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: TEAM A
- Team members
  - Jaron Cummings
  - Soumya Dayal
  - Chris Fitzgerald
  - Dylan Mulligan

Executive Summary

This program codes and builds a functioning game of WebCheckers. This application must allow multiple players to sign in and play a game of WebCheckers with another player who is currently signed-in. Any player must be able to leave the game whenever they want, irrespective of the completion of the game. When a player leaves, their opponent must be declared winner directly.

The game user interface (UI) will support a game experience using drag-and-drop browser capabilities for making moves. Any move made will be shown on the screen of each user. Each move must be validated according to the American Standard Checkers rules. Any unvalidated or wrong move must not be allowed.

Beyond this minimal set of features, we have vision of enhancing this application with some additional features. This includes the AI Mode, where a player must be allowed to play with the computer itself.

Purpose

> *This application builds a working WebCheckers where players who have signed in can play with each other on the web.*

Glossary and Acronyms

> *Provide a table of terms and acronyms.*

Term	Definition
VO	Value Object
UI	User Interface
AI	Artificial Intelligence
MVP	Minimum Viable Product

Requirements

This section describes the features of the application.

> *Multiple players must be able to sign-in to start playing with each other. As a player, they must be able to leave the game when they want to, and have their opponent be the winner. As a player, they must be able to make valid moves to play the game of WebCheckers.*

Definition of MVP

> *The MVP of the application will have players being able to sign in and start playing the game with other signed-in players. The players will make valid moves on the Board in order to reach the other end of the Board and win the game. The valid moves includes not being able to move to a white space, or making moves not defined in the American Checkers rules. The players will be able to leave the game, even before completion, therefore making their opponent automatic winner.*

MVP Features

> *As a Player I want to sign-in so that I can play a game of checkers.*
> *As a Player I want to start a game so that I can play checkers with an opponent.*
> *As a player I want to make valid moves in a game of checkers so that I can play a game of checkers.*
> *As a player, I want to follow the rules so I can capture 1 or more pieces of my opponent.*
> *As a Player I want to resign so that I can end the game if a loss is imminent or I have to leave.*

Roadmap of Enhancements

> *AI Player - The player signed-in must be able to play with the computer following the same rules as before, if they wish to.*
> *Replay Mode - The game played with the player or the computer must be stored so they can be viewed at a later date.*

Application Domain

This section describes the application domain.

> *As shown in the domain model, the player signs in to the game, and performs moves according to the American Checkers rules in order to move the pieces. The pieces move on the Board containing 64 squares, which is used to play the Game. The player signs out of the Game when they want to leave.*

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.

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.

> *The user will first get the screen to enter their name and sign in as they reach the Checkers game. Once they sign in successfully, the HTTP connection is set. The user then waits for another*

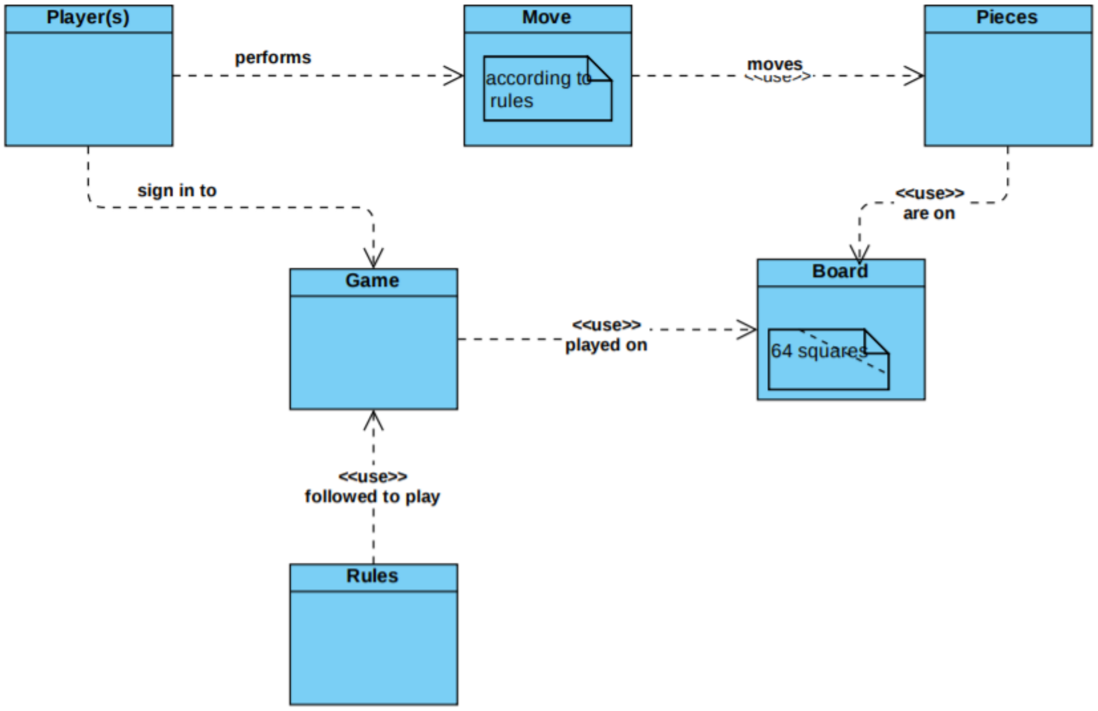


Figure 1: The WebCheckers Domain Model

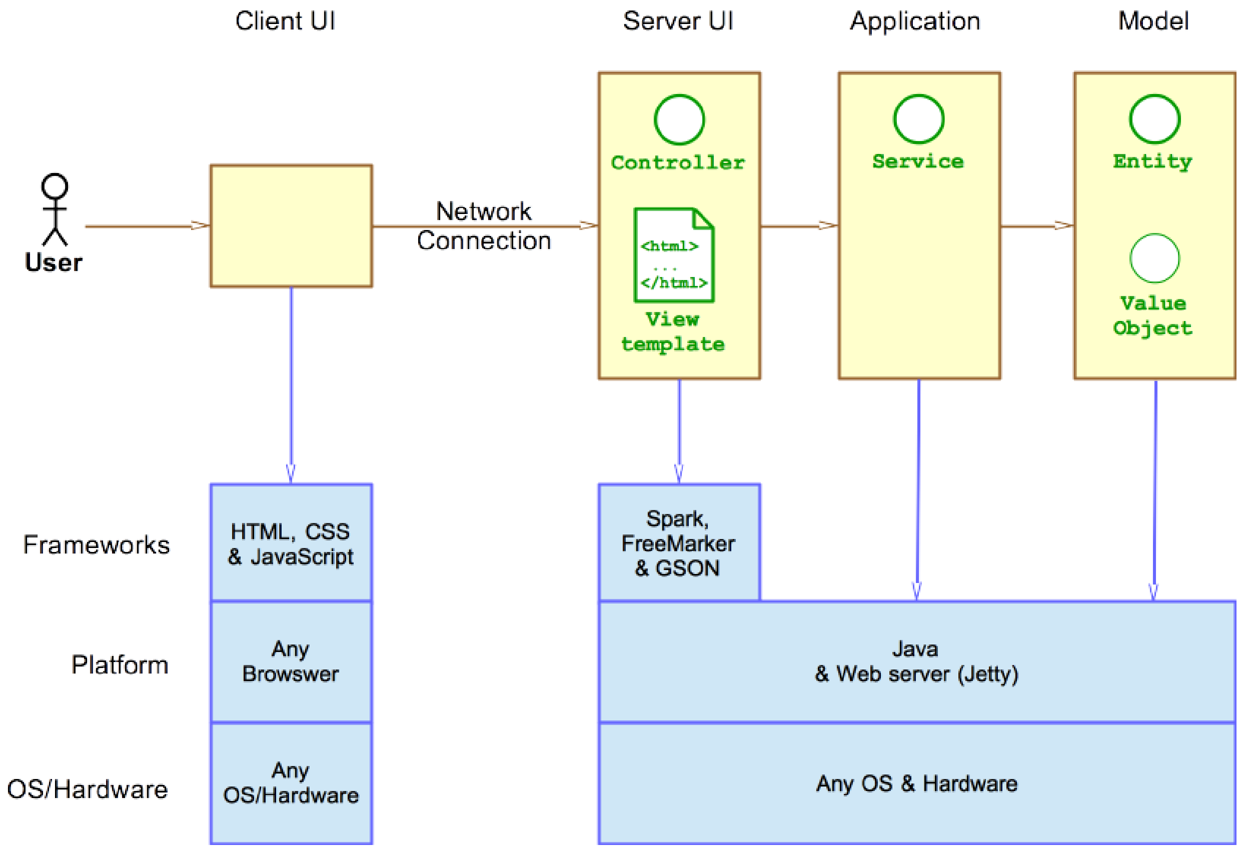


Figure 2: The Tiers & Layers of the Architecture

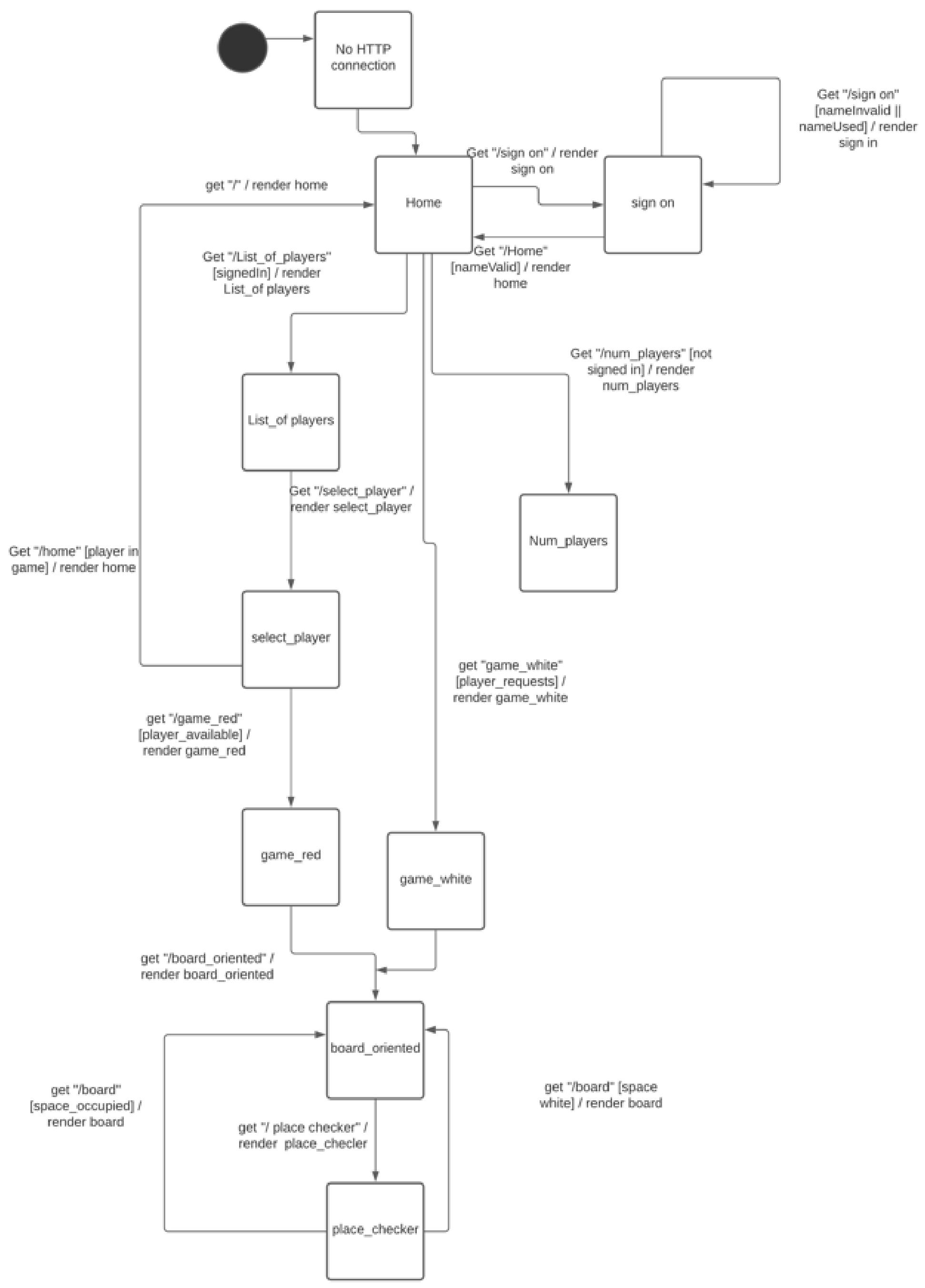

No HTTP connection
Get "/sign on" [nameInvalid || nameUsed] / render sign in
Get "/sign on" / render sign on
get "/" / render home
Home
sign on
Get "/List_of_players" [signedIn] / render List_of players
Get "/Home" [nameValid] / render home
Get "/num_players" [not signed in] / render num_players
List_of players
Get "/select_player" / render select_player
Num_players
Get "/home" [player in game] / render home
select_player
get "game_white" [player_requests] / render game_white
get "/game_red" [player_available] / render game_red
game_red
game_white
get "/board_oriented" / render board_oriented
board_oriented
get "/board" [space_occupied] / render board
get "/board" [space white] / render board
get "/ place checker" / render place_checler
place_checker

Figure 3: The WebCheckers Web Interface Statechart

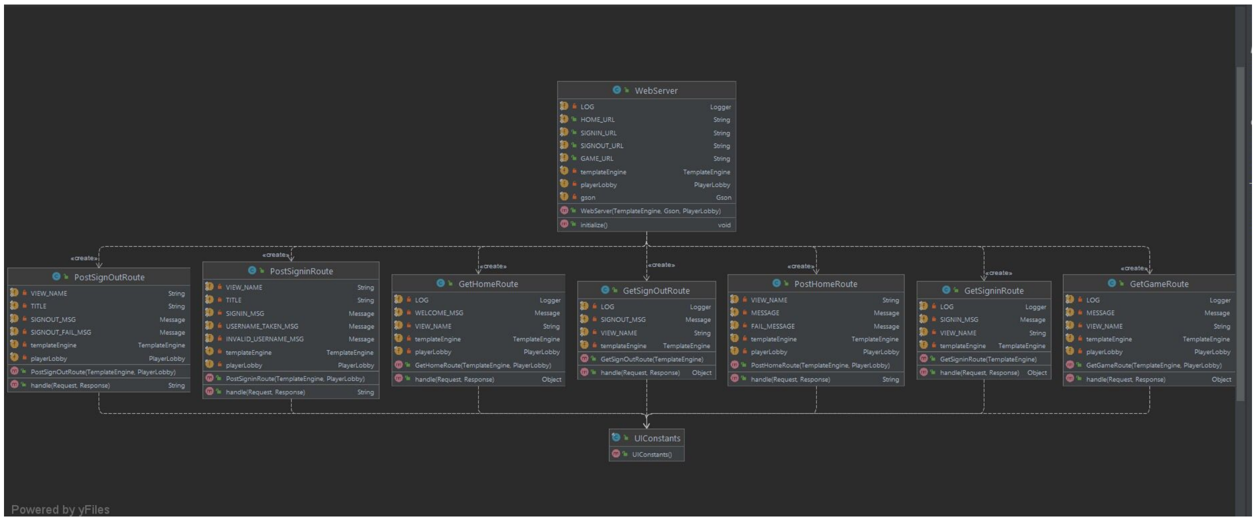
player to sign in before starting a game to be played. The user sees the full board with their color's side towards them. They make a move, which if valid, is allowed on the Board for all players to see. This continues until we have a winner and the game ends or one of the players quits and the game ends.

UI Tier

The UI Tier starts with a GetSignInRoute class which includes sending a request to a server and getting its response. When successful, this allows the player(s) to sign in to the WebCheckers game to play or wait for other players in the Home screen. If a situation arises where player has entered a username that is already taken, or uses special characters in the username, the PostSignInRoute helps display appropriate message from the server. This class is the {POST /signin} route handler.The UI Controller to get the Home page is GetHomeRoute class, It is helped by PostHomeRoute class as that controls the Home page of the game. The Home page contains the names of the players who are currently signed in and if there are two players, the game can begin.

The Game screen is brought by the GetGameRoute class. This UI Controller is to get the game page. Every player can see the Game page from their piece color's perspective. If the HTTP request and response is successful, the GetGameRoute class displays appropriate message for the players.

After playing, the player can choose to sign out which happens using the GetSignOutRoute class, which is the UI Controller to get the signout page. It get the appropriate message from the server to help player sign out, and is able to do so for them. If the user decides to sign out, the appropriate message is sent by the server using the PostSignOutRoute class. The diagram below shows the UML representation of the package ui.



WebServer
LOG Logger
HOME_URL String
SIGNIN_URL String
SIGNOUT_URL String
GAME_URL String
templateEngine TemplateEngine
playerLobby PlayerLobby
gson Gson
WebServer(TemplateEngine, Gson, PlayerLobby)
initialize() void
«create»
PostSignOutRoute
VIEW_NAME String
TITLE String
SIGNOUT_MSG Message
SIGNOUT_FAIL_MSG Message
templateEngine TemplateEngine
playerLobby PlayerLobby
PostSignOutRoute(TemplateEngine, PlayerLobby)
handle(Request, Response) String
PostSigninRoute
VIEW_NAME String
TITLE String
SIGNIN_MSG Message
USERNAME_TAKEN_MSG Message
INVALID_USERNAME_MSG Message
templateEngine TemplateEngine
playerLobby PlayerLobby
PostSigninRoute(TemplateEngine, PlayerLobby)
handle(Request, Response) String
GetHomeRoute
LOG Logger
WELCOME_MSG Message
VIEW_NAME String
templateEngine TemplateEngine
playerLobby PlayerLobby
GetHomeRoute(TemplateEngine, PlayerLobby)
handle(Request, Response) Object
GetSignOutRoute
LOG Logger
SIGNOUT_MSG Message
VIEW_NAME String
templateEngine TemplateEngine
GetSignOutRoute(TemplateEngine)
handle(Request, Response) Object
PostHomeRoute
VIEW_NAME String
MESSAGE Message
FAIL_MESSAGE Message
templateEngine TemplateEngine
playerLobby PlayerLobby
PostHomeRoute(TemplateEngine, PlayerLobby)
handle(Request, Response) String
GetSigninRoute
LOG Logger
SIGNIN_MSG Message
VIEW_NAME String
templateEngine TemplateEngine
GetSigninRoute(TemplateEngine)
handle(Request, Response) Object
GetGameRoute
LOG Logger
MESSAGE Message
VIEW_NAME String
templateEngine TemplateEngine
playerLobby PlayerLobby
GetGameRoute(TemplateEngine, PlayerLobby)
handle(Request, Response) Object
UIConstants
UIConstants()
Powered by yFiles

Figure 4: img.png

TODO in coming sprints: 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.