**Test Plan**

for

Flame Checkers

Version 1.2 approved

Prepared by Cameron Stevenson, Connor Oldmixon, Sergio Cardoza-Molina, Zachary Golla

Flame Army

29 October 2018

# Contents

[Revisions ii](#_Toc117484243)

[1 Introduction 1](#_Toc117484244)

[1.1 Purpose 1](#_Toc117484245)

[1.2 System Overview 1](#_Toc117484246)

[1.3 Definitions, Acronyms and Abbreviations 1](#_Toc117484247)

[1.4 Supporting Materials 1](#_Toc117484248)

[1.5 Document Overview 1](#_Toc117484249)

[2 Architecture 2](#_Toc117484250)

[2.1 Overview 2](#_Toc117484251)

[2.2 Component 1..n 2](#_Toc117484252)

[3 High-Level Design 3](#_Toc117484253)

[4 Low-Level Design 4](#_Toc117484254)

[4.1 Modules Overview 4](#_Toc117484257)

[4.2 Module Specifications 4](#_Toc117484258)

[4.2.1 Module X1 4](#_Toc117484259)

[4.2.2 Module X2 5](#_Toc117484260)

[Appendix A – Group Log 6](#_Toc117484261)

# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Cameron Stevenson | Initial Creation | 10/16/18 |
| 1.1 | Cameron Stevenson | Updated sections 1, 2, and 3 | 10/29/18 |
| 1.2 | Connor Oldmixon | Updated section 4 (Cameron, Zac, Sergio and I all did 2 subsections) | 10/29/18 |

# Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Term** |
| AI | Artificial Intelligence |
| FC | Flame Checkers |
| JRE | Java Runtime Environment |

# Introduction

This test plan will include a complete description of all test strategies, environment requirements, and functions to be tested for the production Flame Checkers.

### Test Plan Objectives

The objective of this test plan is to verify the following functions.

1. The system allows night mode to change the color scheme.
2. The user can play against an AI.
3. The user can play against another user.
4. Allow a computer mouse to interact with the game.
5. The user can select their checkers and move only theirs.
6. The game board is displayed as an 8x8 grid.
7. The options menu allows the game settings to be changed

# Test Strategy

### System Test

We will test different operating systems such as Mac OS, Windows and Linux to make sure each can run flame checkers.

Ex. Playing flame checkers on each operating system listed above.

### Stress/Performance Test

The tester will use negative testing to observe how FC handles incorrect inputs.

1. Tester will play a game and try moving a piece that is not theirs.
2. Tester try to move a piece to an incorrect position.
3. Tester will stress the system by repeatedly selecting any piece.
4. The tester will try and stress the system by pressing different buttons in the options menu.

### Beta Test

The tester will play the game of checkers on their personal computer.

### User Acceptance Test

All testers will play the final version of flame checkers a minimum of eight times. The first time playing the 1-player mode, and the second time playing the 2-player mode. The third and fourth time will be played with the time mode enabled for 1-player and 2-player game modes. The fifth and sixth time will be played with the night mode enabled for 1-player and 2-player game modes. The seventh and eighth time will be played with the night mode and time mode enabled for 1-player and 2-player game modes.

# Environment Requirements

### Environment 1

A laptop with available memory and IntelliJ downloaded.

# Functions To Be Tested

### Night Mode

1. Click on “Options Menu” button in Main Menu

* Expected output: Option Menu with orange background, Night Mode check box unselected

1. Click on “Night Mode” checkbox

* Expected output: Option Menu background changes to grey, “Night Mode” check box becomes selected

1. Click on “Confirm” button in Option Menu

* Expected output: Option window closes, Main Menu background is changed to grey

1. Click on “Start Game” button in Main Menu

* Expected output: a game board with black and grey checkered pattern, populated with black game pieces on the bottom three rows and red game pieces on the top three rows

### Timed Mode

1. Click on “Options Menu” button in Main Menu

* Expected output: Option Menu with orange background, “Normal Mode” radio button selected

1. Click on “Timed Mode” button in Option Menu

* Expected output: “Timed Mode” radio button becomes selected and “Normal Mode” radio button becomes unselected

1. Click on “Confirm” button in Option Menu

* Expected output: Option window closes

1. Click on “Start Game” button in Main Menu

* Expected output: a game board with red and orange checkered pattern, populated with black game pieces on the bottom three rows and red game pieces on the top three rows, a timer counting down from 10 seconds, and above the timer “Turn: Black Team” to the right of the board

1. Click on a black piece

* Expected output: the square with the piece on it is highlighted, and its available moves are highlighted

1. Click on a highlighted square to move the piece to

* Expected output: black piece is moved to that square, timer resets to 10 seconds and starts counting down again, text above timer changes to “Turn: Red Team”

1. Click on a black piece

* Expected output: nothing should change, no highlighting

1. Wait for timer to run out

* Expected output: timer resets to 10 seconds, text above timer changes to “Turn: Black Team”

### Normal Mode

### 2 player Mode

1. In order to test the two players option the user must click the ‘Options Menu’ button at the bottom of the main window.

* The result of this should be that the options menu will then pop up displaying all possible options for the user

1. The user will then select the ‘Player vs. Player’ option from the ‘Options Menu’ and then select ‘Confirm’ at the bottom of the options window.

* The result should be the ‘Options Menu’ will then close down taking in the user selected option of ‘Player vs. Player’

1. The user will the select the ‘Start Game’ option from the bottom the main window

* The result of this should be a new window popping up displaying a game board with the appropriate color patterns (8x8 alternating color) and pieces (12 pieces for each player). The window will also alert the users whose turn it is and which set of checker pieces belongs to each user. (Red belongs to player 1 and black belongs to player 2).

1. User, player one, is now allowed to move any piece that is available to move to their desired destinations. As long as the move is compliance to the rules of checkers.

* The result is that player one’s piece will be moved and then it will display a notification for player two that it is their turn

1. User, player two, is now allowed to move any piece that is available to move to their desired destinations. As long as the move is compliance to the rules of checkers.

* The result is that player two’s piece will be moved and then it will notify player one that it is their turn to make a move.

1. Once either player one or player two makes the final and “winning” move.

* The result should be that the game will display which player won the match.

### Home page

1. The tester runs the application.

* The tester should see a graphic user interface representing Flame Checkers

1. The tester clicks the “options menu” button

* Option submenu opens in new window

1. The tester clicks the “exit” button

* Application closes.

1. The tester clicks the “start game” button.

* Application starts a new game of checkers.

### Game Board

1. In order to test the gameboard the user must click the ‘Start Game’ button at the bottom of the main window.

* The result of this should be a new window popping up displaying a game board with the appropriate color patterns (8x8 alternating red and orange) and pieces (12 pieces for each player) with red for one player (top portion of board) and black for another (bottom portion of board).

1. The user can the click on any piece on the board.

* The result should be that all available moves for that piece will light up and display their paths.

1. The user can then select where they want to move their piece based on the provided legal possibilities

* The result should be that the game piece will move from its current tile on the board to the one selected

### Game Pieces

1. Click on game piece in checkerboard
   * Expected output: Game piece background highlights color and selectable squares highlight as well
2. Click on game piece again in checkerboard
   * Expected output: Game piece background is no longer highlighted and selectable square no longer show
3. Click on highlighted and selectable square in checkerboard
   * Expected output: Game piece moves to selected square
4. Click on square that is not highlighted after selecting piece to move
   * Expected output: All pieces become deselected and user can select which piece they want to move next
5. Click on square that is highlighted to take another piece
   * Expected output: Game piece moves to select square and piece taken disappears
6. Game piece reaches other side of game board and becomes knighted
   * Expected output: Game piece illustration changes from regular piece to a knight piece and gains all knight attributes
7. Last Game Piece is taken
   * Expected output: Game ends and winner window appears

### Option Menu

1. Click on “Options Menu” in the Main Menu
   * Expected output: Option Menu with Orange background, “Player vs. Player”, “Normal Mode”, and “Easy” radio buttons selected, “Night Mode” check box unchecked.
2. Click on “Night Mode” check box
   * Expected output: “Night Mode” check box becomes checked and option menu background changes to grey
3. Click on “Player vs. AI” radio button
   * Expected output: “Player vs AI” radio button becomes selected, “Player vs. Player” radio button becomes unselected
4. Click on “Timed Mode” radio button
   * Expected output: “Timed Mode” radio button becomes selected, “Normal Mode” radio button becomes unselected
5. Click on “Medium” radio button
   * Expected output: “Medium” radio button becomes selected, “Easy” radio button becomes unselected, “Hard” radio button stays unselected