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: Something-Cool
- Team members
  - Kevin Hlavaty
  - Corey Urbanke
  - Russell Lee
  - Ray Lorenzo
  - Danny Gardner

Executive Summary

This project allows players to play local checkers games using any common web-browser. Players can start playing by signing in using their name, and then initiate a game by selecting the name of other signed in players.

Purpose

This project was created in order to allow people to have fun playing checkers, while also teaching some people how to play checkers.

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

> *Provide a simple description of the Minimum Viable Product.*

MVP Features

> *Provide a list of top-level Epics and/or Stories of the MVP.*

Roadmap of Enhancements

> *Provide a list of top-level features in the order you plan to consider them.*

Application Domain

This section describes the application domain.

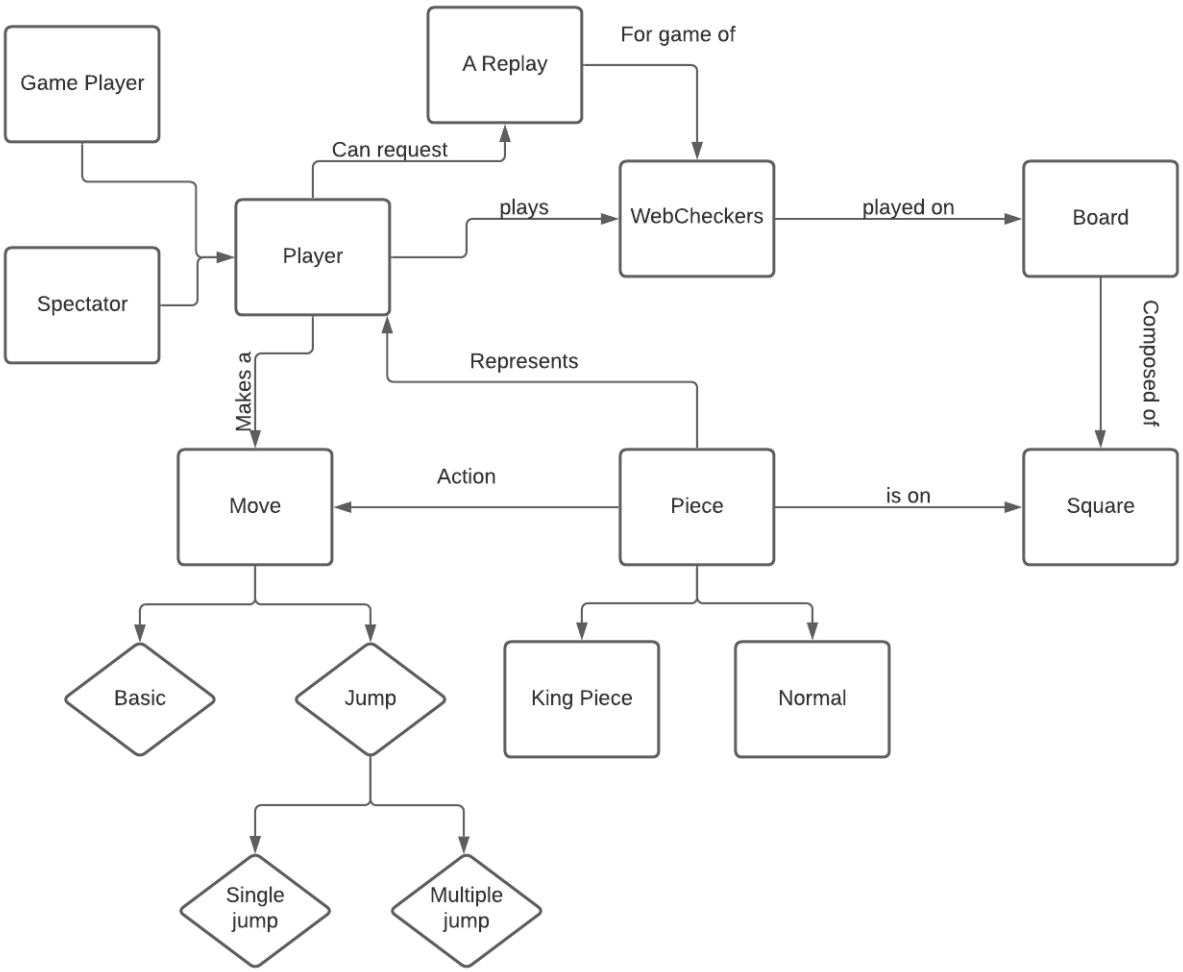

Figure 1: The WebCheckers Domain Model

- A signed-in Player is able to join a match with another signed-in Player who is not already in a game.
- The two players play a WebCheckers game on a Board until one Player wins and the other loses.
- Pieces on the Board can be moved by a Player when it is their turn. There are different types of moves a player can make, either to a valid space or in attempt to capture an opponent Piece.
- Pieces are deemed King Pieces when they reach the last Row of the opponent's side.
- Signed-in Players can spectate a match that two other signed-in Players are in.
- Players who have completed a WebCheckers match can request a replay for that given game.

Architecture and Design

This section describes the application architecture.

Summary

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

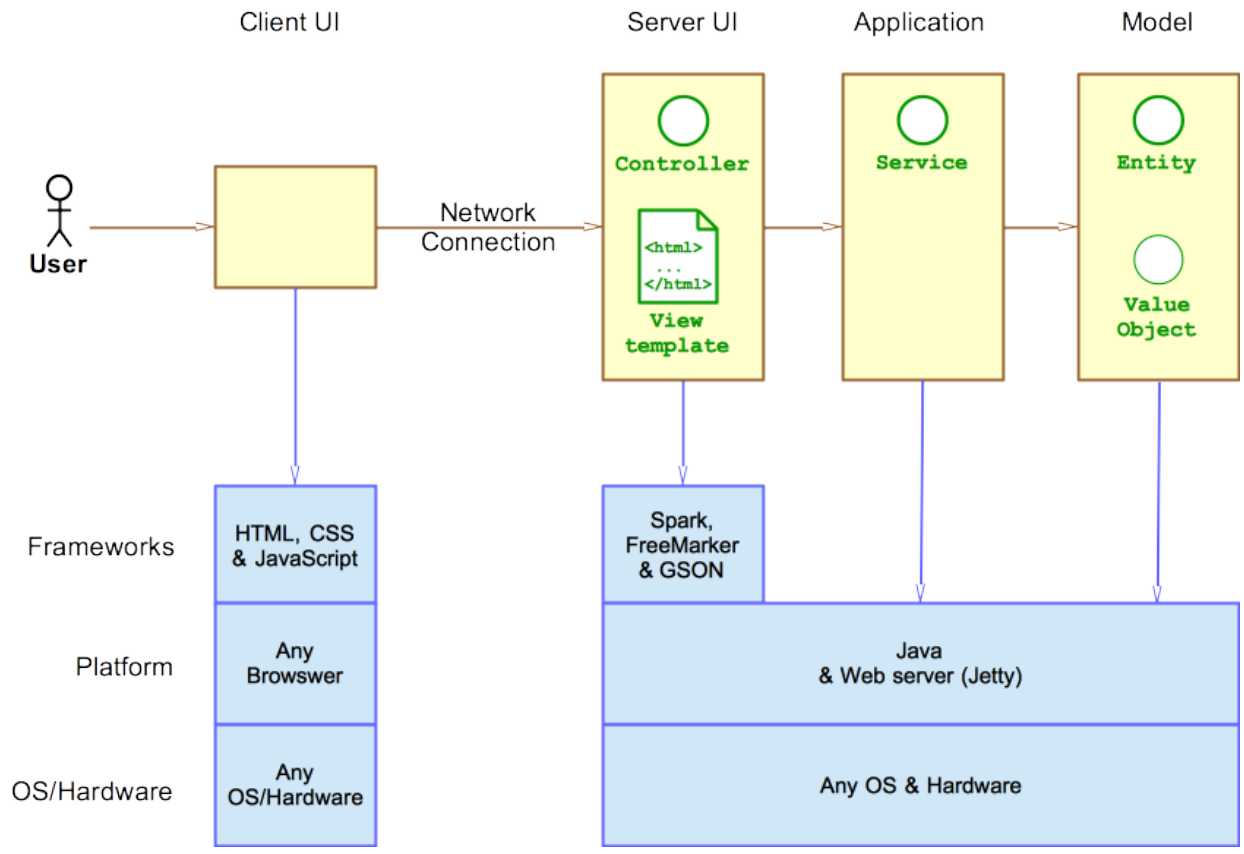

Figure 2: The Tiers & Layers of the Architecture

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.

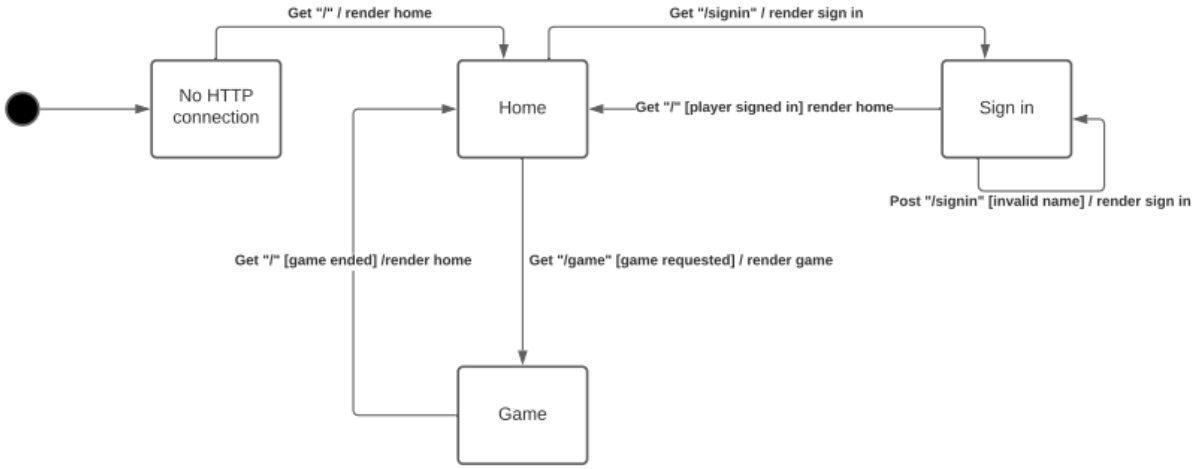

Figure 3: The WebCheckers Web Interface Statechart

Once a connection is established, the user is brought to the Home page where they can click a sign in button. To sign in, the user is brought to the Sign In page and, once signed in, brought back to the Home page. If a user enters aj invalid name, they are brought back to the sign in page until a valid name is entered. From the Home page, the user can enter a game with another player, bringing them to the Game page. Finally, when the game ends, the user is brought back to the Home page where they are still signed in, able to enter another game.

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 com.webcheckers.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.