**Test Plan**

for

Project 1 ChessMate Application

Version 1.0 approved

Prepared by Team FlameArmy

March 8, 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)

# Introduction

The overall approach for testing ChessMate that will drive the testing for this software is Unit Testing, Integration Testing, and System Testing.

### Test Plan Objectives

*The objective is to verify the following functions:*

1. *1. Display the chess rules pertaining to this system.*
2. *Allow the user to play a standard game of chess against an ai.*
3. *Move chess pieces according to their respective class using mouse hardware.*
4. *Move chess pieces according to their respective class using keyboard input.*
5. *Allow the user to play against an AI*
6. *The user can see the score board.*
7. *Give the user an option to play chess under the constrain of a timer.*
8. *Allow the user to see the move log of the game.*
9. *The system has a button that allows the game to restart at any time.*
10. *The system allows the alternative game type “Chaos” mode.*
11. *The system records and displays the name and high score of an individual.*

# Test Strategy

<*For each of the following, you should explain how you plan on testing your application, provide a brief description, and possibly scenario/example of how you will accomplish that sort of test. Some of these may not pertain to your application>*

1. *Displaying the chess rules pertaining to this game.*
2. *The tester shall check that the main menu has a button that pops up a JLablel.*
3. *Tester Clicks the Button that displays the rules of the game.*
4. *The tester checks the output using Junit Testing*

*Example code:*

*PrintStream oldOut = System.out;*

*OutputStream os = new ByteArrayOutputStream();*

*PrintStream ps = new PrintStream (os);*

*System.setOut(ps)*

*String output = new String(os.toString());*

*System.setOut(oldout);*

*assertTrue(output.contains(“Rules of regular chess game”);*

1. *Allow the user to play a standard game of chess against an AI.*
2. *The tester checks that the menu first displays the option to play against AI button “play AI”*
3. *The tester checks if the GUI switches from the main menu to game mode for “Play AI”*
4. *Move chess pieces according to their respective class using mouse hardware.*
5. *The tester begins a game of type “Play AI”*
6. *The test moves each piece according to their class at least for three different ways for each.*
7. *The test captures an AI chess piece to check if the pieces move to the right location.*
8. *Move chess pieces according to their respective class using keyboard input.*
9. *The test being a game of type “Play AI”*
10. *The tester moves each piece according to their class at least in for different ways using keyboard input and not mouse input.*
11. *The tester captures an AI chess piece to check if the piece move the right location.*
12. *Allow the user to play against an AI*
13. *The tester checks if in the main menu play “Play Button”*
14. *Test clicks the button to see if the it branches to the state where the player is playing the AI.*
15. *The system allows the user to see the scoreboard.*
16. *The tester checks if the main menu has a scoreboard display and clicks it.*
17. *The tester checks if the scoreboard corresponds to the games he or she has played.*
18. *The user checks that zero games have been played.*
19. *The tester checks if the scoreboard displays the correct number of games after playing one game.*
20. *Give the system allows the option to play chess under the constrain of a timer.*
21. *The tester chooses a game of “Play AI Timed” to play against an AI.*
22. *The tester determines if a loss of time results in a move lose.*
23. *The tester lets the begins a timed game and forgoes all moves to check if it results in an AI win.*
24. *Allow the user to see the move log of the game.*
25. *The tester begins a game a regular game of chess and inputs chess commands instead of using the mouse.*
26. *The tester checks if the JUnit TextArea displays the history of the moves played.*

*9 The system has a button that allows the game to restart at any time.*

1. *The tester begins a regular game against the AI*
2. *The tester confirms the game restarts during different states of the game.*
3. *The tester begins a player vs player game.*
4. *The tester confirms the restarts during different states of the game.*
5. *The system allows the alternative game type “Chaos” mode.*
6. *The tester checks if the menu has the option to place in Chaos mode.*
7. *The tester begins a game in Chaos mode and records the positions of the chess pieces.*
8. *The tester repeats a game in Chaos mode and records the positions of the chess pieces.*
9. *The system records and displays the name and high score of an individual*
10. *The tester checks if the main menu has the option to check if there is a high score menu*
11. *The tester completes a game and checks the game score matches the number of games played in that session.*

### System Test

The approach for the system test is to test all components of the chess game are working correctly. The tester will play each game type at least one while checking the scoreboard matches the correct number of games.

### Stress/Performance Test

The tester will use negative testing approach to test how the chess game handles incorrect inputs.

1. The tester plays a regular game and attempts to move pieces to positions not allowed by that piece.
2. The tester attempts to move chess position into incorrect chessboard locations using keyboard input.
3. The Tester uses attempts to stress the system by attempting to move pieces into incorrect positions several times
4. The tester stresses the system by pressing different buttons multiple times.

### Automated Test

The automated test will be performed using JUnit testing. The Tester will create at least one test case for each class and every function within that class.

# Environment Requirements

*The environment will take place using a laptop. It will be performed using Java Eclipse to run both the test an game.*

# Functions To Be Tested

<*This section is critical for validatoinl, and needs to map the functions and requirements laid out in the SRS that need to be validated.>*

Class ChessTimer

startTime(), estimateTime(), restartTime(), returnInMin(), toMinutes(long millis)

Rook Class

Rook(booean taken2, String image2)

Bishop

Bishop(Boolean taken2, String image2)

Chessboard

Chessboard()

King Class

King(Boolean taken, String image)

Knight Class

Knight(Boolean taken, String image)

Pawn Class

Pawn(Boolean taken, String image)

SetProm(Boolean status)

Piece Class

Piece(Boolean taken2, String image2)

setTaken(boolean taken)

setImage(String image)

Queen Class

Queen(boolean taken2, String image2)