**Software Requirements**

**Functional Requirements**

#1

**Description:** The software shall implement the basic game of chess.

**Rationale:** Our software is an implementation of the classic board game.

**Testing Criterion:** All pieces should be able to move exactly like their real-world counterparts.

#2

**Description:** The software shall allow two players to play simultaneously over the internet.

**Rationale:** Chess is a two-player game - users should be able to play seamlessly over the internet

with other players.

**Testing Criterion:** Players should be able to put in a username.

**Requirement Dependencies:** #1

#3

**Description:** The software shall contain a virtual chess game clock.

**Rationale:** Competitive chess is usually played with a game clock, thus our software shall

implement this functionality.

**Testing Criterion:** When a player hits their clock, the timer for the other player should start.

**Requirement Dependencies:** #1

#4

**Description:** Users shall be presented with a list of available users who have already created a

game.

**Rationale:** When first connecting to the site, users should be able to find other players wanting to

play very easily.

**Testing Criterion:** If a player has started a game and is waiting for someone to connect, then

another user should be able to see that user’s username on their home screen.

**Requirement Dependencies:** #2

#5

**Description:** Users shall be able to create a username when first visiting the site.

**Rationale:** Allows other users to find their friends to play with.

**Testing Criterion:** When inputting a username to search for, that user should be connected to the

correct game.

**Requirement Dependencies:** #2, #7

#6

**Description:** Usernames shall not persist when a player exits the game.

**Rationale:** Users should be able to connect to the person they want to connect to - but the

overhead of creating an account system is beyond the scope of this project.

**Testing Criterion:** Usernames should be wiped from the database whenever a user disconnects

from a game.

**Requirement Dependencies:** #5, #7, #8

#7

**Description:** User shall be able to create a new game and wait for another player to join him/her.

**Rationale:** A user needs to start a game which will show up in the list of available players so a

multiplayer game can be initiated.

**Testing Criterion:** A game is created whenever a user click on Create a Game.

**Requirement Dependencies:** #2, #7

#8

**Description:** Users shall be able to join a game from the list of available players.

**Rationale:** If a user has already created a game, a different user should be able to join that game.

**Testing Criterion:** A game starts whenever a user joins a game created by another user.

**Requirement Dependencies:** #2, #7

#9

**Description:** Users shall be able to end a game while they are playing.

**Rationale:** Users should be able to quit out of a game if the other player has gone idle, or they

simply want to leave and find another player to play with.

**Requirement Dependencies:** #2, #8

#10

**Description:** Users shall be able to log out of the game.

**Rationale:** When a user leaves the webpage, their username will be freed up so another user can

use it. This allows us to not worry about implementing an entire account system to

accomodate the game, but also allowing the software to uniquely identify each user.

**Testing Criterion:** When a user exits the browser page, their username should be freed up so

another person can come in and log in with the same name.

**Requirement Dependencies:** #5, #6

#11

**Description:** Users shall be able to log in to the game.

**Rationale:** Users should be able to use a uniquely identifying moniker so other users can find them

easily - We will not be storing passwords or anything, and there will be no registration.

**Testing Criterion:** When a user picks a username upon first visiting the page, that username

should not be usable by another person trying to log into the software.

**Requirement Dependencies:**#2, #8

**Non-functional Requirements**

#12

**Description:** The software shall be implemented on top of Node.js so that a persistent, stateful

connection can be made between players.

**Rationale:** Using this technology allows us to circumvent the inherent statelessness of the

request-response model that HTTP is built on.

**Requirement Dependencies:** #2

#13

**Description:** The software shall be implemented using Express server.

**Rationale:** To create connections between users in the client-server model, we will need a server

implementation to use with Node.js. Enter Express.

**Requirement Dependencies:** #2, #12

#14

**Description:** The software shall be implemented using socket.io.

**Rationale:** We will need a real-time connection between players, and will be using socket.io to

circumvent the request-response HTTP is built on.

**Requirement Dependencies:** #2, #12, #13

#15

**Description:** The software shall be implemented using the Strategy/Policy Design Pattern.

**Rationale:** Using the Strategy/Policy Design Pattern will allow the same game core to be used

with other modules (Chess, Halma, Checkers, etc.).

#16

**Description:** The software shall be written in JavaScript

**Rationale:** Since we will be using Node.js as our server, the game will need to be implemented

using Javascript as our scripting language.

**Requirement Dependencies:** #12, #13, #14

#17

**Description:** Nothing shall persist when the game is closed.

**Rationale:** When the user leaves the website or logs out, their username will be freed from the

system so that another user can use that same name. No leaderboards or the like shall

exist that need to persist when users are logged out.

**Testing Criterion:** Nothing should be saved on the server besides the usernames of those playing

and the configurations of their games - namely, which lobbies are open.

#18

**Description:** The software shall be self-contained in the browser.

**Rationale:** Nothing shall exist outside of the browser page - nothing shall be downloaded, and the

player does not need anything when they first visit the web page.

#19

**Description:** The game-board shall be implemented using HTML5 Canvas.

**Rationale:** To get better flow and feel of the game, the software’s game-board shall be

implemented in HTML5 Canvas.

**Requirement Dependencies:** #12, #13, #14, #16, #20

#20

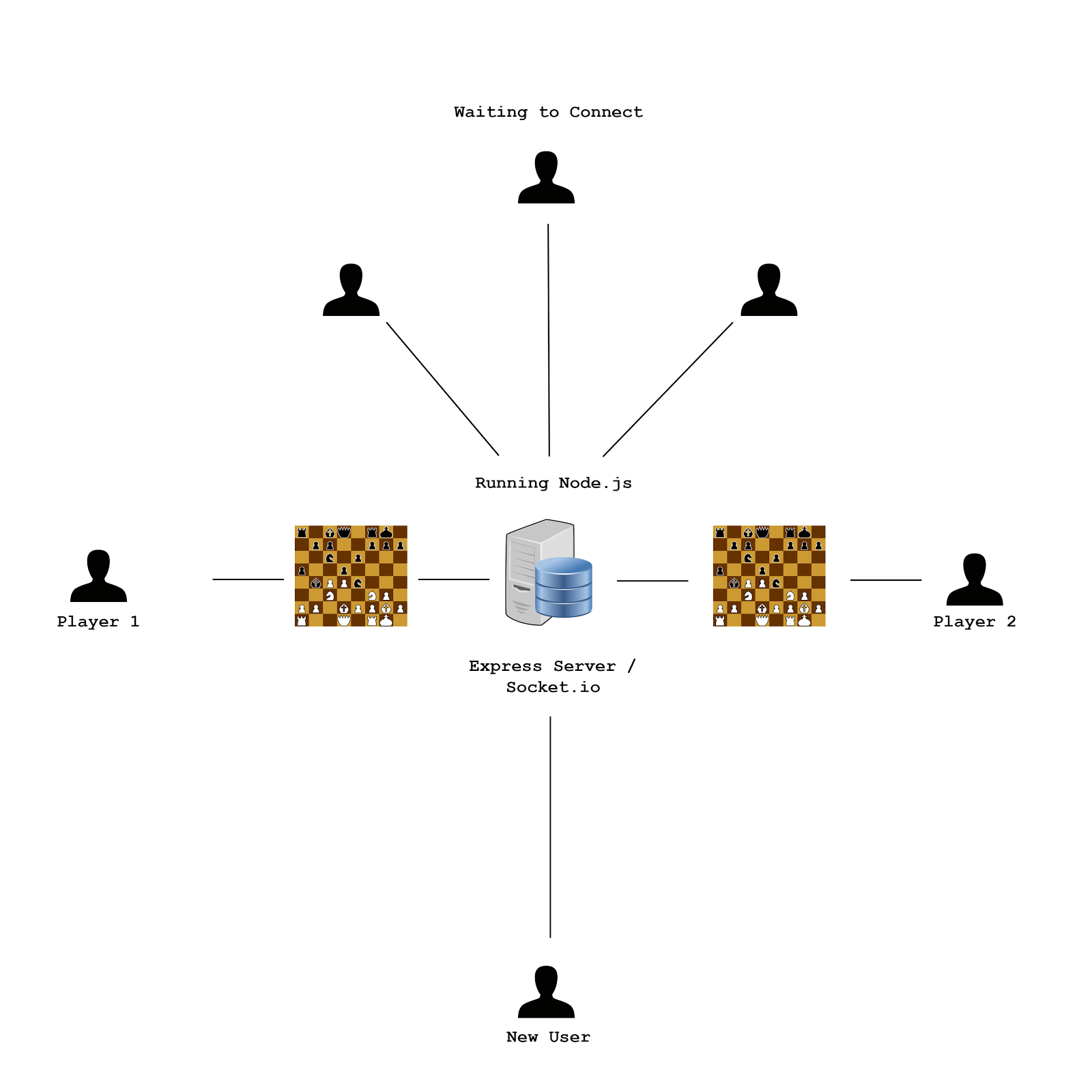
**Description:** The software shall make extensive use of jQuery.

**Rationale:** To ease implementation, we shall be using the jQuery library extensively to implement

things such as pieces moving, playing sounds, and drawing the game-board.

**Requirement Dependencies:** #12, #16, #17

**Software Architectural Diagram**



**Version Control**

**System**

We will be using Git as our version control system.

**Repo URL**

https://github.com/ntaylorbishop/SEPGroupProject