**Design Documentation**

* Purpose
* Requirements

-> Definition of MVP

-> MVP Features

* Application Architecture

-> Summary

* System Design

Authors:

Raseshwari Pulle

Spoorthy Annapareddy

**Purpose**

This documentation gives the general idea of the product being developed and the technologies, models being used to implement the product. It contains every detail that’s being added to the product in order to let a non-team member to understand what’s being implemented and why.

The Webcheckers project is a web application for people to play checkers online. This project allows a person to play a game of checkers according to the american rules. A person can simply sign-in with a username and select an opponent from the list of players available and have a game with that person.

**Requirements**

**Definition of MVP**

The MVP of this project is to :

* Have a player sign-in with a username before he/she begins the game.
* Let two players play a game of checkers on web based on the american rules.
* Allow a player to come out of a game whenever he/she wants by signing out.

**MVP Features**

* Sign-in

The player sign-in is one feature of MVP where the player is able to sign-in with a username of his/her choice and that username is in use and not available for another person to use till the initial player ends his/her game or signs out of the application.

* Game View

The game view feature displays the 8x8 checkers board with the initial setup of pieces.

* Resignation

A player can sign-out from the application whenever he/she wants. Once the player signs out the username used him/her is released and anyone can use that name again. When the player signs-out when a game is in progress the game gets cancelled.

**Application Architecture**

**Summary**

We are using Spark framework for our Web Checkers game implementation. Spark is a Java-based, web micro-framework which handles HTTP requests and delegates HTML generation to a template Engine. We are using FreeMarker template engine for this purpose. Some benefits of using FreeMarker is that it supports JSP tags, templates can be nested in it in run-time and it also supports JSON.

The architecture of our game is divided in three layers mainly as follows:

1. Application - This layer has the class which maintains the state of the game.
2. Model - This layer has classes which define the behaviour of the game (or the rules for the game) along with all the components used for playing the game like discs, board, etc.
3. UI - This layer has all the classes which govern the routing of the view for certain set of actions performed while playing the game.

Following is the list of classes implemented for current Sprint Release-

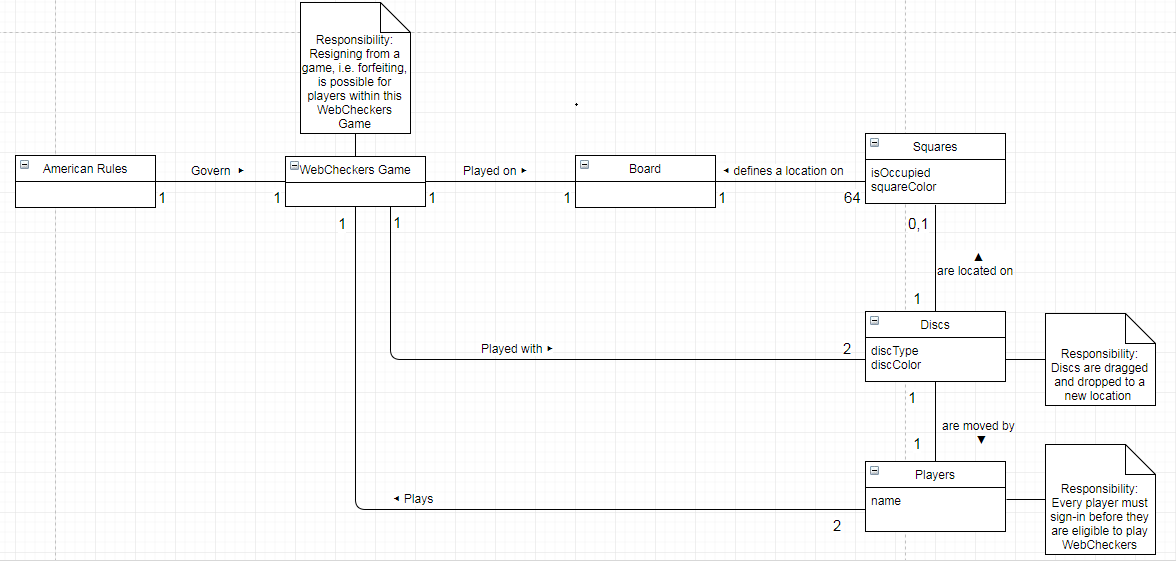
1. Application layer - CheckersCenter.java
2. Model layer -
   1. CheckersGame.java
   2. Board.java
   3. Piece.java
   4. Space.java
   5. Row.java
   6. Position.java

3. UI layer-

1. GetGameMenuRoute.java
2. GetGameRoute.java
3. GetSigninRoute.java
4. HomeController.java
5. JsonUtils.java
6. PostNameRoute.java
7. PostOpponentRoute.java
8. WebServer.java

**System Design**

We have implemented the domain model for our Web Checkers game implementation. Purpose of designing a domain model is to describe the abstractions in our proposed applications and to give a clear understanding of the system.



We have implemented the StateChart Diagram to demonstrate the MVPs implemented for the Sprint 1. It covers state of the system while the player is signed in and begins to play the game.

