Webcheckers Design Documentation

BrapChads

Modification Log

Date	Version	Description	Author(s)
10/21/2019	1.0	Added overview of user interface	**Carter Nesbitt (CN)**
10/22/2019	1.1	Added overview of major design areas along with domain diagram	**Austin Theodoroff (AT)**
11/03/2019	2.0	Added details pertaining to second release	**CN**

Team Information

Team Name: **BrapChads**

Team Members: **Carter Nesbitt**

Kobe Oley

Sanchit Monga

Austin Theodoroff

Andrew Tarcza

Christopher Curtice

1 Overview

1.1 Executive Summary:

Webcheckers is a web-based application modeling the game of checkers online. Players must sign in with a unique username in order to play, and can sign out anytime. A user may compete against other online players, in addition to spectating games, competing against an AI, and replaying previous matches. Playing a game of Webcheckers involves taking turns between two players moving pieces on a board, while following the general rules of checkers.

1.2 Purpose:

The purpose of this document is to highlight all aspects in the design process for the Webcheckers application, as well as to justify important design decisions and final product. Design considerations are visualized using various models and diagrams in order to capture the decisions made by the development team into a cohesive format.

1.3 Glossary and Acronyms:

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

1.4 Requirements:

1.4.1 Sign-in/Sign-out

- A player must choose an alias not yet taken by any player in the lobby
- The username chosen by the player must be alphanumeric
- A player may not view the players in the lobby until signed in, but may see the number of players in the lobby
- Once signed in the player can view all other players in the lobby
- Once signed in the user has the option to sign-out via a button

1.4.2 Player Lobby/Game Entry

- Any player may challenge any other player to a game of Webcheckers.
- If challenged by an opponent, a player will be taken to the game page and assigned control to the white checker pieces.
- If challenging an opponent in the player lobby, a player will be taken to the game page and assigned control over the red checker pieces.
- Once in a match, both players will be removed from the lobby.

1.4.3 Gameplay

Standard Single Moveset

- Players move diagonally on the dark squares of the checkerboard during a checkers match.
- Pieces can only move one diagonal square at a time for a standard move.

Jump Single Moveset

- Players have the option to jump over their opponents pieces in order to remove them from the game.
- A piece may jump over a diagonal piece of the opposite color when there is an open spot behind the piece to be jumped.
- If a piece has the option to jump over a player, that move must be performed.
- A piece may also jump over multiple pieces by chaining the jumps together in a single match.
- A player may not end their turn with a piece when additional jumps are possible with this piece.
- If there is a branch in which two jumps may be taken by a piece, a player may choose which path they would prefer to take.

King Instantiation and Moveset

- If a piece is successful in reaching the opposing side of the checkerboard, it will be recognized as a king piece.
- A king piece is granted additional move functionality.
    - A king piece can move diagonally backwards and forwards.
    - A king piece can jump backwards.
    - A king piece can chain jumps together that move both backwards and forwards.

1.4.4 Game Resignation

- If a player wishes to forfeit the match, they may do so by pressing the resignation button contained in the game view.
- A player may only press this button when it is their turn.
- If pressed, the player will immediately be returned to the player lobby, incurring a loss for the match.
- The opposing player will be notified that the opposing player resigned, and receive a win for the match.

1.4.5 Game Completion

- If all pieces of a competitors are removed from the game board, the match is over.
- The player with no remaining pieces will receive a message alluding to their loss
- The player with checker pieces on the board will receive a message describing their victory.

1.4.6 Player Lobby Return

- Once a game completes, both players will return to the home menu
- At this menu they will see the current list of players in the player lobby
- At this time, they both may challenge another player in the lobby, or be challenged by such a player.

Definition of MVP:

A Minimum Viable Product (MVP) is a design technique in which, when creating a program, you first create the bare minimum of the project. This means that only the base product is developed, with no consideration for any additional enhancements. In this version of Webcheckers, the base game of checkers has been developed before creating extra features such as ‘Help’ or ‘AI Player’.

MVP Features:

1. Every player must sign-in before playing a game, and be able to sign-out when finished playing.
2. Two players must be able to play a game of checkers based upon the American rules.
3. Either player of a game may choose to resign, at any point, which ends the game.

Enhancements:

Our development team has decided to add two enhancement features to our standard Webcheckers application. These enhancements may be described as a spectator mode, and a replay module. Spectator mode is a setting in which a player may join an active game session to view the match between two players. The spectator may communicate with other spectators via a messaging chat to contribute to the discussion of the ongoing match. A spectator will be able to see all names of other spectators, as well as the names of the two competitors. A spectator may not modify the game of checkers in any way.

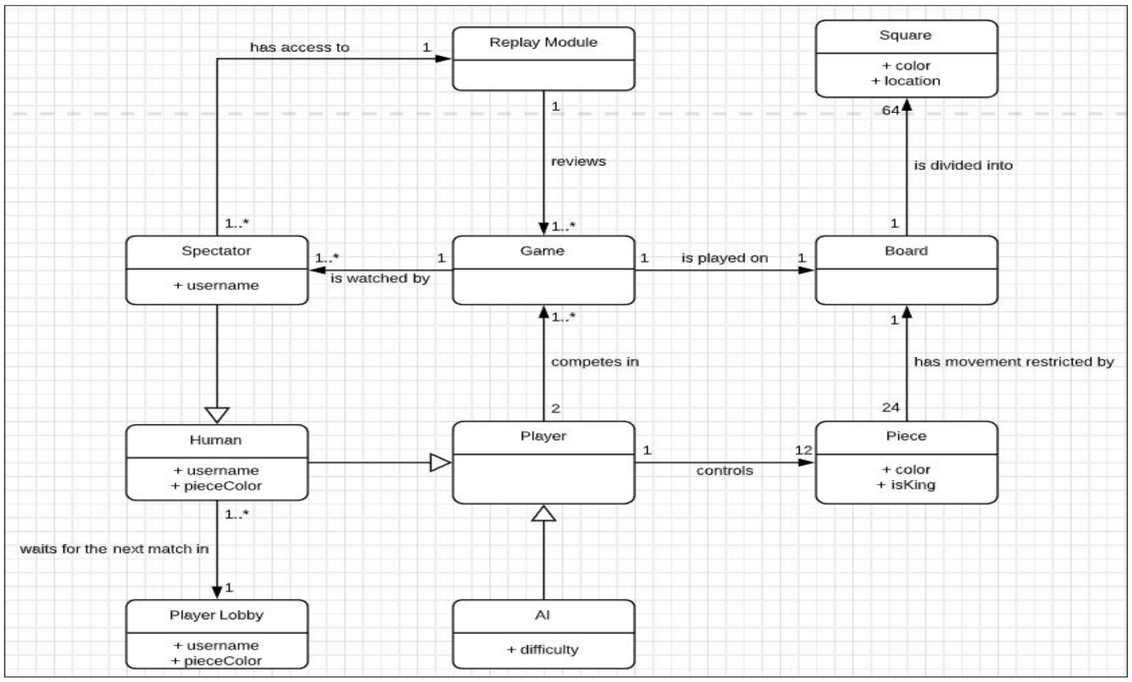
The replay module is a component that will allow for a review of a previous game played. The replay module will maintain a record of the most recent game played by a user. After entering replay mode, a user may traverse through each move made in the match to reflect on strategy, or show-off their victory to friends.

Application Domain:

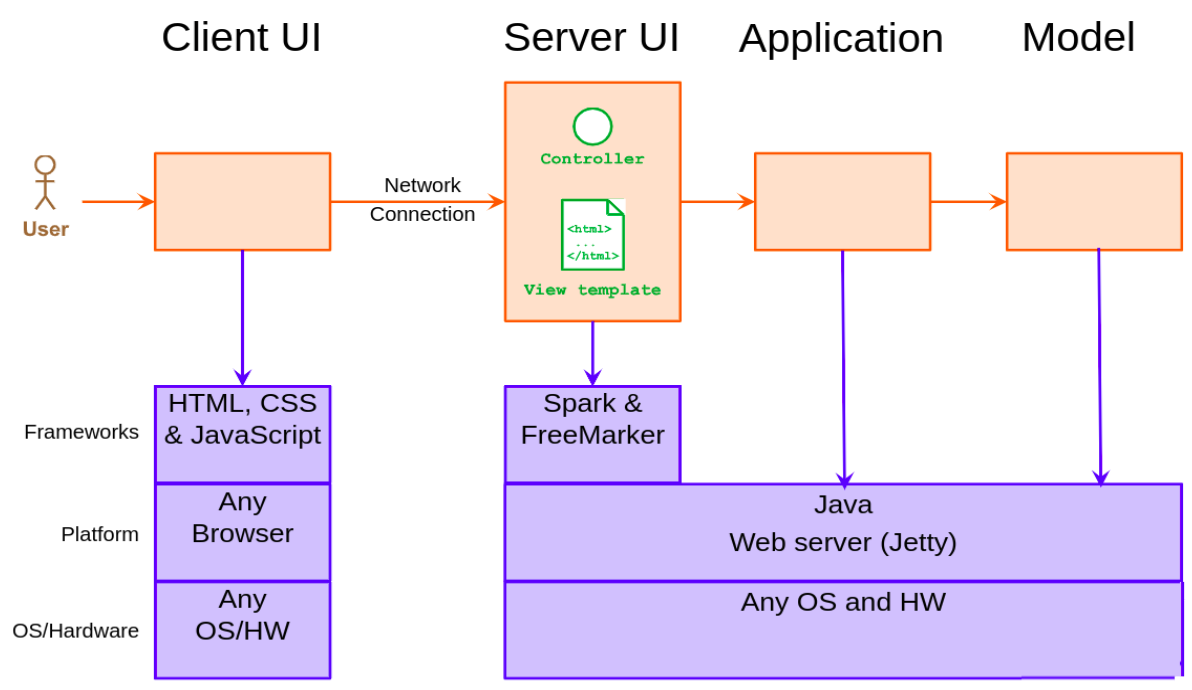
Overview of Major Domain Areas

The diagram represents the domain model and provides a fundamental structure for which the product will be written around. Included in the diagram are the main domain elements which represent different areas of functionality for the code. Written inside of each are the essential attributes that each element will have to hold. The connections (arrows) between each element represents the associations between them. These are all labeled with more detail of how the classes relate to each other. For example, The Board is connected to the Square by an arrow labeled "is divided into", which conveys that the Board class is divided into Square objects.

Domain Area Detail



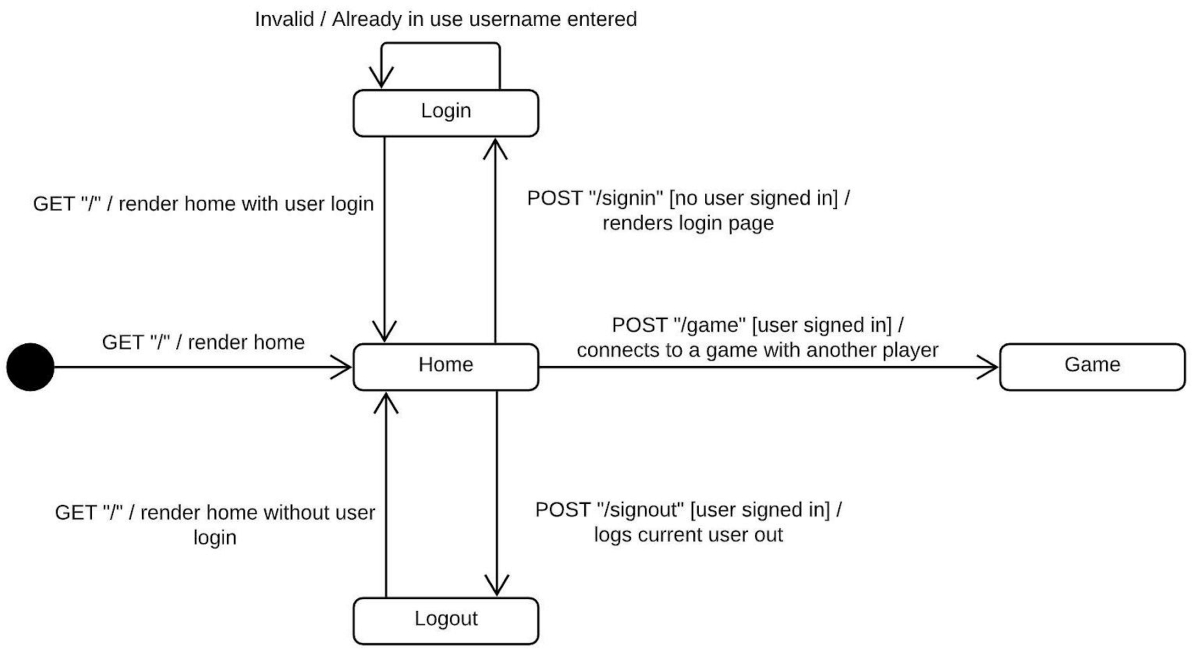
Application Architecture



Summary:

The architecture diagram above represents the high-level architecture overview for the Webcheckers application. The interactions between the client UI and server UI are handled via a network connection. To organize these interactions, various frameworks are utilized. Components such as HTML. JavaScript, and Spark make communication between a client and server fundamental. Once data is received from the client to the server, the server has the capability to pass the data onward to the application and model components of the design. Each web-based framework and supporting application is executed on a browser of the user's choice. The application-based components are executed on top of a Java application. All supporting software is ultimately executed on a supporting hardware and operating system.

Overview of User Interfaces:


Invalid / Already in use username entered
Login
GET "/" / render home with user login
POST "/signin" [no user signed in] / renders login page
GET "/" / render home
Home
POST "/game" [user signed in] / connects to a game with another player
Game
GET "/" / render home without user login
POST "/signout" [user signed in] / logs current user out
Logout

There are three main pages that serve as the interface of the Webcheckers application for a user. What follows is a list of each page that a user may interact with, along with a brief description of its purpose and functionality.

Home Page

The home page serves as the main hub for the WebCheckers site. This is the default route that a user is directed to when they first reach the website. If a user is not yet signed in, they will be notified of the number of players online, and are encouraged to sign-in. Given that a user is already signed in or decides to sign-in, they will also be directed to the home page. However, having met the sign-in requirement, they can now view all players existing in the player lobby. They now have the option to sign out or choose a player to compete against. If a player chooses to sign-out, the home page will refresh and once again display the number of players online. If instead they choose a player to compete against, both players will be redirected to the game interface.

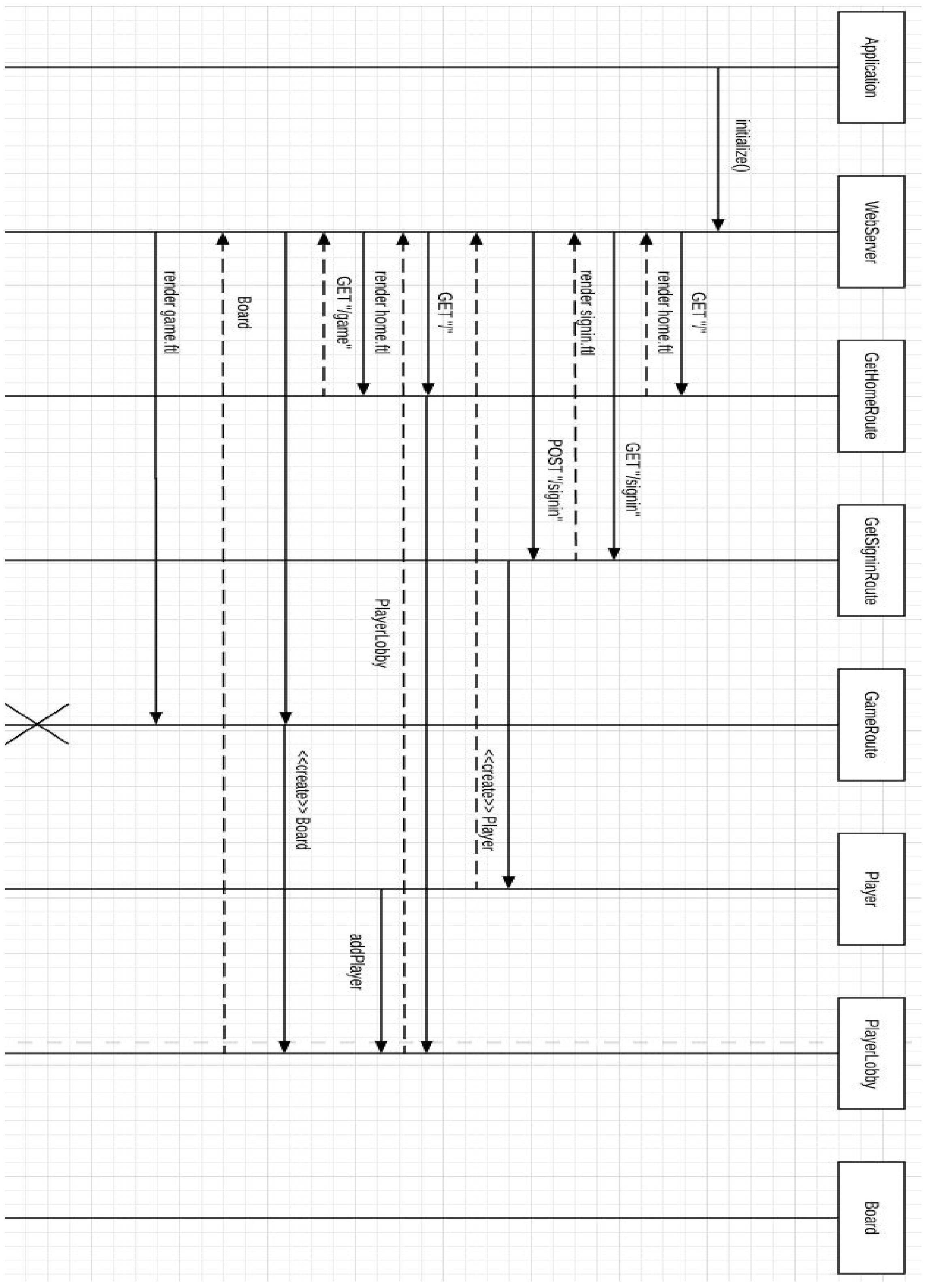
Sign-in Page

The sign-in page is a simple window that allows for a user to enter their preferred username. This name is recognizable by other members in the player lobby. Once a name is entered into the provided text box, and the user hits the submit button, the user will be redirected to the home page. Usernames must be unique, and may only contain alphanumeric characters. Spaces are allowed.

Game Interface

The game interface displays a standard black and white checkerboard containing the correct placement and orientation of the pieces with respect to both players. Both players have the option to resign, revert the turn they are contemplating, or submit the move they have selected. Turns are alternated between both players until a player wins, or one resigns. The name of the opponent can be viewed within the information window, with their respective piece color.

Overview of Client UI Interface



Overview of Server UI Interface:

To be included.

Overview of Application Interface:

To be included.

Overview of Model Interface:

To be included.

Testing:

Aside from creating and running unit tests, impromptu tests were performed via a direct run of the Webcheckers application. Doing so was very helpful for testing small changes within a class and ensuring that the expected result is reflected in the application. Concepts such as move validation, jumping, and final submission checks were prime examples of additions that could quickly be verified through direct testing.

Acceptance Testing:

Acceptance plan helped us to keep track of all the user stories and helped us organize our work. We updated the acceptance plan as we moved forward and as we implemented and tested each story. We tested all the user stories including the previous Sprint's stories combined and made sure that the project as a whole is functional.

Unit Testing and Code Coverage:

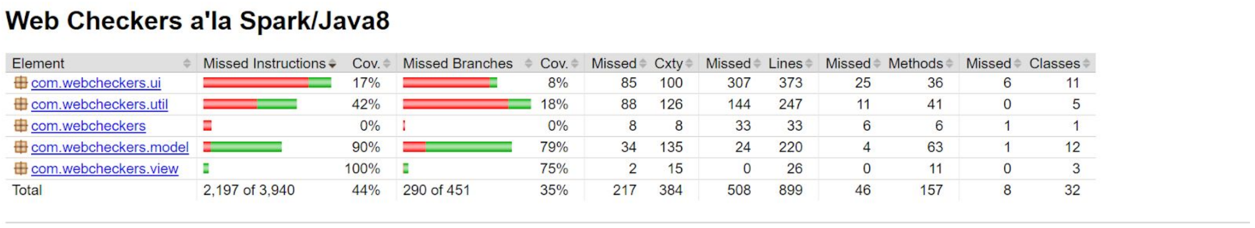
The code coverage results from the unit tests of the Webcheckers application

provides verification that all segments of the application software is functional and used. Having access to such metrics is very important during code review and revisions of any application.

The results of the code coverage are shown below. Our percentage of code coverage falls below what our team deems satisfactory; we are actively working to improve our project-wide code coverage percentage and focus on unit testing to validate our application software.

Code Metrics:

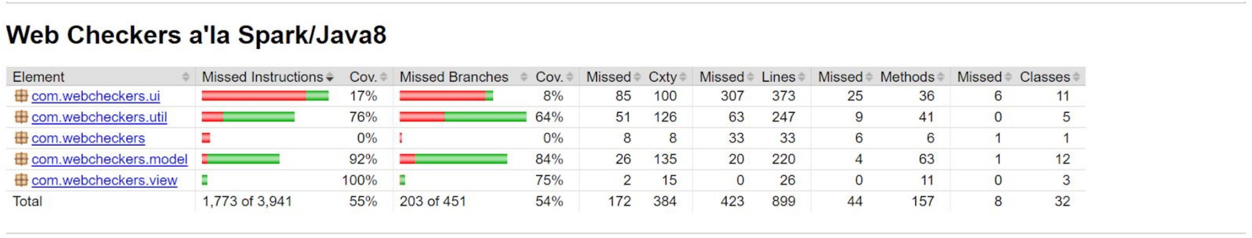
Coverage without Board model fix:



Web Checkers a'la Spark/Java8

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.webcheckers.ui		17%		8%	85	100	307	373	25	36	6	11
com.webcheckers.util		42%		18%	88	126	144	247	11	41	0	5
com.webcheckers		0%		0%	8	8	33	33	6	6	1	1
com.webcheckers.model		90%		79%	34	135	24	220	4	63	1	12
com.webcheckers.view		100%		75%	2	15	0	26	0	11	0	3
Total	2,197 of 3,940	44%	290 of 451	35%	217	384	508	899	46	157	8	32

Coverage with Board model fix:



Web Checkers a'la Spark/Java8

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.webcheckers.ui		17%		8%	85	100	307	373	25	36	6	11
com.webcheckers.util		76%		64%	51	126	63	247	9	41	0	5
com.webcheckers		0%		0%	8	8	33	33	6	6	1	1
com.webcheckers.model		92%		84%	26	135	20	220	4	63	1	12
com.webcheckers.view		100%		75%	2	15	0	26	0	11	0	3
Total	1,773 of 3,941	55%	203 of 451	54%	172	384	423	899	44	157	8	32

Our unit tests were designed around a fix for the Board Model that broke other functionality, thus when they fix was reverted, which hurt the coverage rate significantly due to newly failing tests.

Design Quality and Possible Revisions:

The Webcheckers application adheres to the principles of software design that are pertinent to maintainable and effective code. Segregating the UI, model, and application into sub-packages within the application allows for an easier method to follow design principles such as high-cohesion and low-coupling.

While the design of the Webcheckers application is certainly sufficient, there is much opportunity for improvement within both the quality of the current design of the application, as well as potential for revisions.

To follow release version 2.0 will be a code review for the Webcheckers application. This will entail a team meeting dedicated to a deep investigation and discussion of the current product in development with respect to the software that has been written thus far. Critiques and modifications of the code base are certain to be present during the code review. The ultimate goal of this meeting is to remove unnecessary and non-maintainable code throughout the Webcheckers application in order to continue development without code-related impedances.