PROJECT Design Documentation

Team Information

- Team name: Team B++
- Team members
  - Ryan Kohn
  - Christopher Johns
  - Ryan Floersch
  - Stone Warren
  - Steven McLeod

Executive Summary

Web Checkers is a simple web game for playing checkers that Team B++ has undertaken under the direction of our SWEN 261 professor.

Purpose

Team B++'s purpose with this project is to create an easy to use way for users to play a game of checkers online. Our most important user groups are our professor and our TA, who will be the main users of our game and will expect to have a completely functional, error-free experience.

Glossary and Acronyms

Term	Definition

Requirements

- The user will be able to sign-in to the player lobby and sign-out when done playing.
- The user will be able to start and play checkers games with other players.
- The user will be able to follow the rules of Checkers to move pieces and play the game.
- The user will be able to enter games in alternate modes, including spectator mode and replay mode.

Definition of MVP

For our minimal viable product, the first thing we need is a functional player lobby that allows players to sign in, out, and start games with unoccupied players. Secondly, we need to complete all in-game functionality such that both players are able to play a game of checkers while following the rules, moving and promoting pieces. Finally, we need to be able to handle games ending through win conditions or player resignation and direct players appropriately.

MVP Features

- Player Sign In/Out - Players will be able to sign in and out of the player lobby with a unique username.
- Start Game - Players will be able to start a game with other unoccupied players.
- Play Turn - Players will be able to make legal moves during their turns, and the game will handle alternating turns as players submit their moves.
- Game Ending - Game will recognize and handle win conditions and player resignation.
- Piece Movement - Pieces will be movable according to checkers rules, including handling simple movement, jump moves, multiple jump moves, and king movement.
- Promotion - Pieces will promote themselves to kings when they move to the opposite side of the board from where they started.

Roadmap of Enhancements

- Spectate Game - Players will be able to join and leave ongoing games as a spectator.
- Game Replays - Finished games will be saved and players can watch replays of them.

Application Domain

This section describes the application domain.

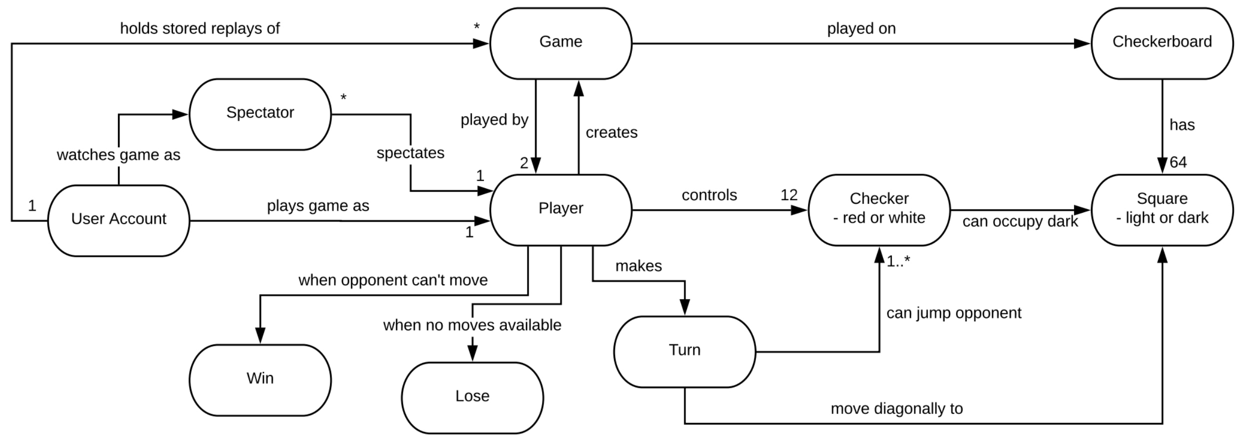


Figure 1: The WebCheckers Domain Model

Users sign in to the application and create a user account. From there, they start games with other users and become players in those games. Each game has an 8x8 board made up of light and dark squares and pieces for each player. Players take turns moving the pieces, and the game ends when the win/loss conditions are met. Users can also spectate or watch replays of games rather than play in them.

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 be directed to the home page first. After the user decides on a username, they will then be directed to a home page that includes online players.

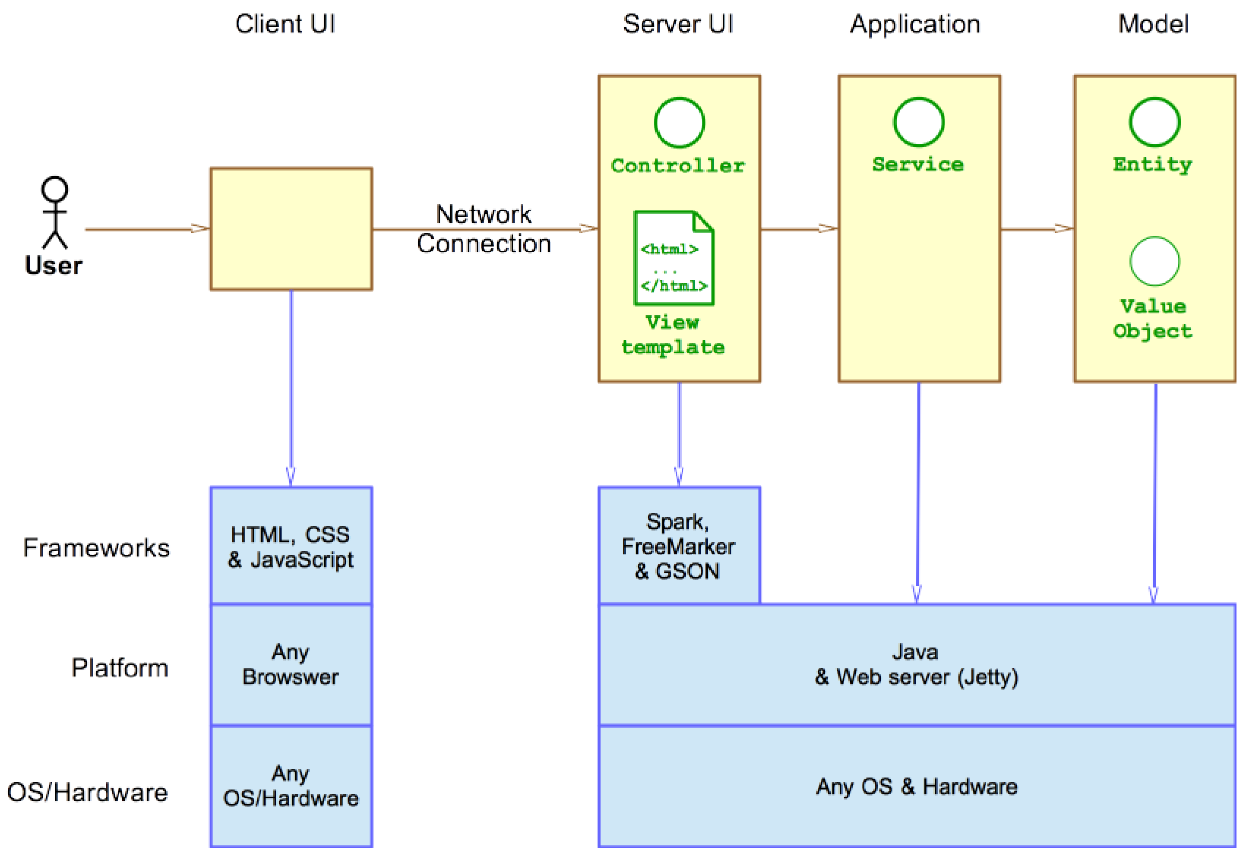
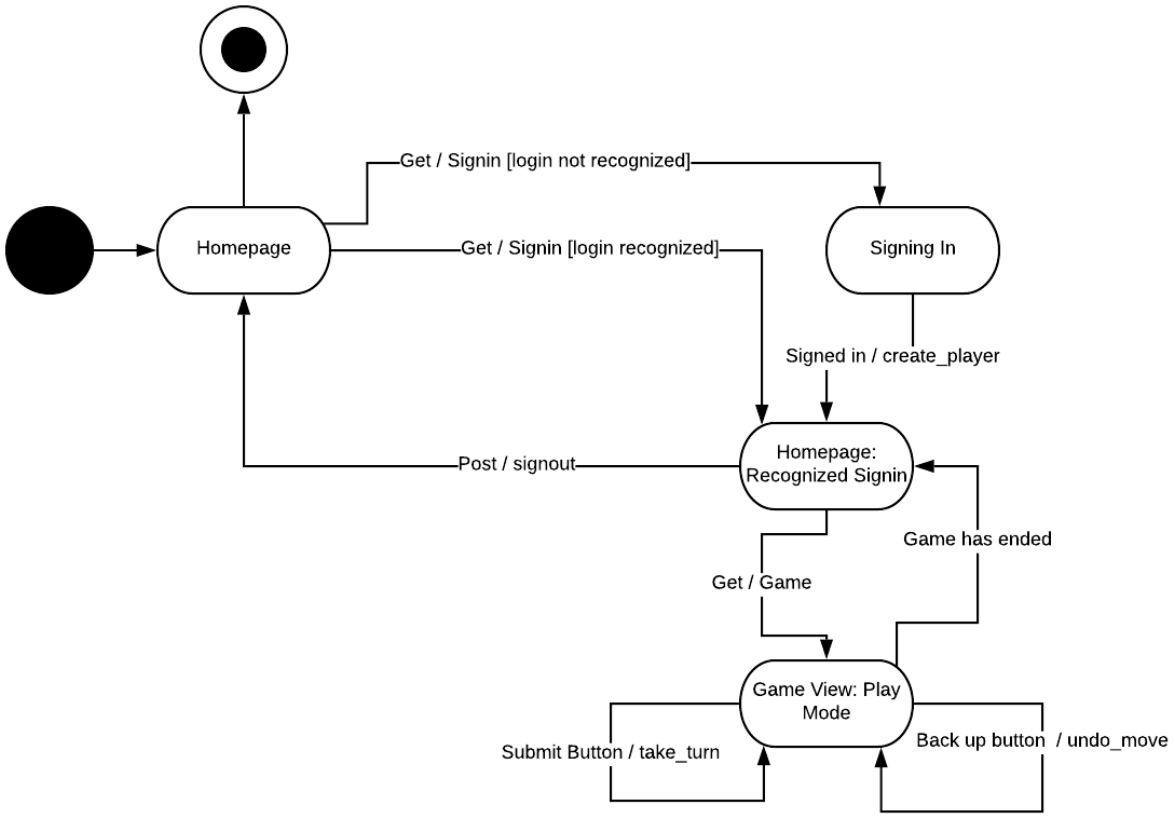


Figure 2: The Tiers & Layers of the Architecture



Homepage
Get / Signin [login not recognized]
Get / Signin [login recognized]
Signing In
Signed in / create_player
Post / signout
Homepage: Recognized Signin
Get / Game
Game has ended
Game View: Play Mode
Submit Button / take_turn
Back up button / undo_move

Figure 3: The WebCheckers Web Interface Statechart

If the user wants to start a game with someone, they can click on the players name and if that player is unoccupied both players will be redirected to a game page.

Once the game has ended, the players are redirected back to the signed in home page.

UI Tier

Our UI tier classes consist of different Get and Post Route classes, including GetGameRoute, GetHomeRoute, GetSignInRoute, PostHomeRoute, PostSignInRoute, and PostSignOutRoute. These Route classes handle directing users between the home page, the signed-in homepage, and the game page. They also give the template engine information on what the webpage should have/look like.

We also have our WebServer class that initializes the different routes, as well as Space and Row classes that make up the gameboard interface.

Application Tier

The application tier contains our PlayerLobby class, which keeps track of all players and games, handles sign-ins and outs, and creates new games.

Model Tier

The model tier has a CheckersGame class, which keeps track of an individual game's players and board state. It also has classes for the components of the game, including a Piece class, a Player class, and Type/Color enumerations for the pieces.

Design Improvements

At this time, we believe that all of our class structures and relationships are working well for us, and there isn't anything we would change. This may change after we discover hot spots in our code, or if we're having trouble creating unit tests.

Testing

Acceptance Testing

All 18 acceptance criterias for sprint 1 have been met and passed. For the signin feature there where no problems when testing was done. For starting the game there where problems when setting up the color of pieces that appeared on the side of the board. The problem was that the colors appeared the same for both players, meaning that both players saw themselves as the red checker pieces For sprint 2 we have 7 user stories that have not yet been tested.

Unit Testing and Code Coverage

So far our unit testing plan has focused on testing the if the game exists and if the player has logged in successfully. We plan to test the players moves next to the such as jumping and moving pieces. We have not yet done much code coverage yet due to problems that arrised when trying to use it.

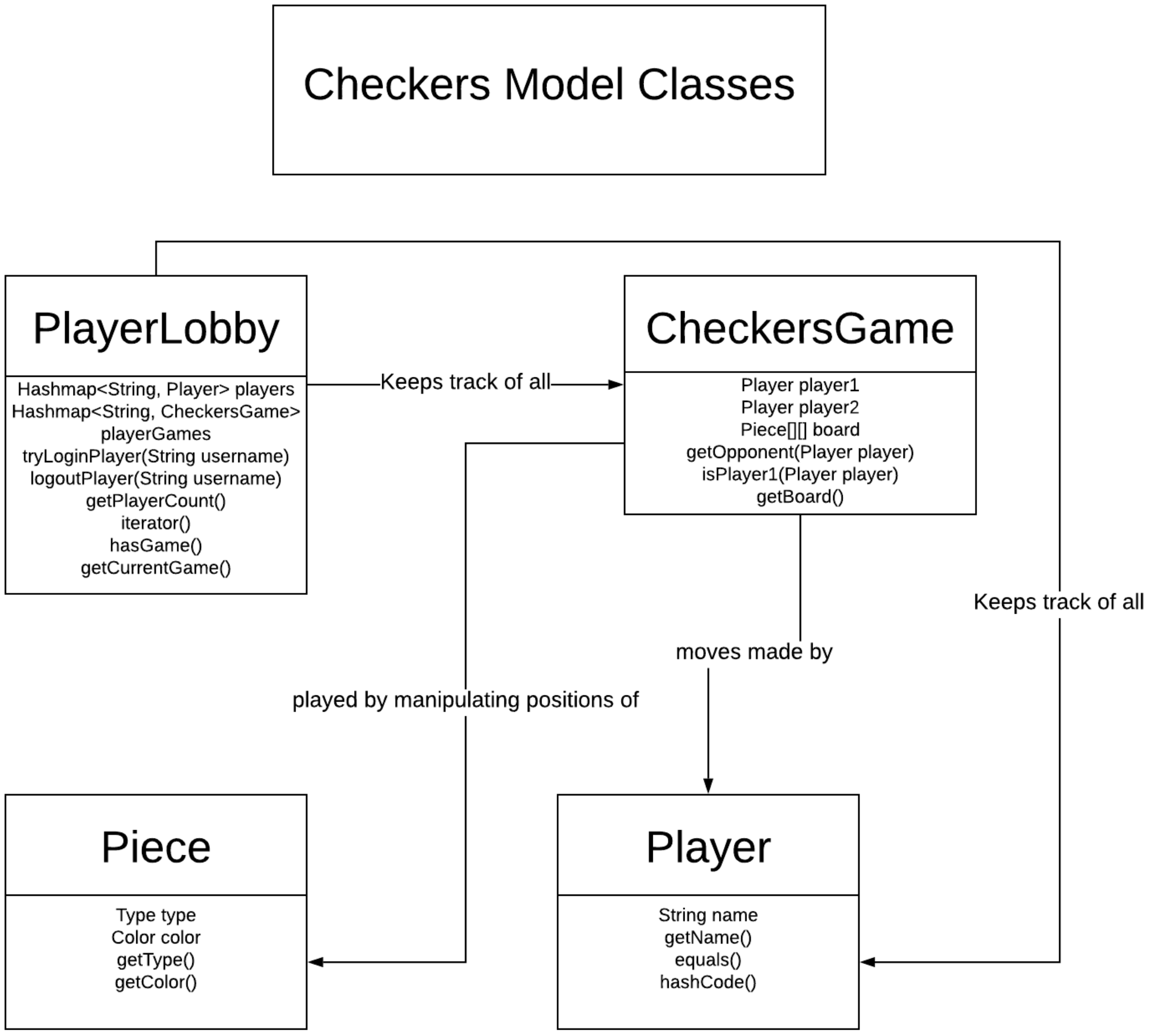


Figure 4: The WebCheckers Web Model Tier Classes