# PROJECT Design Documentation

> *The following template provides the headings for your Design Documentation. As you edit each section make sure you remove these commentary 'blockquotes'; the lines that start with a > character and appear in the generated PDF in italics.*

## Team Information

- Team name: Checkm8
- Team members
  - Spencer Josi
  - Casper Loveless
  - Ryan Madulka
  - Alan Read

## Executive Summary

Our web-based checkers application will provide an interactive user experience as players can play against other online users. It will incorporate sign-in functionality as each user can specify who they want to play against, directing both users to a game page presenting a GUI with the game board. The checkers game will also have drag-and-drop piece capabilities as a user can submit their moves when they drop their piece in a valid position. Not only will users be able to play against other online users, but they can compete against an AI or can record games to replay move-by-move.

### Purpose

We want to develop a web application for all users to play checkers against other live opponents, abiding by the American checkers rules and providing a fully functional server-client connection.

### Glossary and Acronyms

> *Provide a table of terms and acronyms.*

| Term | Definition |
|---|---|
| VO | Value Object |

## Requirements

This section describes the features of the application.

> *In this section you do not need to be exhaustive and list every story. Focus on top-level features from the Vision document and maybe Epics and critical Stories.*

### Definition of MVP

The Minimum Value Product expresses the most essential requirements that must be developed and included within the product in order for it to be satisfactory. For the checkers web-app, users must be able to sign in and start a checkers game that abides by the American rules, playing on a fully functional GUI board.

### MVP Features__

A user must be able to sign in and out of their account, and if they remain logged in they can initiate a game with a selected online user. Both users will be transferred to a game board screen consisting of all of the checker pieces, along with buttons giving them the option to resign the game the game or submit their valid move. When done playing, users can sign out of their account.

### Roadmap of Enhancements

We have already completed sign-in functionality and the basic structure of the board, so our current sprint plan is to complete valid piece movement, develop game structure as users must be able to submit their turn or resign, and also to create sign-out functionality. We are expecting to finish the MVP within this sprint, as sprint 3 is devoted to completing our enhancements where we want to implement an AI and a replay mode to record and watch previous games.

## Application Domain

This section describes the application domain.

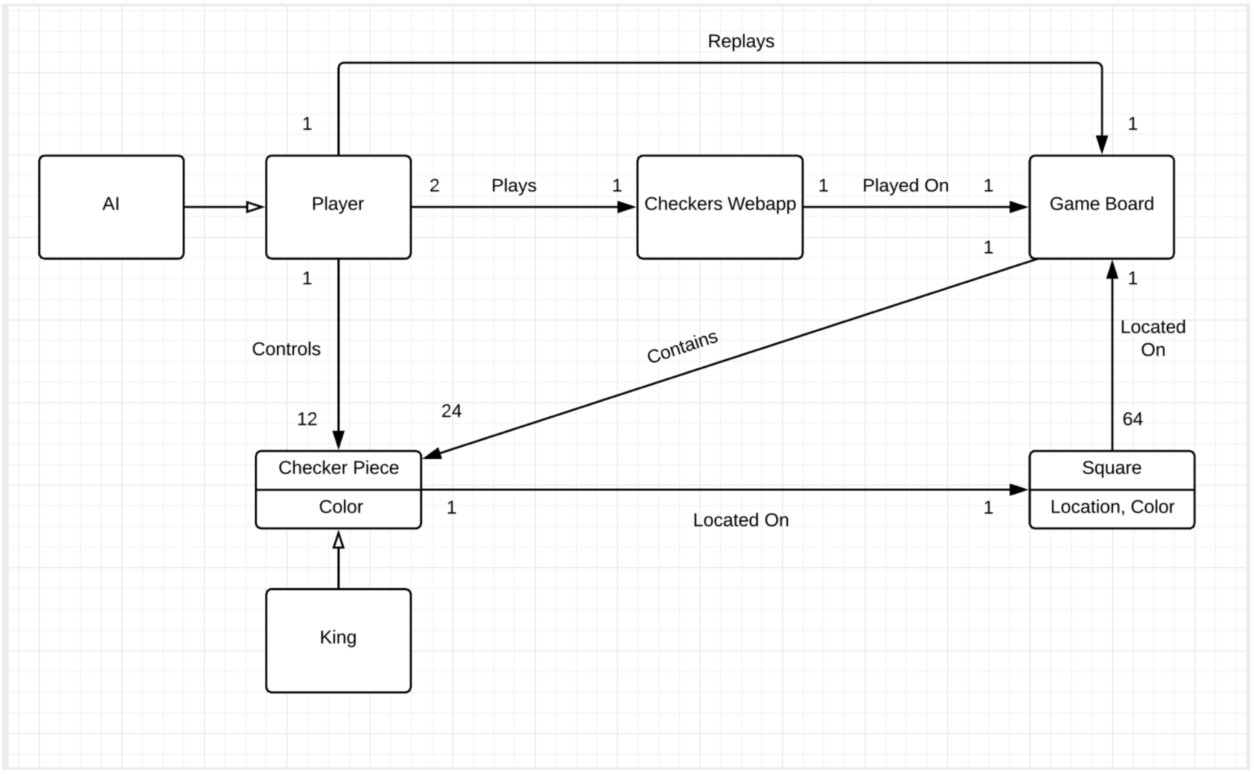


Figure 1: The WebCheckers Domain Model

The Checkers WebApp is played by two Players on a single checkers GameBoard. The GameBoard contains 64 Squares, with each having a Location and a Color(Dark/Light). 24 Pieces are initially placed on the GameBoard, and each Player takes control of 12. One Piece can occupy one Square. Each Piece has a Color(Red/White) and can either be a Single Piece or a King Piece. A Player can either be a human Player or an AI Player, and a Player has the ability to replay a Game.

## Architecture and Design

This section describes the application architecture.

### Summary

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

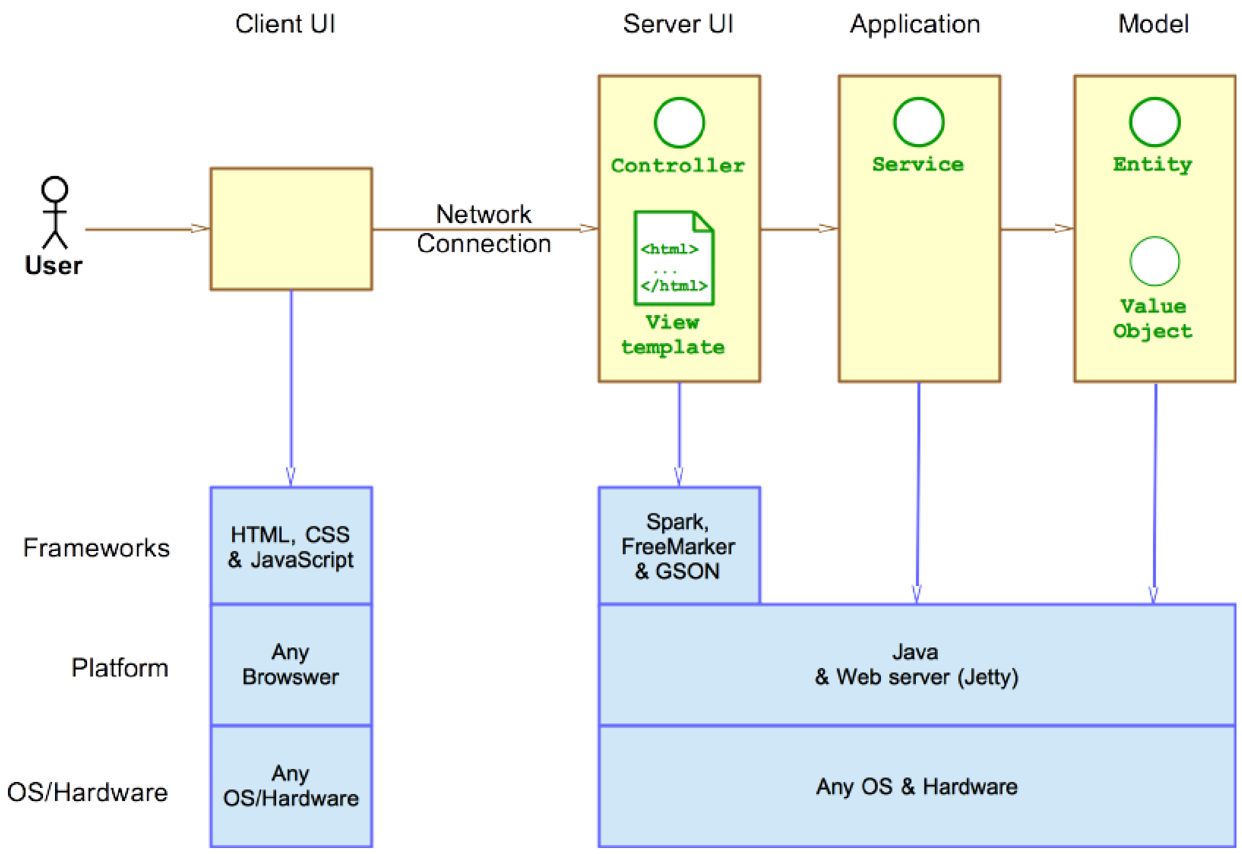


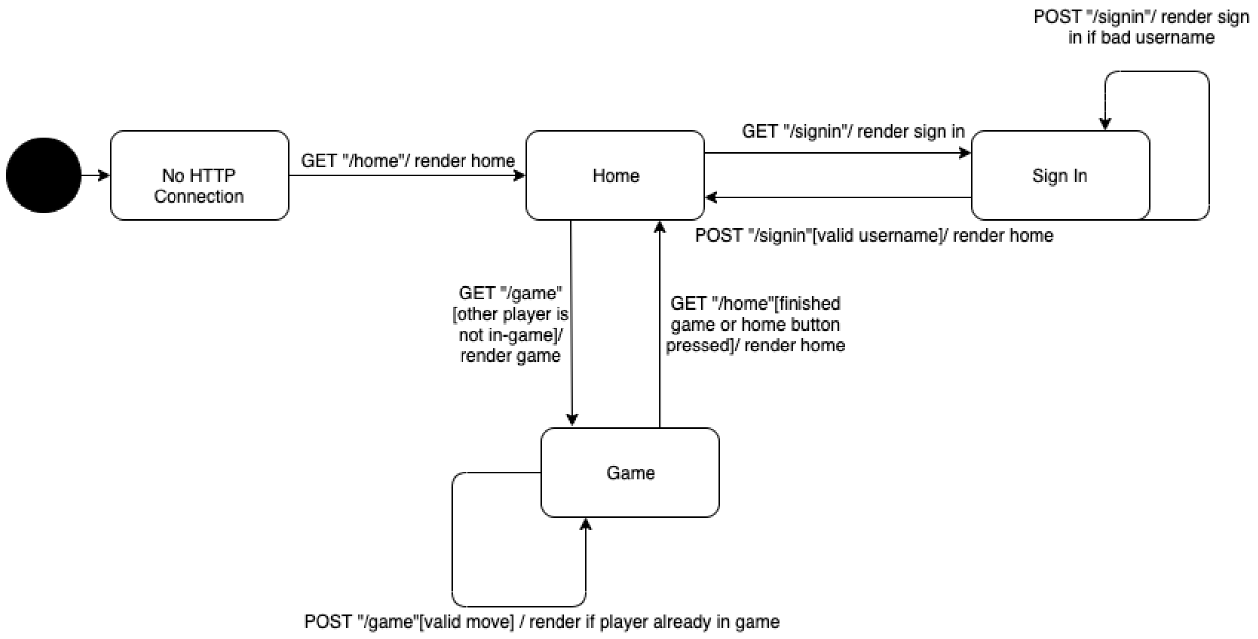
Figure 2: The Tiers & Layers of the Architecture

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.


POST "/signin"/ render sign in if bad username
No HTTP Connection
GET "/home"/ render home
Home
GET "/signin"/ render sign in
Sign In
POST "/signin"[valid username]/ render home
GET "/game" [other player is not in-game]/ render game
GET "/home"[finished game or home button pressed]/ render home
Game
POST "/game"[valid move] / render if player already in game

When the user first connects to the web application, they are sent to the home page. By clicking on the sign-in button, on the top left of the screen, the user is then sent to the sign-in page in which they can then enter a username. Assuming the user inputs a valid username and they click on the sign-in button, they will be sent back to the home page. If the player enters an invalid username, the sign-in page will reload. Once they are back on the home page and the user chooses to play an online user, they will both be sent to the game screen. On the game page, each player takes turns making moves and the page updates after each move is made. Once the game is over, the player will be sent back to the homepage.

### UI Tier

*Provide a summary of the Server-side UI tier of your architecture. Describe the types of components in the tier and describe their responsibilities. This should be a narrative description, i.e. it has a flow or "story line" that the reader can follow.*

*At appropriate places as part of this narrative provide one or more static models (UML class structure or object diagrams) with some details such as critical attributes and methods.*

*You must also provide any dynamic models, such as statechart and sequence diagrams, as is relevant to a particular aspect of the design that you are describing. For example, in WebCheckers you might create a sequence diagram of the `POST /validateMove` HTTP request processing or you might show a statechart diagram if the Game component uses a state machine to manage the game.*

*If a dynamic model, such as a statechart describes a feature that is not mostly in this tier and cuts across multiple tiers, you can consider placing the narrative description of that feature in a* *separate section for describing significant features. Place this after you describe the design of the three tiers.*

### Application Tier

*Provide a summary of the Application tier of your architecture. This section will follow the same instructions that are given for the UI Tier above.*

### Model Tier

*Provide a summary of the Application tier of your architecture. This section will follow the same instructions that are given for the UI Tier above.*

### Design Improvements

*Discuss design improvements that you would make if the project were to continue. These improvement should be based on your direct analysis of where there are problems in the code base which could be addressed with design changes, and describe those suggested design improvements. After completion of the Code metrics exercise, you will also discuss the resutling metric measurements. Indicate the hot spots the metrics identified in your code base, and your suggested design improvements to address those hot spots.*

## Testing

*This section will provide information about the testing performed and the results of the testing.*

### Acceptance Testing

*Report on the number of user stories that have passed all their acceptance criteria tests, the number that have some acceptance criteria tests failing, and the number of user stories that have not had any testing yet. Highlight the issues found during acceptance testing and if there are any concerns.*

### Unit Testing and Code Coverage

*Discuss your unit testing strategy. Report on the code coverage achieved from unit testing of the code base. Discuss the team's coverage targets, why you selected those values, and how well your code coverage met your targets. If there are any anomalies, discuss those.*