

PROJECT Design Documentation

Team Information

- Team name: Team D
- Team members
 - Andrew
 - Navid
 - Alex
 - Nick

Executive Summary

This project is an online web checkers application. The game allows two players to play a game of checkers adhering to standard rules. Players can see other online players and attempt to join a game. The game will continue until one player wins, or one player resigns.

Purpose

The goal of this project is to allow users to carry out an online game of checkers.

Glossary and Acronyms

Term	Definition
VO	Value Object

Requirements

The primary feature of the application is to start an online checkers game between two players. When a player connects to the web server they will be prompted to sign in to a player account. Once then they will be presented with a list of other online players. If that player is not in a game the user can create a game with that player. The webapp will then generate the checkers board, with the each player's pieces in their respected positions. A standard game of checkers will then begin with each player taking turns moving their pieces on the board. Each move is validated before it is completed. The players will switch off after every move. If a player jumps over an enemy piece, that piece is removed. If a player reaches the other side of the board with a piece, that piece becomes a king and gets access to king moves. The game will continue until one player loses all of his pieces, or a player resigns.

Definition of MVP

The minimum viable product for this application consists of being able to successfully start a game and perform valid moves in order to carry out the game between two players. Each player should be able to take alternating turns, and each turn should be validated.

MVP Features

The following Epics, are features that are required for the MVP

Epic - Start Game
 Epic - Move Piece
 Epic - Take Turn
 Epic - Ensure Valid Move

Roadmap of Enhancements

The following Epics are product enhancements to be implemented

Epic - Resignation

Player Forfeit

Epic - Player Spectator Mode

Spectator Count

Exit Spectator Mode

Chat Box

Epic - Multiple Games

Additional Connection

Seamless Play

Application Domain

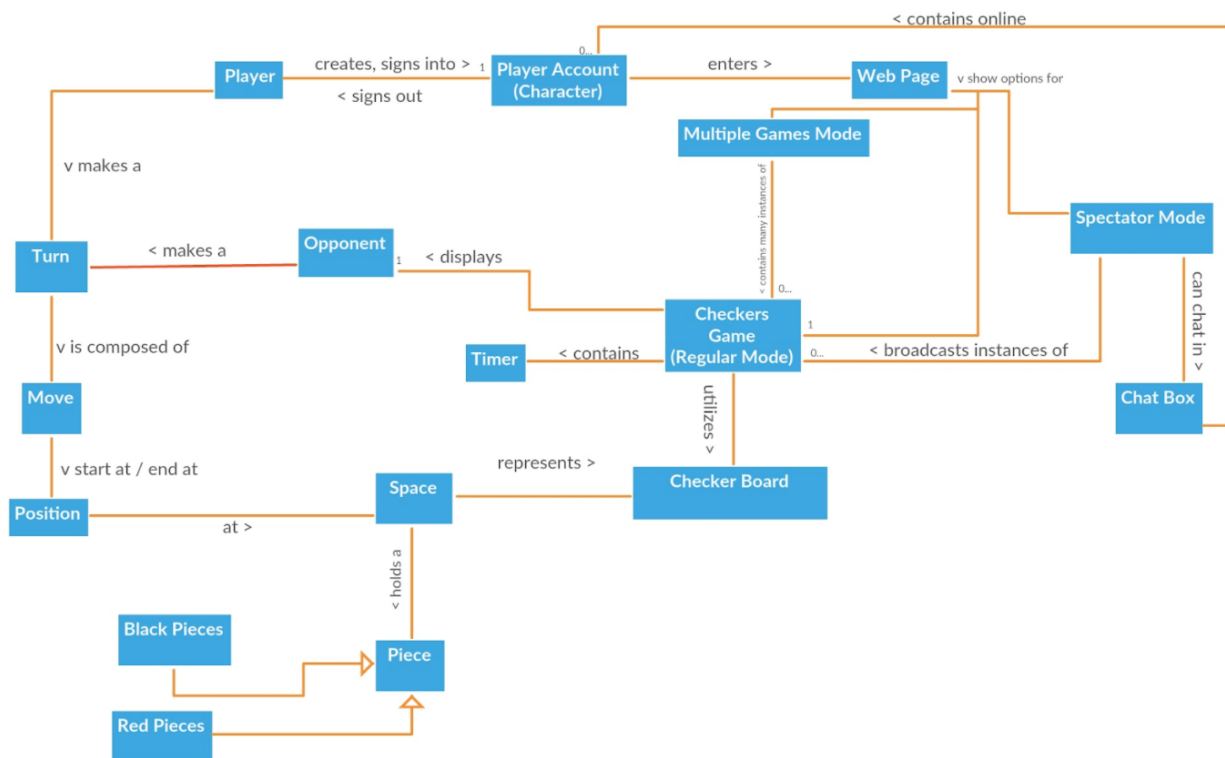


Figure 1: The WebCheckers Domain Model

The figure above represents the application domain of the project. The player starts by signing into a player account, which then enters the webpage. The player can then choose whether they want to start a standard game, start a multiple game, or spectate. Regardless of the choice the application launches a checkers game mode. The game mode contains the board, which in turn contains the various game pieces, which in turn represent the player.

Architecture

Summary

The following Tiers/Layers model shows a high-level view of the webapp's architecture.

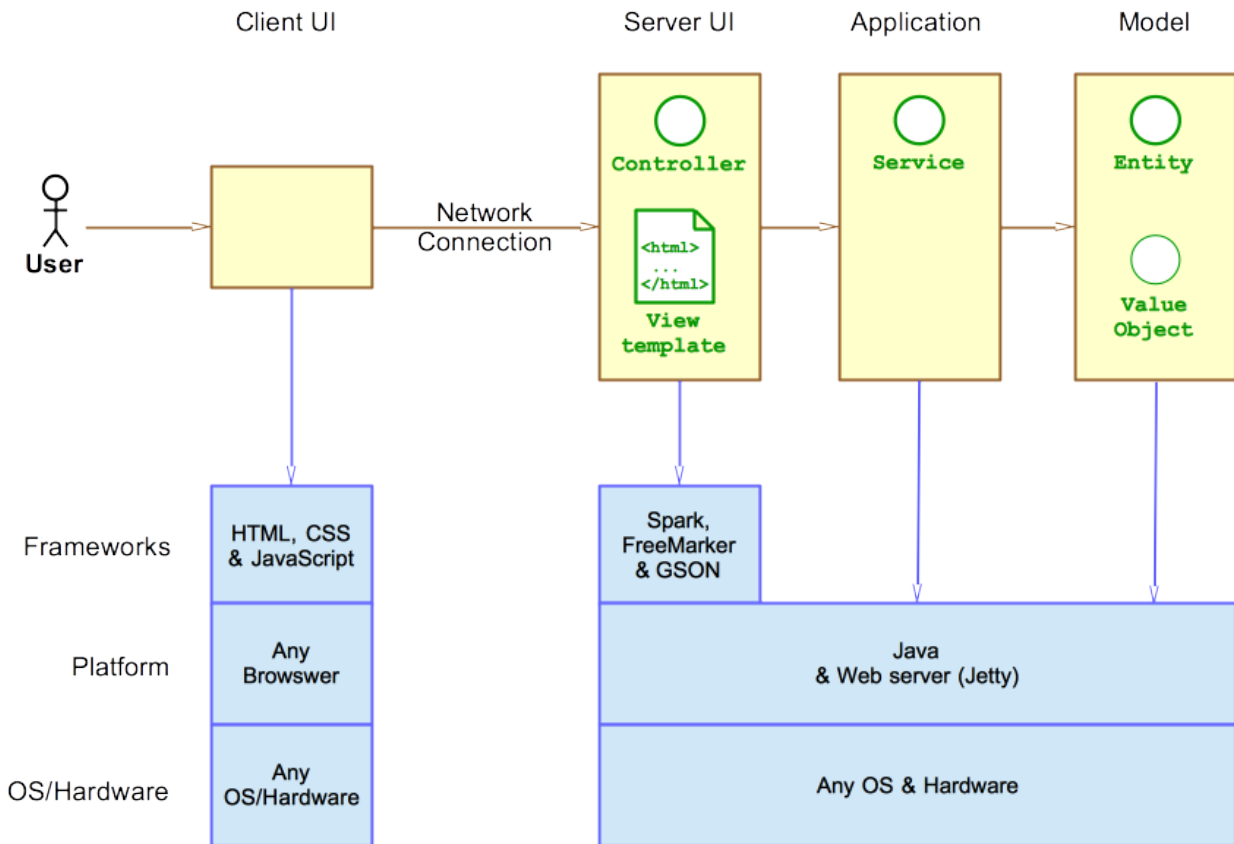


Figure 2: The Tiers & Layers of the Architecture

As a web application, the user interacts with the system using a browser. The client-side of the UI is composed of HTML pages with some minimal CSS for styling the page. There is also some JavaScript that has been provided to the team by the architect.

The server-side tiers include the UI Tier that is composed of UI Controllers and Views. Controllers are built using the Spark framework and View are built using the FreeMarker framework. The Application and Model tiers are built using plain-old Java objects (POJOs).

Details of the components within these tiers are supplied below

Overview of User Interface

This section describes the web interface flow; this is how the user views and interacts with the WebCheckers application.

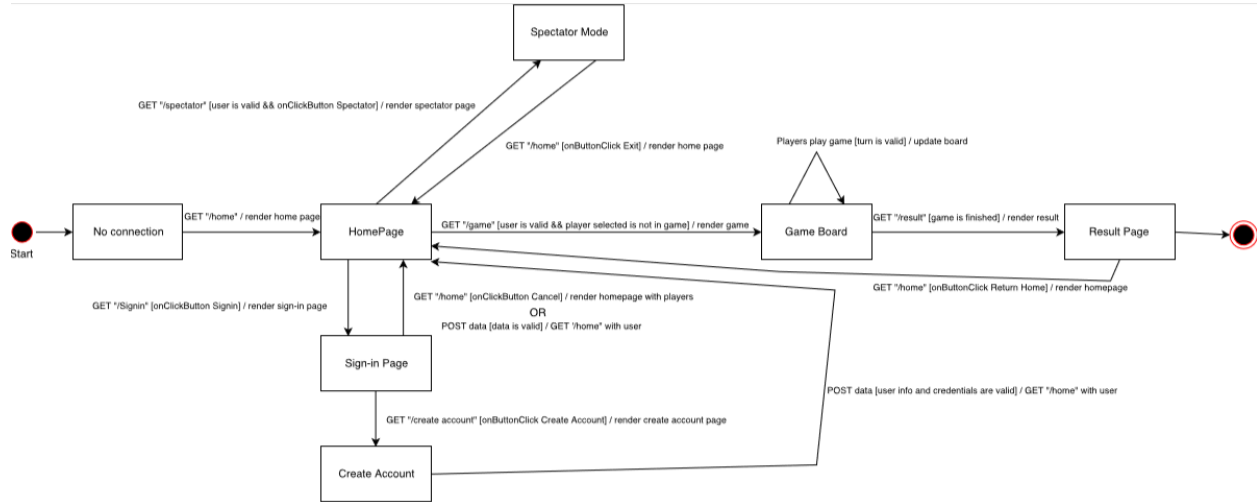


Figure 3: The WebCheckers Web Interface Statechart

The standard procedure for a user, is to first connect to the homepage. This renders the webapplication interface. From there the user can click on the Signin button and then signin page is rendered. If the user is new, they can click the create account, which will render the create account page. In either case the user will then be taken back to the homepage. The user can now either go into spectator mode, or play a game. If the user clicks the spectate button, they will be taken to the spectator page. When done they will be taken back to the homepage. If the user clicks the play game button, the user will be taken to the game board, where they can play a game. Once done they will be returned to the homepage.

UI Tier

The UI tier consists of several classes that function as the server side backend. The classes interface and provide hooks from the client side to the server side

Application Tier

The application Tier consists of the PlayerLobby and CenterGame class. The center game class manages an arraylist of available games and creates a lobby for a player using PlayerLobby. It is also used to start the game of checkers with startgame(). The player lobby manages the players waiting for a game.

Model Tier

The model tier for the application, consists of the various game pieces, parts and their functionalities. The foundation of the model tier is the game class. This class interfaces with the board and player classes. The board class in turn interfaces with the row, space, and piece classes. The player class interfaces with the move, moverule, turn, piece and position classes.

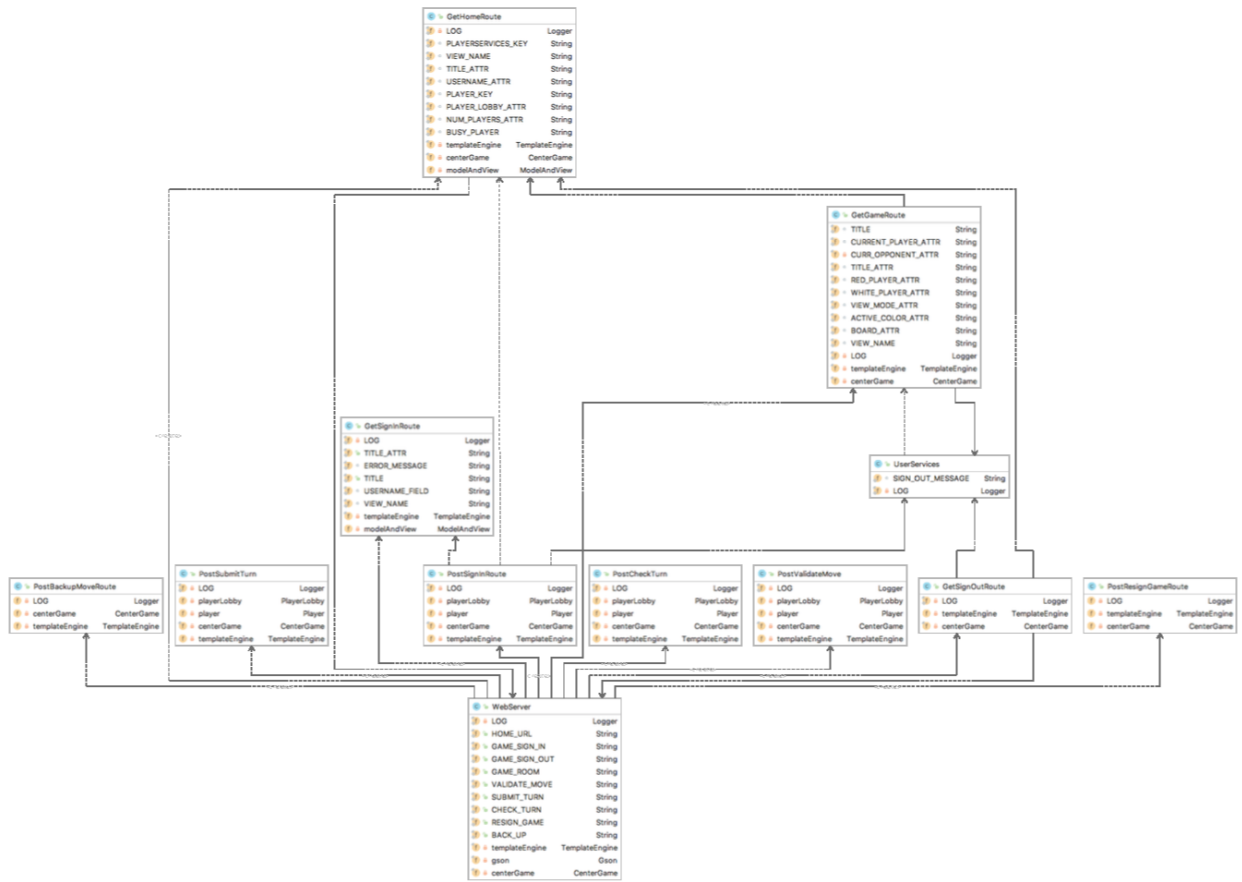


Figure 4: Class Diagram of UI Tier

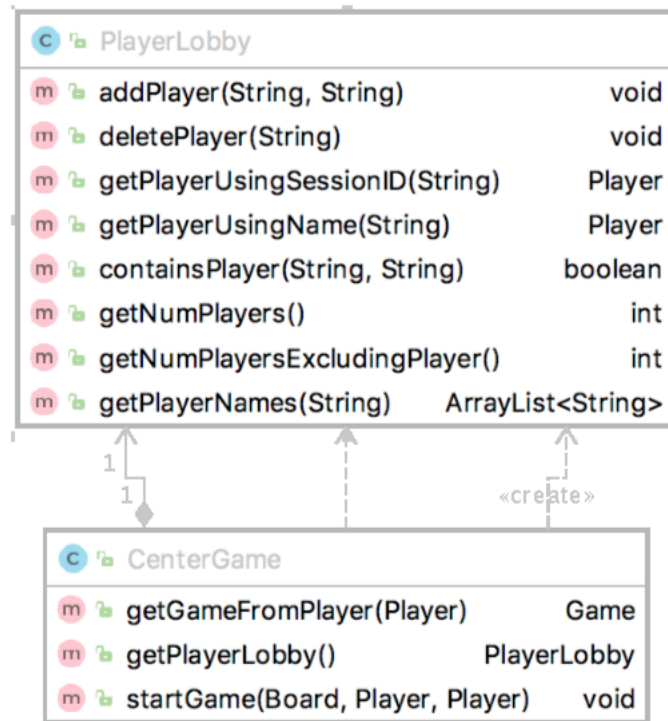


Figure 5: APPUML

Code Coverage

The code coverage for our WebCheckers Game is around 90%. We tried to do extensive testing for our every class and methods ensuring that everything is working as expected.

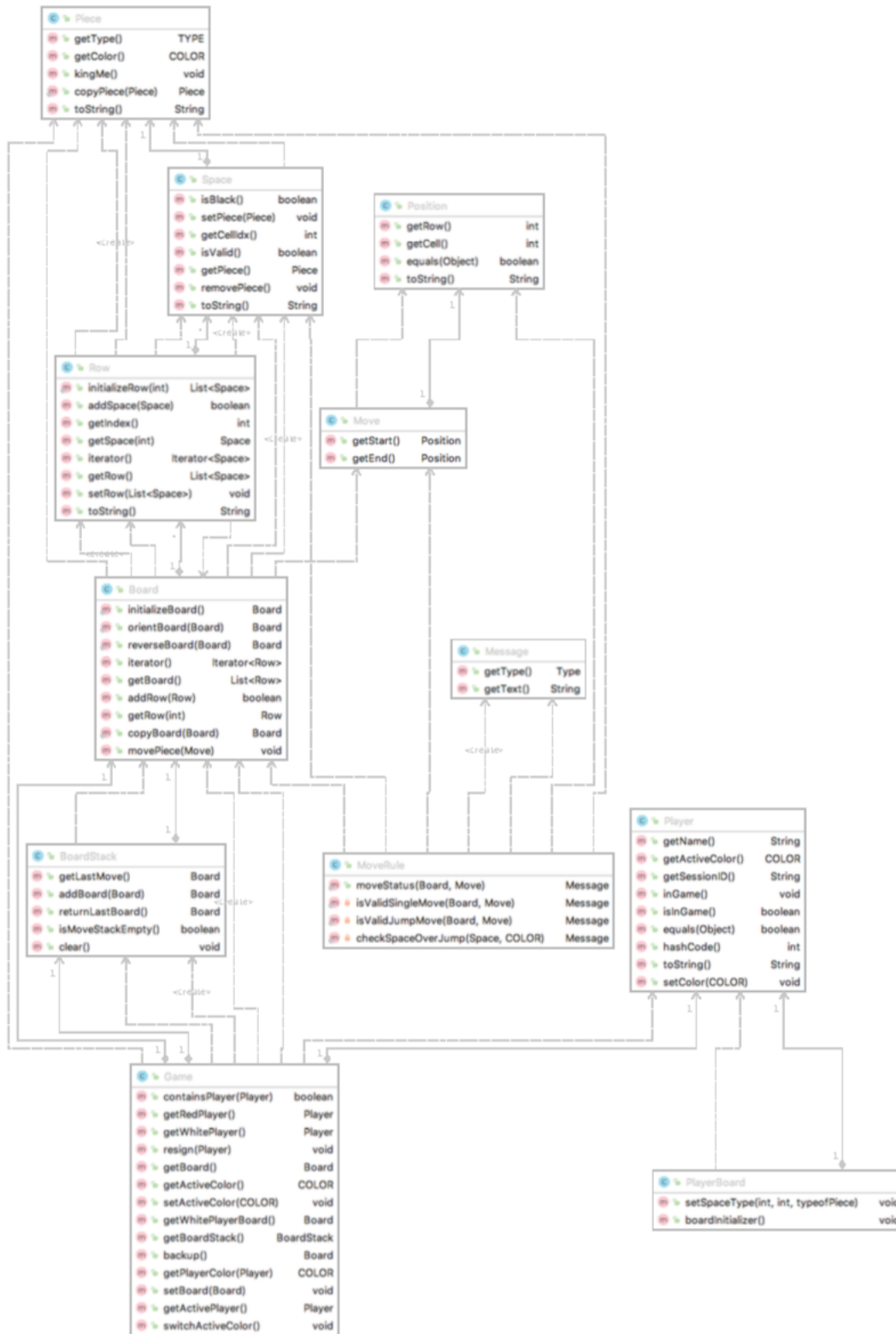
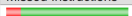
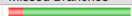




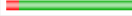
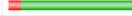


Figure 6: MUMLPng

Web Checkers a'la Spark/Java8

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.webcheckers.model		90%		90%	3	10	2	285	2	96	1	18
com.webcheckers.ui		93%		93%	10	58	23	260	37	41	1	13
com.webcheckers.appl		91%		91%	2	20	5	39	14	14	2	2
com.webcheckers		93%		93%	1	4	23	23	4	4	1	1
Total	2,950 of 3,122	92%	188 of 192	92%	16	252	53	607	57	155	5	34

Created with JaCoCo 0.8.0.201801022044

Figure 7: CodeCoverage