

Software Requirements Specification

for

CS320 Project Super Check

Version 0.1

Prepared by

Gamers Rise Up

|  |  |  |
| --- | --- | --- |
| Quinlan Boney | 11651240 | [q.boney@wsu.edu](mailto:q.boney@wsu.edu) |
| Keegan Sanchez | 11641250 | [keegan.sanchez@wsu.edu](mailto:keegan.sanchez@wsu.edu) |
| Riley Moreland | 11576770 | [riley.moreland@wsu.edu](mailto:riley.moreland@wsu.edu) |

|  |  |
| --- | --- |
|  |  |
| Date: | 11/06/2020 |
|  |  |
|  |  |
|  |  |

Content

Table Of Contents

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 2

1.5 Document Conventions 2

1.6 References and Acknowledgments 2

2 Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 5

3.3 Behavior Requirements 6

4 Other Non-functional Requirements 7

4.1 Performance Requirements 7

4.2 Safety and Security Requirements 7

4.3 Software Quality Attributes 7

Appendix A – Data Dictionary 8

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Rough Draft, Number 1.0 | Riley Moreland  Keegan Sanchez  Quinlan Boney | Initial rough draft. | 11/05/2020 |
| Rough Draft, Number 1.1 | Riley Moreland  Keegan Sanchez  Quinlan Boney | Rough draft expanded upon. | 11/05/2020 |
| Rough Draft,  Number 1.2 | Riley Moreland  Keegan Sanchez  Quinlan Boney | Rough draft expanded upon. | 11/06/2020 |
| Final Rough Draft, Number 1 | Riley Moreland Keegan Sanchez Quinlan Boney | A finalized version of our rough draft. | 11/06/2020 |
| Final Draft, Number 1 | Riley Moreland Keegan Sanchez Quinlan Boney | A finalized version of our SRS. | 12/16/2020 |

# Introduction

This project is an online version of a board game that has an associated site with information regarding the game. In this section readers will find information assisting how to interpret and understand the product.

## Document Purpose

The product this document describes is the boardgame Checkers and website associated with it that we will design. This document is intended to describe the overall system of the product created by our group. This should include the general description of the product itself, how it should behave, and what it must include.

Revision/Release Number: 0.2

## Product Scope

This software is a virtual representation of the game Checkers. The goal is to have a fully functioning site in addition to the product that provides context to the software. Our objectives are to implement multiple game sessions, rules and information, and the game itself. The benefit of this software is an accessible checker game that is free to play.

This software will require access to a machine capable of using the internet on a modern browser, we will be using a combination of Node.js and WebSocket to have multiple game sessions. Games will be capable of being played online and then updating the associated pages when needed. We plan to have an associated username to a user and unique game codes for each session. The game will be browser based, and feature online multiplayer, as well as being free to play.

## Intended Audience and Document Overview

Intended Audience: Client, Professor, Testers

The rest of the document features a more in-depth description of the product in section 2. Section 3 which focuses on the necessary requirements of the product, section 4 which has listed non-requirements and section 5 which are optional requirements.

The client will want to focus on sections 3 and 4 first as to ensure their expected or unexpected demands are met. For the professor reviewing the entire document from start to finish would be the best course of action save section 5. Testers should read the document in the order of 3,4,5 as they want to ensure the functionality of the product.

## Definitions, Acronyms and Abbreviations

| Term | Definition |
| --- | --- |
| DO | DigitalOcean |
| FR | Functional Requirement |
| GRU | Gamers Rise Up |
| ID | Identification |
| Pages | The landing page that is also connected to the leaderboards and game pages. |
| UI | User Interface |

## Document Conventions

The document uses Arial font throughout. Sections are written in size 11, with headings bolded, and using size 14. The text is all single spaced, with a 1” margin.

## References and Acknowledgments

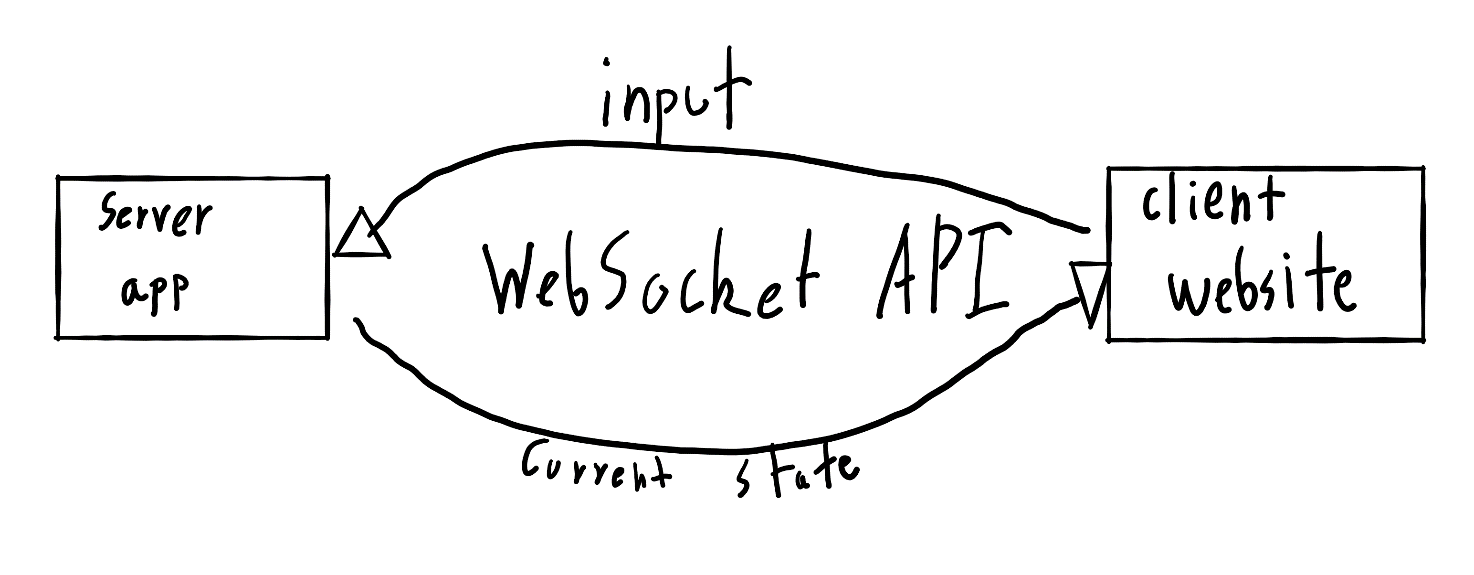
[1] DigitalOcean, Pricing, 03-Feb-2019. [Online]. Available: https://www.digitalocean.com/pricing/. [Accessed: 06-Nov-2020].

[2] M. D. N. Contributors, “The WebSocket API (WebSockets),” MDN Web Docs, 01-Mar-2020. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\_API. [Accessed: 06-Nov-2020].

# Overall Description

## Product Perspective

We wanted to design a simple online game to play with friends. We are taking inspiration from board games, and attempting to apply this to an online setting, to make it easier to engage with. Furthermore, during Covid-19 quarantine, face to face interaction is limited, so creating an online version of the game will allow those who are separated to still play together.

There are a few key pieces of this project. The first is the game itself. This consists of a server, written in Java, which holds all of the logic for the game, and the client page, written in HTML and JavaScript, which simply sends inputs to the server. Both will use the server will work with Java’s sockets, while the client pages will use the WebSocket API to send information [1]. The second is the website. This will consist of all of the webpages that make up the site including the landing page, tutorial page, and game page.

## Product Functionality

Users will be able to navigate to the game page from the landing page that is linked by the play game option. This will result in users creating their username and then setting up a room ID to start or connect to a match. At this point users can, play the game, finish a game choosing to leave or restart, or return to the landing page. From the same landing page users will also have access to the other pages that provide information regarding the game consisting of rules or a tutorial.

* Will be able to interact and use the game
* Will be able to play against another User
* Will be able to see rules and a tutorial for the game

## Users and Characteristics

| User | Frequency of Use | Functions Used | Technical Expertise | Access To Source | Education Level | Importance |
| --- | --- | --- | --- | --- | --- | --- |
| Tester | Each phase of development | Current features for testing | Limited/None | No | Any | 3rd |
| Professor | After completion | All | Sufficient or Above | Yes | Professor | 2nd |
| Client (Player) | After completion | All | Limited/None | No | Any | 1st |

## Operating Environment

The web pages will be hosted on an Ubuntu Apache HTTP server, which will handle connecting clients. The pages should display and function well in most modern browsers (including Google Chrome, Microsoft Edge, and Firefox).

The game server will run in Ubuntu, and will be deployed onto a DigitalOcean server.

## Design and Implementation Constraints

The project will use HTML, JavaScript, and Java. The target size of the virtual machine the product will run on is 1 GB of memory, 1 CPU, a 25 GB SSD for storage, and 1 TB of data transfer per month, with a target hosting cost of 5$ a month [2]. We will host the web pages on an Apache2 server on Ubuntu, and the server will be written in Java and run on an Ubuntu. We will be using sockets with the TCP protocol to transfer information. We will not be maintaining software after the final release.

## User Documentation

The product will have a landing page consisting of a brief tutorial on playing the game and connecting to other users. This brief tutorial should provide detail describing the rules and expectations of the game as well as how to interact with the software. It should then have another section dedicated to explaining how to interact with other players with the menus.

## Assumptions and Dependencies

We are assuming that we will be able to host on DigitalOcean using the 5$ tier [2].

# Specific Requirements

## External Interface Requirements

### User Interfaces

The user interface will be the website consisting of

A landing page, with links to a tutorial, and game.

A leaderboard page with a return link to the landing page.

The game, which will consist of a few entries and choices:

* The username prompt, which asks for an input from a user to associate with. This will then ask you to enter a room id and choose to join/host.
* There will be menus, which have the options, return to home, restart the game, or exit to the username input.

### Hardware Interfaces

This project will require the user to make clicks using a mouse and enter text using a keyboard.

### Software Interfaces

The Apache2 server will handle managing the website. Our Java server will use the Java libraries to send and receive information. We will be using WebSockets to handle sending and receiving data from the client. The server will be written in Java, and use the standard Socket library [1]. The client will interact with the website using their computer, using the mouse to interact with the game

### Communications Interfaces

We will be using the WebSocket API to handle sending information from the client webpage to the server, and HTTP to send webpages. The WebSocket API uses TCP for communication, so this is the standard we will be using [1]. As no important user data is being transferred, we will not use any encryption.

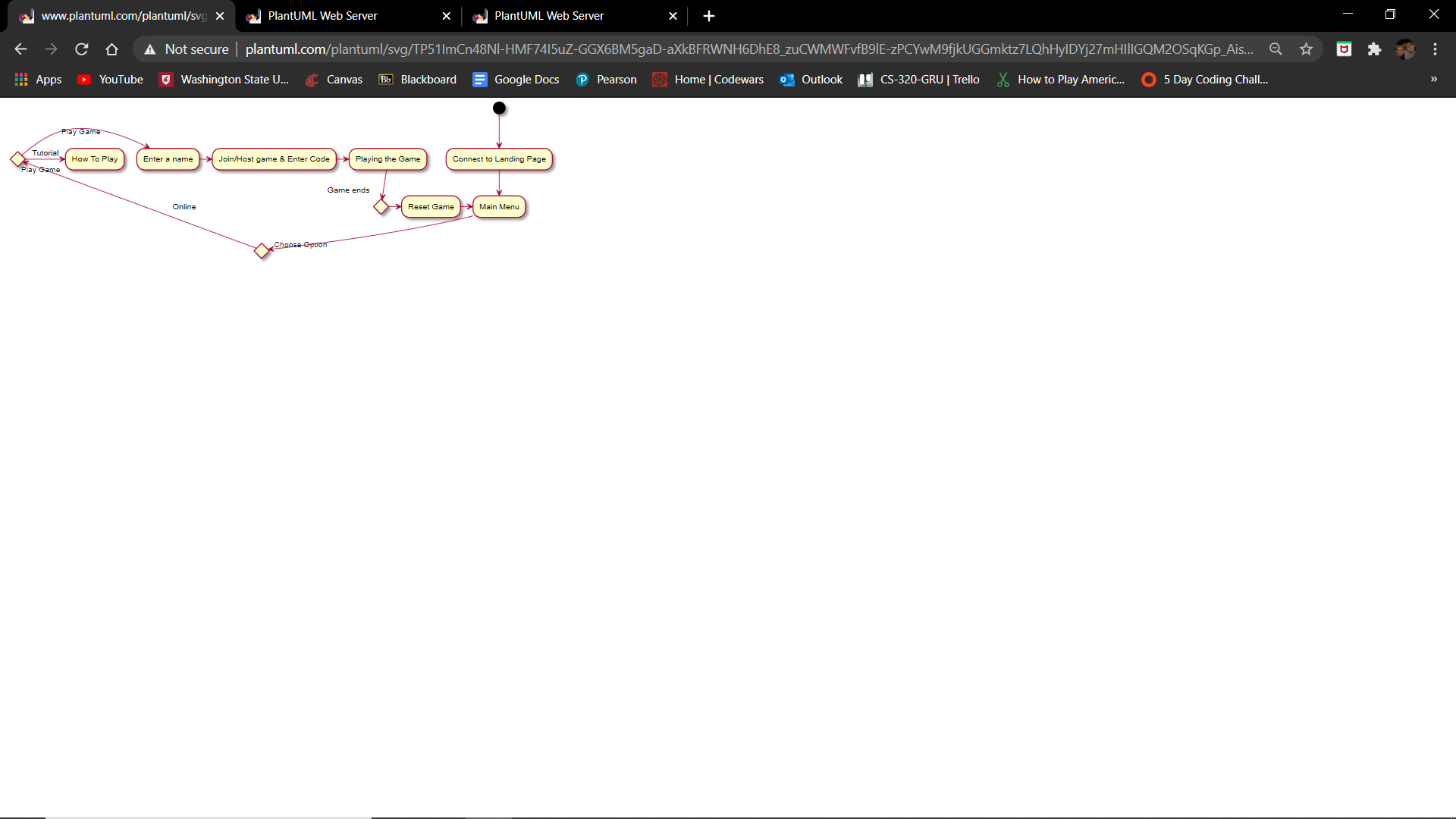
## Functional Requirements

* Website – FR1
  + Landing Page – Links to the other pages and a short description.
  + Tutorial Page – A description of our rules and how to navigate the site.
  + Game Page – The client page, that displays the game to the client
    - Packet Handling – Sending and reading information from the server
* Game Server – FR2
  + Game Logic – The code for updating the game state
  + Packet Handling – Sending and reading information from the client

## Behavior Requirements

### Use Case View

**Using the Landing Page**



# Other Non-functional Requirements

## Performance Requirements

The performance of our product must consist of, web pages will be served quickly (less than a second), applying user’s input should be done quickly (less than a second), the multiplayer connection should be established and working.

## Safety and Security Requirements

No important user data is being transferred, so we will not be using any encryption methods. As this is a simple project, we will not be designing net code intended to prevent malicious users from cheating.

## Software Quality Attributes

### Usability

The product will adhere to performance requirements; we will focus on writing clean efficient code in order to achieve this. Furthermore, the site will be easily navigable, with large icons that stand out. This means using simplistic graphics, and icons that contrast with the surroundings.

### Testability

The server and client will output logs in text format containing important information (sent/received data, current state of the game) for debugging.

Appendix A – Data Dictionary

FR2 – Game Server:

| Name | Access | Description | Input | Output |
| --- | --- | --- | --- | --- |
| constructor | Public | Initialize our server | N/A | N/A |
| notifyRoomOfTurn | Public | Notifies player it is their turn | Room ID | N/A |
| sendBoardTo | Public | Sends board state to each client | Room ID, Peer ID | N/A |
| notifyRoomOfNames | Public | Updates player name UI | Room ID | N/A |

Room Class:

| Name | Access | Description | Input | Output |
| --- | --- | --- | --- | --- |
| constructor | Public | Initialize new room session | ID and Host | N/A |
| joinPlayer | Public | Joins player to a room session | ID | Returns team ID, 0 if none |

Game Client-

| Name | Access | Description | Input | Output |
| --- | --- | --- | --- | --- |
| constructor | Public | Initialize a new game client | UI, URL | N/A |
| sendMove | Public | Sends a valid move to server | Room ID, piece ID, x, y and isJump | N/A |
| onRecieve | Public | Notifies player if it their turn or not | Message | N/A |

Board class:

| Name | Access | Description | Input | Output |
| --- | --- | --- | --- | --- |
| constructor | Public | Initialize a new board | N/A | N/A |
| printBoard | Public | Print board as text (debug purposes) | N/A | N/A |
| clearBoard | Public | Removes all pieces | N/A | N/A |
| debugBoard | Public | Sets a board for debugging | N/A | N/A |
| fillBoard | Public | Adds pieces to the board | N/A | N/A |
| getMoves | Public | Lists all moves a piece can make | Piece ID and distance | List of all valid moves |
| getJumpedPiece | Public | Calculates position of piece being jumped | startPos, endPos | Returns the piece that gets jumped |
| doJump | Public | Jumps current piece and removes jumped piece | startPos, endPos | Returns true if jump was successful, false otherwise |
| getValidPieces | Public | Calculates pieces that have moves | Team ID | Return list of valid pieces |
| movePiece | Public | Moves or jumps a piece | Piece ID, newX, newY | Return true if a piece was jumped, false otherwise |

Appendix B - Group Log

**Git Link:**

https://github.com/Lurgypai/CS-320-GRU

**Timestamps:**

11:00am – 12:00pm, 10/02/20 : Setup Git repository and read over SRS

11:00am – 12:10pm, 10/23/20 : Used discord to introduce research and findings necessary for project between group members

11:00pm – 12:30pm, 10/30/20 : Used discord to discuss and clarify our ideas

3:10pm – 5:15pm, 11/5/20 : Used Discord to discuss and collaboratively work on the SRS

1:30pm – 5:00pm, 11/6/20 : Used Discord to discuss document collaboratively work on the SRS

9:50pm – 11:30pm, 11/6/20 : Used Discord to discuss document collaboratively work on the SRS