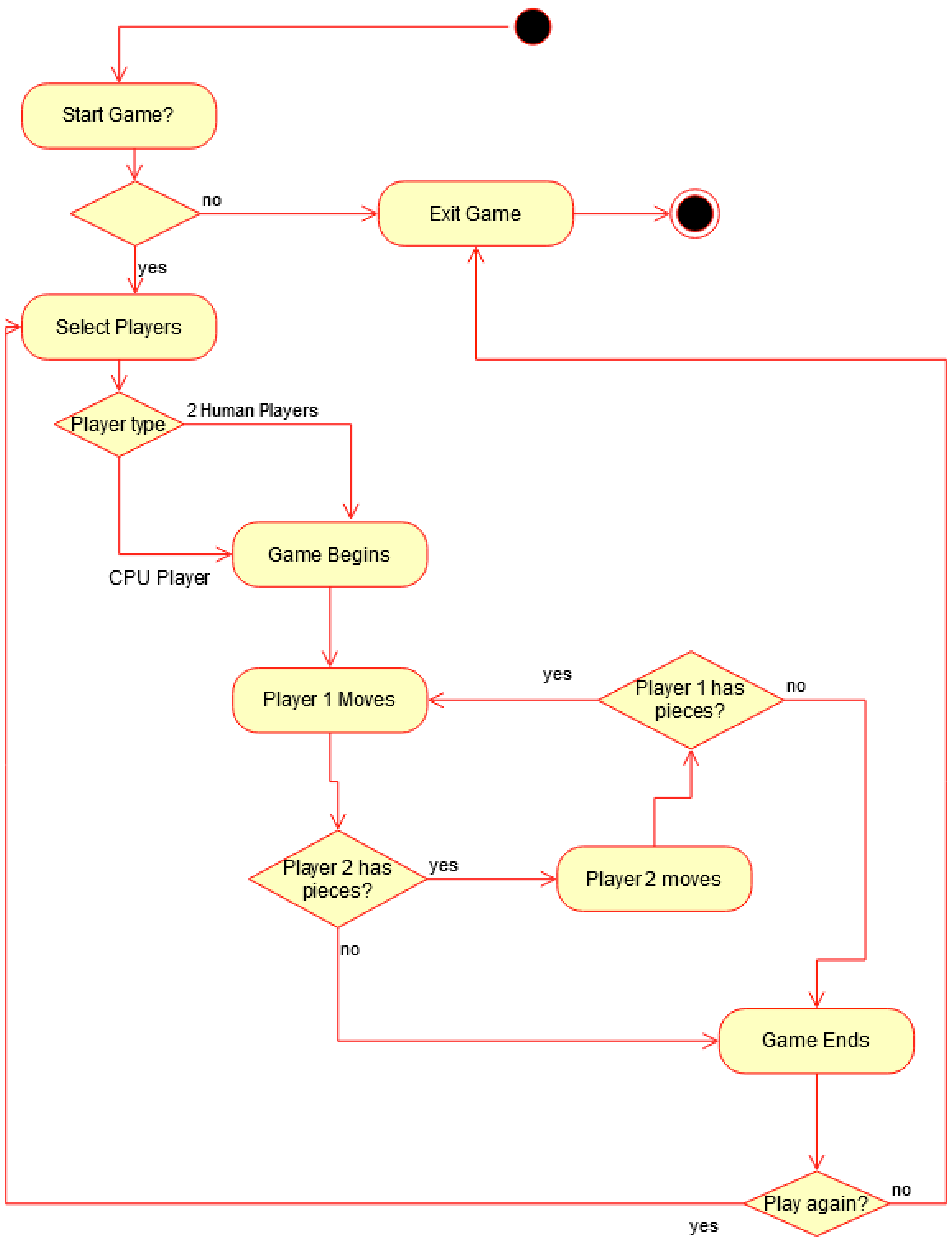
There are no other requirements for this project since it is to be a simple structured game and design with minimal requirements for the user as well as from the user interface. For the legal requirements, it was discussed and described in Security Requirements in section 5.3.

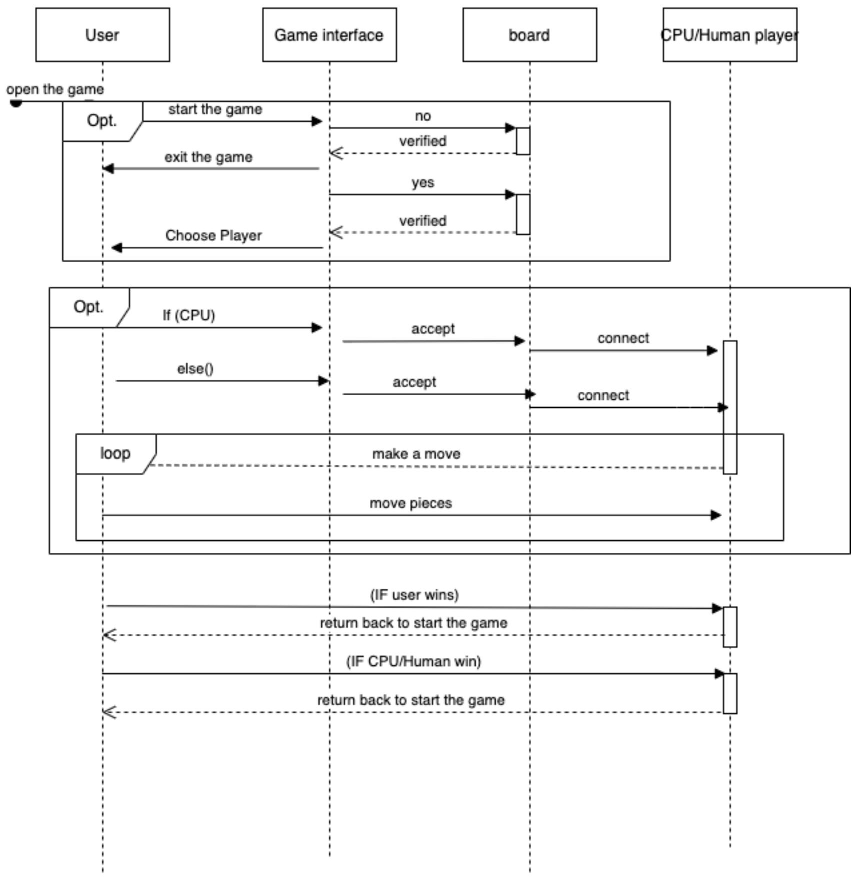
Appendix A: Glossary

As of now, phase 1 does not include any terms to properly interpret the SRS.

Appendix B: Analysis Models




Start Game?
no
Exit Game
yes
Select Players
Player type
2 Human Players
Game Begins
CPU Player
Player 1 Moves
yes
Player 1 has pieces?
no
Player 2 has pieces?
yes
Player 2 moves
no
Game Ends
Play again?
no
yes



Appendix C: Issues List

Potential requirements issues that remain to be solved will be how to undo pieces, store data in arrays, and implement two human players.

GIT Repositor for our project

https://github.com/saparova2022/CS_491_Project_1.git