CS451

Checkers Requirements Specification

Summer 2017

Group members	Hajer Karoui, Samuel Nathanson, Curtis Bechtel, Julie Soderstrom
Faculty advisor	Dr. Filippos Vokolos, Ph. D.

Revision History

Name	Date	Reason for change	Revision
Hajer Karoui Samuel Nathanson Curtis Bechtel Julie Soderstrom	07/12/2017	First Draft - Sections outlined	0.5
Hajer Karoui Samuel Nathanson Curtis Bechtel Julie Soderstrom	07/30/2017	Second Draft - Content added to all sections	1.0
Hajer Karoui Samuel Nathanson Curtis Bechtel Julie Soderstrom T. Adams	08/01/2017	Third and final Draft - Revisions from other students' feedback.	1.1

Table of Contents

1. Introduction

1.1. Purpose of the document

This document describes the specification requirements for the real-time two-player checkers game accessed remotely. It is used as a reference guide by customers or end users and developers in order to establish a common understanding of the final deliverable.

1.2. Scope of the document

This document will serve as a reference guide for developers during the coding process of the checkers game. It can also be used in the future by end-users and/or stakeholders in order to compare set expectations and requirements with the final product

1.3. Overview of the document

In this document, the reader should expect detailed information about functional and nonfunctional requirements, user interface guide and description, use cases defining the various interactions between the actor (user) and the system throughout different steps of the game, glossary and a references section. The requirements will describe the roles of the client-side application and the server side.

1.4. Definitions

- **A game** consists of a standard checkers board, pieces, and at least one online player. (further describe standard checkers board and pieces)
- **Public games** are displayed on a list to all players interested in joining a game.
- **Private games** are only visible to and accessible by players with the game's private ID key.
- **New games** are ones which do not yet have a second player.
- **Active games** are ones with two online players.
- **Checker piece**: a piece that can be moved and which did not reach the last row yet.
- **King**: a piece that reaches the last row (the King Row).

2. Description

2.1. Product Perspective

This checkers game enables participants to choose between a game open to whoever wants to join, and a private game in which a secure key ID specific to the game needs to be provided by both players. The two opponents play against each other in real-time, remotely. The objective of the game is to deplete the other player's amount of checker pieces through strategy. Each move will be timed and checked for validity.

2.2 User Description

Users of the game are of any age. There should be two players that play against each other remotely. This should be able to be played anywhere as long as they have access to internet.

2.2. Requirements Apportioning

Priority Level	Description
P1	**Priority 1** requirements are essential to the product and must be in the final build. These requirements must be tested and verified to ensure proper functionality
P2	**Priority 2** requirements are not required for the final build, but will be provided if there is sufficient time. The system will be designed such that it is extendable to easily incorporate these requirements in the future.
P3	**Priority 3** requirements are not required, and will not be considered in the design of the system. If sufficient time remains the requirement will be incorporated.

3. Functional Requirements

3.1. Gameplay

R1. Starting a game

A player can initiate a new game by

- **R1.1** Clicking 'Start' **Priority 1**
- **R1.2** Choosing *public* or *private* game mode **Priority 1**
- **R1.3** Choosing their tile and piece color (red or black) **Priority 2**
- **R1.4** Entering their username **Priority 1**
- **R1.5** A player can join a public game by selecting a game from the list of public games and entering their username. **Priority 1**
- **R1.6** A player can join a private game by entering the game's private key ID and entering their username. **Priority 1**

R2. Game Rules

- **R2.1** The player with black pieces is given the first move. **Priority 1**
- **R2.2** Players alternate making moves and cannot make a move until it is their turn. **Priority 1**
- **R2.3** At their turn, a player has a limited time to make their move. Once the allowed time has elapsed, they automatically forfeit the game to their opponent. **Priority 1**
- **R2.4** After a player takes one step with a single piece, their turn is over. **Priority 1**
- **R2.5** As soon as player jumps over their opponent's piece, the opponent's piece is then removed from play. **Priority 1**
- **R2.6** After a player makes any number of consecutive jumps with a single piece, their turn is over. **Priority 1**

R2.7 When a player moves one of their checker pieces onto a square on the first row of the opponent's side of the board, the checker piece becomes a king and the turn is ended. **Priority 1**

R3. Move Validation

R3.1 Valid moves for a checker piece:

R3.1.1 **A step:** One square in a forward diagonal direction if the square is unoccupied. **Priority 1**

R3.1.2 **A jump:** Two squares in a forward diagonal direction if the first is occupied by their opponent's piece and the second is unoccupied. **Priority 1**

R3.2 Valid moves for a King:

R3.1.1 **A step:** One square in any diagonal direction if the square is unoccupied. **Priority 1**

R3.1.2 **A jump:** Two squares in any diagonal direction if the first is occupied by their opponent's piece and the second is unoccupied. **Priority 1**

R4. Endgame

R4.1 If either player runs out of pieces, they lose the game. **Priority 1**

R4.2 If either player cannot make a valid move during their turn before their time elapses, they lose the game. **Priority 1**

R4.3 Either player may leave the page, close the window, or opt to forfeit a game. A player may forfeit at any time, even when it is not their turn. **Priority 1**

R4.4 When a player forfeits the game, the opponent will be notified and the game ends. **Priority 1**

R4.5 Either player may select an option to end the game in a draw. This option can be selected at any time and will give the opponent the opportunity to accept or reject the offer. If accepted, the game ends in a draw. If rejected, play continues as usual. **Priority 2**

3.2. Server

R5. New Game Requests

R5.1 New game requests include data for the username of the player creating the new game and for the private/public status of the game.

R5.2 Receive HTTP requests for a new game. **Priority 1**

R5.3 Generate a unique game ID. **Priority 1**

R5.4 Generate a unique player ID. **Priority 1**

R5.5 Add game data and current game state to database. **Priority 1**

R5.6 Return player ID and initial game state to user. **Priority 1**

R6. Join Game Requests

R6.1 Join game requests include data for the username of the joining player, the game ID. **Priority 1**

R6.2 Receive HTTP requests for joining a game. **Priority 1**

R6.3 Check that the game is still active and waiting for a second player. **Priority 1**

R6.4 Generate a unique player ID. **Priority 1**
R6.5 Update game data in database. **Priority 1**
R6.6 Return player ID and initial game state to user. **Priority 1**

R7. Update Game Requests

R7.1 Update game requests include data for the game ID, the player ID, moved pieces, captured pieces, pieces which have become kings, forfeits, requests to end the game in a draw, and responses to draw requests. **Priority 1**
R7.2 Receive HTTP requests for game updates. **Priority 1**
R7.3 Check that the game ID and player ID match an active game session. **Priority 1**
R7.4 If the request includes a move, also check that it is the player's turn and that the move is valid. **Priority 1**
R7.5 Update game data and game state in database (including the update number and player). **Priority 1**
R7.6 Return a success to player making the request. **Priority 1**

R8. Get Game State Requests

R8.1 Get game state requests include data for the game ID, the player ID, and the last update received. **Priority 1**
R8.2 Receive HTTP requests for game updates. **Priority 1**
R8.3 Check that the game ID and player ID match an active game session. **Priority 1**
R8.4 Return game updates made by the other player. **Priority 1**

4. Non-functional Requirements

R9. Response time

R9.1 Response time after refreshing board must be at most 3 seconds. **Priority 2**
R9.2 Response time after making a move must be at most 3 seconds. **Priority 2**

R10. Capacity

R10.1 The game allows up to two players to play against each other in real-time. **Priority 1**

R11. Deployment

R11.1 The application is deployed as a JS file. Once the user starts the server, they can host or join a game. **Priority 1**

5. UI

5.1. Checkerboard

The 2-D checkerboard will consist of black/white or black/red tiles and pieces, depending on the player's' choice. Each player's view of the board should display the board with their side/pieces on the lower part of the screen. **Priority 1**

5.2. Main screen

The main menu has 2 buttons:

5.2.1. Start" Button

When the user clicks the *Start* button, the right section of the screen displays a form that prompts the user to provide a user name of their choice, a color combination for the checkerboard (Black/White or Black/Red), and a radio button to specify if the game is public or private. **Priority 1**

- **Public Game**

A public game will be displayed on the list of public games to any potential joiner. If the user selects a private game A submit button, which will bring them to the *Game Screen*.

- **Private Game**

The host will be securely provided with a unique key ID to share with their opponent of choice to join the game.

5.3. Public Game list

5.4. Game Screen

The game screen consists of the checkerboard, and 4 buttons:

5.4.1. "Surrender" Button

This button should end the game and "give the win" to the opposing player**. Priority 1**

5.4.2. "End Game in a Draw" Button

Either player may select an option to end the game in a draw. This button can be clicked at any time during the game and will give the opponent the opportunity to accept or reject the offer, using a button. If accepted, the game ends in a draw. If rejected, the game continues. **Priority 2**

*5.4.3. "*Pause" Button

This button locks the game by preventing both players from making any moves and stopping the countdown timer from ticking. In order to prevent the user from using this *pause* feature as a way to get around the timer, an overlapping window will appear on top of the checkerboard to block the view from the board. The new window has a "*Continue*" button for the player who initiated the pause. The opponent will simply wait for the game to be resumed by the other player, having a similar overlapping window on the board. **Priority 2**

5.4.4. "Continue" Button

This button resumes the game after pausing it. **Priority 2**

5.4.5. Information Box

The information box to the side of the board displays the timer countdown, opponents' usernames, and the player currently making the move. **Priority 1**

5.5. End Game Screen

When the game ends, the checkerboard collapses and a new view appears displaying the winner username, statistics of the game such as average time spent per move by each player. At the bottom of the screen, a button "**Back to Main Menu**" takes the player to the main screen. **Priority 3**

6. Use Cases

6.1. Use case flow

6.1.1. Hosting a game

Public Game

Precondition: User clicks on the "Start" button on the main menu and selects the option "Public" to initiate a public game.
Main flow: User enters their username, preferred color combination of the tiles/pieces for the game, selects "Public" mode, and submits the form.
Postcondition: A new public game is created and publicly available in the list of games.

Alternate flow: If the user fails to fill one of the fields of the form, they will be asked in a popup warning window to complete the mandatory information.

Private Game:

Precondition: User clicks on the "Start" button on the main menu and selects the option "Private" to initiate a private game.
Main flow: User enters their username, preferred color combination of the tiles/pieces for the game, selects "Private" mode, and submits the form.
Postcondition: User is provided with a private unique key ID to share with one other player of their choice to join the game.

Alternate flow: If the user fails to fill one of the fields of the form, they will be asked in a popup warning window to complete the mandatory information.

6.1.2. Joining a game

Public Game

Precondition: User clicks on the "Join" button on the main menu. A new form is generated on the top half of the page, along with a list of publicly available games waiting on a 2nd player to join. The form is for the user to fill out information before joining one of the public games.

Main flow: User enters their preferred username, selects a game to join using the radio buttons from the list of games, and clicks “Submit”.
Postcondition: The form is successfully submitted and the user is taken to the game room with the checkers board to start the game.

Alternate flow: If the user fails to fill one of the fields of the form, they will be asked in a popup warning window to complete the mandatory information.

Private Game
Precondition: User clicks on the “Join” button on the main menu. A new form is generated on the bottom half of the page, for the user to fill out information before joining one of the private game. The user has a private key ID provided to them by the host of an initiated private game.
Main flow: User enters their preferred username, enters the private key ID to join the private game, and clicks “Submit”.
Postcondition: The form is successfully submitted and the user is taken to the game room with the checkers board to start the game.

Alternate flow: If the private key provided by the user does not match any of the keys of the currently active private games, they will be notified and prompted to try again/get a valid key. If the user fails to fill one of the fields of the form, they will be asked in a popup warning window to complete the mandatory information.

6.1.3. Making a step with a piece
Precondition: Player still has pieces on the board and it is their turn to make a move. The next square following player’s piece in a forward diagonal direction is unoccupied.
Main flow: Player makes a step of one square in a forward diagonal direction.
Postcondition: Piece advances and checker board is updated.

Alternate flow: If the move is invalid, it is canceled and the player is cautioned at the first instance.

6.1.4. Making a jump with a piece
Precondition: Player still has pieces on the board and it is their turn to make a move. The next square following the player’s piece in a forward diagonal direction is occupied by the opponent’s piece and the second is unoccupied.
Main flow: Player moves piece two squares in a forward diagonal direction by jumping over the opponent's piece.
Postcondition: Player captures the opponent’s piece they jumped over. The captured piece is removed from the board. Piece advances and checker board is updated.

Alternate flow: If the move is invalid, it is canceled and the player is cautioned at the first instance.

6.1.5. Making a step with a king

Precondition: Player still has at least one king piece on the board and it is their turn to make a move. The next square following the player's piece in any diagonal direction is unoccupied.
Main flow: Player makes a step of one square in any diagonal direction.
Postcondition: Piece advances and checker board is updated.

Alternate flow: If the move is invalid, it is canceled and the player is cautioned at the first instance.

6.1.6. Making a jump with a king

Precondition: Player still has at least one king piece on the board and it is their turn to make a move. The next square following the player's piece in any diagonal direction is occupied by the opponent's piece and the second is unoccupied.
Main flow: Player moves piece two squares in any diagonal direction by jumping over the opponent's piece.

Postcondition: Player captures the opponent's piece they jumped over. The captured piece is removed from the board. Piece advances and checker board is updated.

Alternate flow: If the move is invalid, it is canceled and the player is cautioned at the first instance.

6.1.7. Forfeiting the game

Precondition: The game has started and both players still have pieces on the board. It can be either player's' turn.
Main flow: Player selects an option from the menu to forfeit or navigates away from the page (either by closing or refreshing).
Postcondition: The opponent receives a message that the other player has forfeit the game, and the gameplay ends.

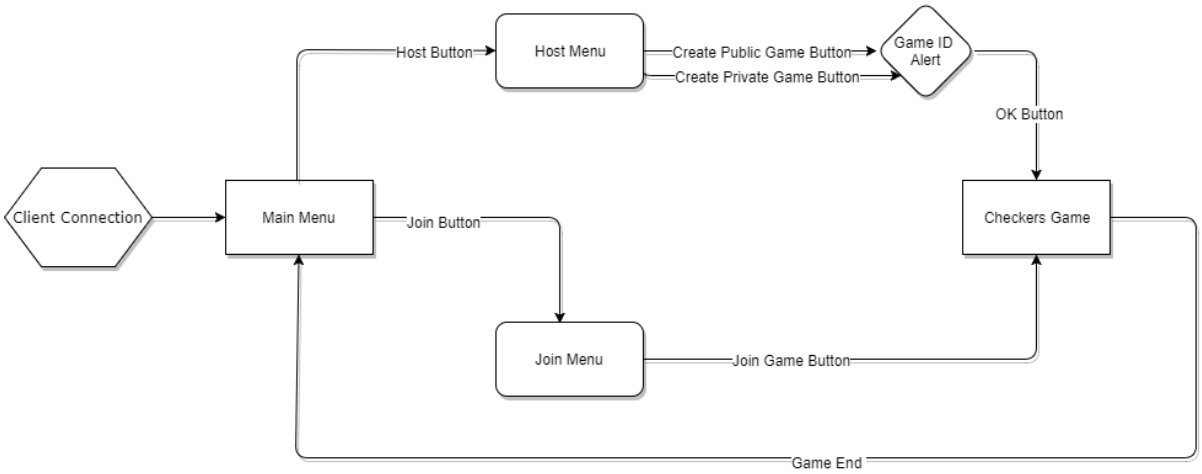
6.1.8. Ending the game in a draw

Precondition: The game has started and both players still have pieces on the board. It can be either player's' turn.
Main flow: Player selects an option from the menu to end the game in a draw. The opponent receives a message prompt offering to end the game in a draw. The opponent selects 'Yes' from the prompt.
Postcondition: Both players receive a message that the game has ended in a draw, and the gameplay ends.

Alternate flow: Player selects an option from the menu to end the game in a draw. The opponent receives a message prompt offering to end the game in a draw. If the opponent selects 'No' from the prompt, the message is closed and the game continues as normal. The original player receives a notification that the offer to draw was rejected.

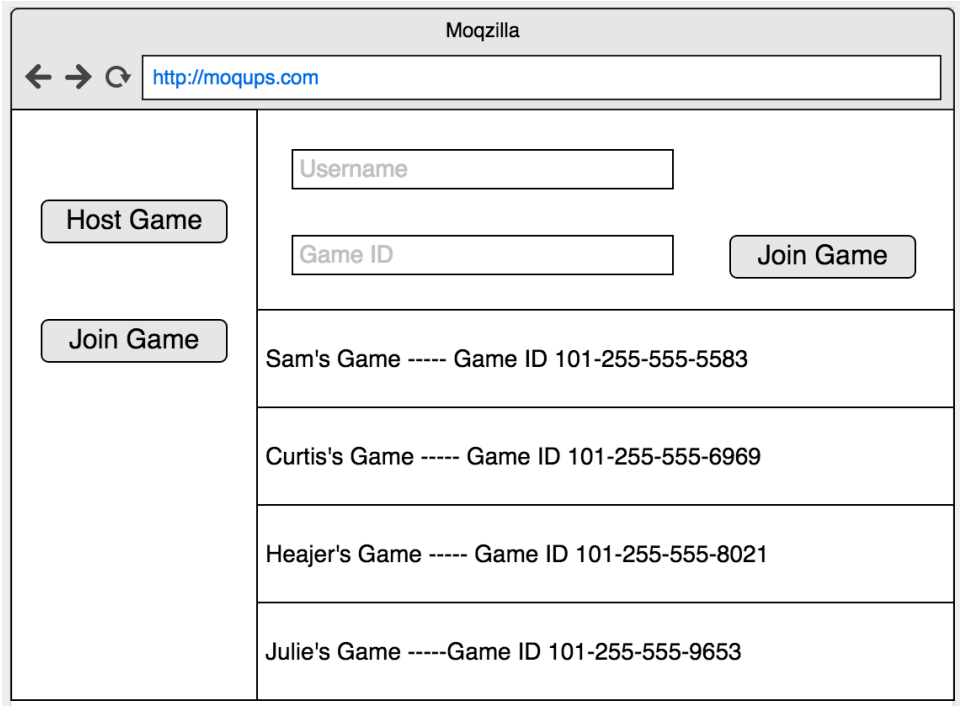
7. Activity diagram

7.1. User Interface Mockups

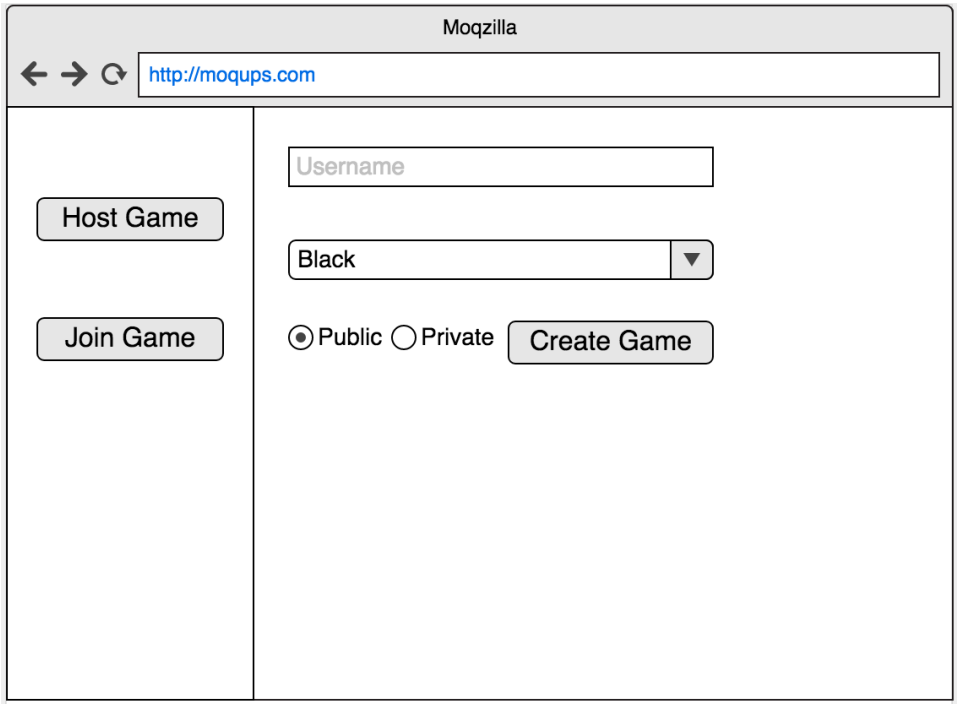


7.2. Main Menu

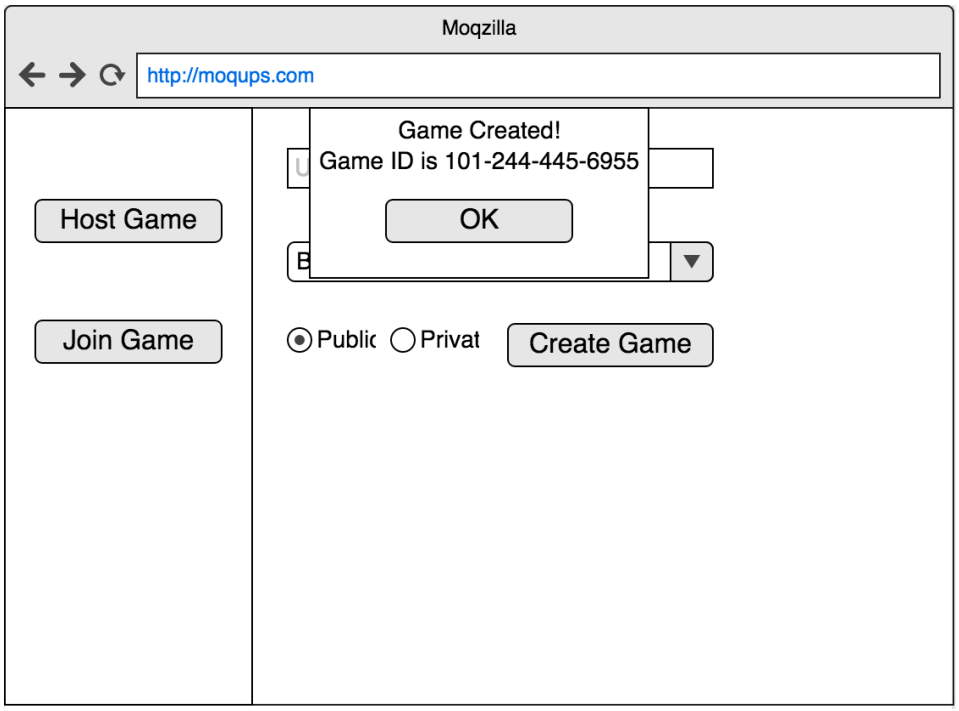
"Join Game" button is clicked



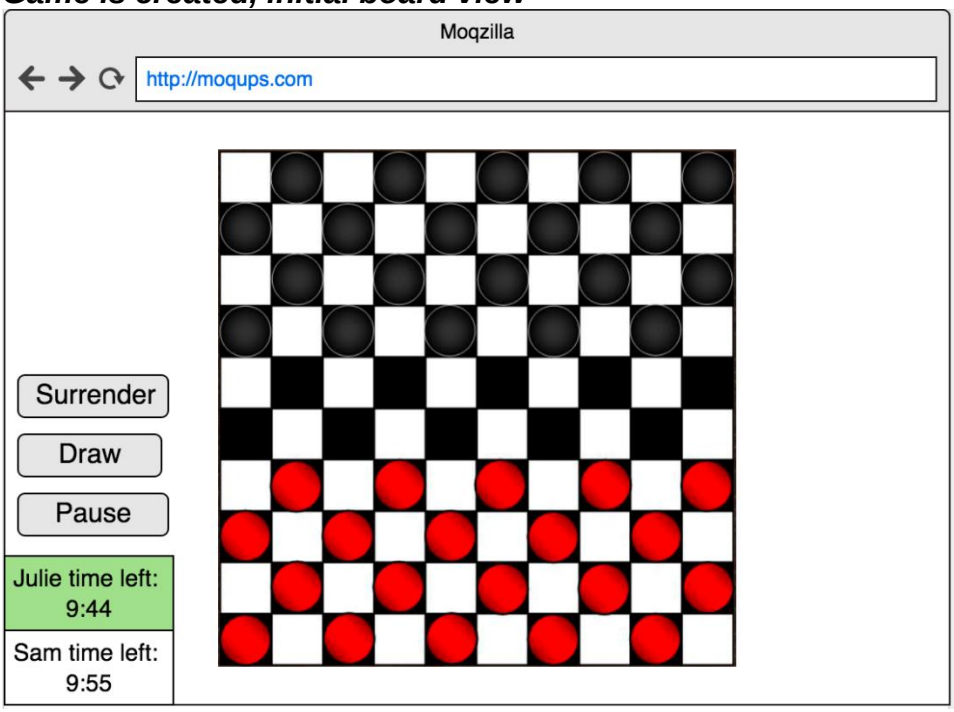
"Host Game" button is clicked



"Create Game" button is clicked



Game is created, initial board view



8. References

- https://www.itsyourturn.com/t_helptopic2130.html
- http://keyja.com/help/play_checkers.html
- http://www.checkerschest.com/play-checkers-online/fundamentals2.htm
- **http://gotwarlost.github.io/istanbul/**
- https://stackoverflow.com/questions/300855/javascript-unit-test-tools-for-tdd
- http://karma-runner.github.io/1.0/index.html