Software Engineering Group Project

System Test Specification

|  |  |
| --- | --- |
| Author: | Jasper Crabb [jac127], Ng Wei Yuan [wyn] |
| Config Ref: | TSGroup18 |
| Date: | 19th April 2023 |
| Version: | 1.1 |
| Status: | Release |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2023

CONTENTS

1. Introduction 3

1.1 Purpose of this Document 3

1.2 Scope 3

1.3 Objectives 3

2. System Test Specification 3

REFERENCES 8

DOCUMENT HISTORY 9

# Introduction

## Purpose of this Document

This document will be used as reference to system testing and specifies how we will test the system as a whole.

## Scope

This document’s tests are derived from requirements specification for the software group project [1].

The document’s standard and format are from the test procedure standards for the software group project [2]

The use case document [3] is referenced several times in this document and it is recommended that the reader is familiar with it.

## Objectives

The objective of this document is to guide the system tester during testing, so that they know what the system should be able to do and how it should behave when doing so.

# System Test Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Ref** | **Req being tested** | **Test Content** | **Input** | **Output** | **Pass Criteria** |
| SE-F-001 | FR1 | Check that the user is prompted with a menu on startup | Opening or running the program | The main menu with “New Game”, “Continue”, Replay” and “Quit” should be displayed | The menu has the following buttons, in the layout specified in the use case document |
| SE-F-002 | FR1 | Check that “New Game” button functions correctly | “New Game” button is pressed | The program opens the player setup screen | The player setup screen is displayed as specified in the use case description |
| SE-F-003 | FR1 | Check that player 1 can enter their name | The string “Andy” is entered into the text box | The player 1 name is set to “Andy” | The player 1 name string is set to “Andy” |
| SE-F-004 | FR1 | Check that player 1 can choose their colour (white) | The player selects button “White” | The player’s piece colour is set to white. | The Boolean values for the colours should be White = 1, Black = 0 |
| SE-F-005 | FR1 | Check that player 1 can choose their colour (black) | The player selects button “Black” | The player’s piece colour is set to black. | The Boolean values for the colours should be White = 0, Black = 1 |
| SE-F-006 | FR1 | Check that player 2 can enter their name | The string “Chris” is entered into the text box | The player 2 name is set to “Chris” | The player 2 name string is set to “Chris” |
| SE-F-007 | FR1 | Check that player 2 is assigned the correct colour | Player 1’s colour has been selected | The colour assigned to player 2 is the opposite of player 1’s | The Boolean values for player 2’s colours should be the inverse of player 1’s |
| SE-F-008 | FR1 | Check that the cancel button works correctly | Player 1 presses the “cancel” button | The menu displayed changes to the main menu | The program goes back to the beginning |
| SE-F-009 | FR1 | Check that the button to start the game works correctly | Player 1 presses the “next” button | The program leaves the player setup screen and brings up the chess game | When pressed the chess game should begin, with all parameters matching what was selected in the previous menus as specified in the use case document |
| SE-F-010 | FR1 | Check that the “continue” button functions correctly | The “continue” button is selected | The load game screen is bought up as shown in the use case document | The program gets the location of previous save games and displays them to the user |
| SE-F-011 | FR2 | Check that the player names are tracked for the duration of the game | The tester inputs the test strings “Andy” and “Chris” into the text boxes on the player configuration screen | The player name is stored for the duration of the game and is displayed alongside the chess board | The strings “Andy” and “Chris” are stored in memory and do not change for the duration of the game. The pieces must also be associated to the same player for the duration of the game |
| SE-F-012 | FR2 | Check that the player colour is tracked for the duration of the game | The player presses the button for either black or white on the player configuration screen | The colours of each player are stored for each of the players for the duration of the game | If the colour white is selected for player 1 then it should be associated with them for the duration of the game. The same applies to black and the other player |
| SE-F-013 | FR2 | Check that program is keeping track of where the players pieces are for the duration of the game | Player 1 moves pawn E2 to E4 | The pieces and their locations are stored for the duration of the game | The tiles in the 2d array have the correct x and y values. In this case the E2 pawn’s tile should have x = 4 and y = 1. |
| SE-F-014 | FR3 | Check that program visually indicates which player should move | The player selects the menu option “start a new game” or makes a move in game | The program displays which player should move | The player who’s turn it will be displayed in the top right of the game |
| SE-F-015 | FR3 | Check that the players' names are displayed on the screen | The tester inputs the test strings “Andy” and “Chris” into the text boxes on the player configuration screen | The program displays the player names | The names displayed are Andy and Chris |
| SE-F-016 | FR3 | Check that the board is displayed on the screen | The player selects the menu option “start a new game” or makes a move in game | The chess board should be displayed on the screen | The chess board is correctly displayed with no graphical errors |
| SE-F-017 | FR3 | Check that the pieces are displayed | The player loads in a game using our test file “testStart.txt” [4] | The pieces should appear on the board in the standard starting position | The pieces should be displayed in the correct locations, in the correct colour with no graphical errors |
| SE-F-018 | FR4 | Check if the player can only select their pieces | Player 1 should attempt to select the D7 pawn | The player will be told if they try to move a piece that isn’t theirs | The pawn should not be selected |
| SE-F-019 | FR4 | Check that the player can only move one piece at a time | Player 1 should attempt select both the D2 and F2 pawn | The player will be told if they try to move more than one piece | Only the F2 pawn can be moved |
| SE-F-020 | FR5 | Check that the possible legal moves are shown to the user this includes locations where an opponent’s piece is captured | The player loads in a game using our test file “testLegal.txt” [5] | The program will show the legal moves for that turn to the user by colouring squares that can be moved to | After the rook has been selected all squares on row 7 should have dots as well as all squares on the ‘a’ file. The squares around the kings should also have dots when the kings is selected |
| SE-F-021 | FR5 | Check that the pieces can moved | Player 1 selects the H2 pawn and moves it 1 square forward | The piece selected will move | The H2 pawn’s coordinates are updated to x = 7, y = 1 |
| SE-F-022 | FR5 | Check that the piece can only be moved if it is a valid move. | Player 1 attempts to move the queen to A4. After loading up the “testStart.txt” [4] | The player will be told the piece cannot be moved | The program does not allow an illegal move and the queen’s coordinates are not changed |
| SE-F-023 | FR5 | Check that the player cannot move a piece to an invalid square when the king is in check | Player 1 attempts move the A1 rook to A4 after loading up the “testCheck.txt” [6] | The program will prevent the player from making this move | The program does not allow a move to an illegal square and the rook’s coordinates are not changed |
| SE-F-024 | FR5 | Check that the pawn can do the En Passant special move | Player 1 attempts to capture a pawn piece using their own where all appropriate conditions are met for an en-passant. | The program will show that the pawn piece is captured, and the piece is shown at a correct position. | The program lets the pawn piece captures the enemy pawn piece and ends at a correct position. |
| SE-F-025 | FR5 | Check that the king can castle on either side | Player 1 attempts to perform a castling with an unmoved rook. Repeat the procedure on another rook. | The king piece and a rook piece moved to an expected spot which indicates that a castling is being done. | The program allows player to perform castling and is being shown so. |
| SE-F-026 | FR5 | Check that the king can’t castle in check | Player 1 attempts to perform a castling with an unmoved rook while in check. Repeat the procedure with the other rook. | The program does not indicate the move being legal hence nothing happens. | The program does not allow player to perform castling and is being shown so. |
| SE-F-027 | FR5 | Check that the king can’t castle if the rook or the king have moved | Player 1 moves either king or a rook and back to its original position, then attempts to perform castling. Repeat procedure on another piece until all pieces are accounted for. | The program does not indicate the move being legal hence nothing happens. | The program does not allow player to perform castling and is being shown so. |
| SE-F-028 | FR6 | Check that the program detects check and indicates it to the user | Player 1 moves the queen to B3 after opening “initiateCheck.txt” [7] | The player is told that they are in check by their king being highlighted in red | The program highlights the king in red |
| SE-F-029 | FR7 | Check that the program detects checkmate | Player 1 moves the A1 rook to A8 after loading up “testCheckmate.txt” [8] | The player is told that they have lost | The program ends the chess game and stores the winner |
| SE-F-030 | FR8 | Check that program clearly indicates a game over | A condition that causes the game to end is triggered | The program tells the players that the game is over | The program displays a game over screen at the correct time |
| SE-F-031 | FR8 | Check that the program has an option to resign | The player selects the menu option “start a new game” or makes a move in game | There is a button somewhere that allows either player to resign | The player that resigns loses and the game is over |
| SE-F-032 | FR8 | Check that the players can make a draw | The player selects the menu option “start a new game” or makes a move in game | There is a button somewhere that allows either player to ask for a draw | Both players are presented with an option to draw and if accepted the game ends |
| SE-F-033 | FR8 | Check that the program saves the game as it ends | A condition that causes the game to end is triggered | n/a | The entire game and its details are saved to disk |
| SE-F-034 | FR9 | Check that the players can exit the game | The player selects the menu option “start a new game” or makes a move in game | There is a button somewhere that allows either player to quit the game | The game is adjourned and saved for later |
| SE-F-035 | FR9 | Check that the game is saved to disk each move | The player makes a move in game | The text file for the game should update with the current FEN string for the game | The correct game state is saved to the text file |
| SE-F-036 | FR10 | Check that the player can select a game to replay | The player selects the menu option “restore previous game” | The player is presented with a menu that has a list of games to replay | All the of the saved games are present and can be selected (highlighted) |
| SE-F-037 | FR10 | Check that the player can go back to the main menu | The player selects the menu option “restore previous game” | The current menu has a button labelled “cancel” | When the button is pressed it should take the user back to the main menu |
| SE-F-038 | FR10 | Check that the player can delete a game save file | The player selects the menu option “restore previous game” | The current menu has a button labelled “Erase” | When the button is pressed the save game that is highlighted is deleted |
| SE-F-039 | FR10 | Check that the player can load the selected game | The player selects the menu option “restore previous game” | The current menu has a button labelled “Load” | When the button is pressed the save game that is highlighted should be loaded into and be able to be replayed |
| SE-F-040 | FR10 | Check that the player has the option to review the game backwards | The player selects the menu option “restore previous game” | There is a button that allows the player to move backwards through the game | The board displays the previous move from the current game state |
| SE-F-041 | FR10 | Check that the previous move button cannot be pressed if there is no previous move | The player presses the “Load” button | The button should be greyed out | When the button is pressed it should not do anything |
| SE-F-042 | FR10 | Check that the player has the option to review the game forwards | The player selects the menu option “restore previous game” | There is a button that allows the player to move forwards through the game | The board displays the next move from the current game state |
| SE-F-043 | FR10 | Check that the next move button cannot be pressed if there is not another move | The player presses the “Load” button | The button should be greyed out | When the button is pressed it should not do anything |
| SE-F-044 | FR10 | Check the program allows the user to exit the replay mode | The player selects the menu option “restore previous game” | There is a button that allows the player to exit the replay mode | The button takes the user back to the start menu of the program |
| SE-F-045 | FR11 | Check the program saves each move made | The player makes a move in game | n/a | A file is updated with the new board position |
| SE-F-046 | FR11 | Check that the program can restore a game that has been quit | The player selects the menu option “restore previous game” | The is presented with an option to restore the game in question | When the game is selected it is in the correct (previous) state |
| SE-EI-001 | External Interface | Check that a user’s input is acted in on in under a second | Any input | The program responds within 1 second | The output is correct and took less than one second to display |
| SE-EI-002 | External Interface | Check that the program can run on the IS PCs | n/a | n/a | The program runs and behaves correctly on the IS PCs |

REFERENCES

[1] Software Engineering Group Projects – Chess Tutor Requirements Specification 1.1 (Release)

[2] QA Document SE.QA.06 - – Test Procedure Standards/2.1 (Release)

[3] Software Engineering Group Project – Use Case Document Group 18 1.2 (Release)

[4] testStart.txt – A file containing a FEN string representing a standard chess board layout

[5] testLegal.txt – A file containing a FEN string that we can easily use to test legal moves

[6] testCheck.txt – A file containing a FEN string that we can easily use to test check

[7] initiateCheck.txt – A file containing a FEN string that we can easily use to test causing check

[8] testCheckmate.txt - A file containing a FEN string that we can easily use to test checkmate functionality

DOCUMENT HISTORY

| *Version* | *Issue No.* | *Date* | *Changes made to document* | *Changed by* |
| --- | --- | --- | --- | --- |
| 0.1 | N/A | 15/02/23 | N/A - original version | Jac127 |
| 0.2 | N/A | 22/02/23 | Added required sections and changed test reference numbers | Jac127 |
| 0.3 | #14 | 01/03/23 | Added more specific tests for some features | Jac127 |
| 0.4 | N/A | 06/03/23 | Made changes outlined in the review meeting | Jac127 |
| 1.0 | N/A | 06/03/23 | Document release | Jac127 |
| 1.1 | N/A | 19/04/23 | Changed document according to feedback | Jac127 |
| 1.2 | N/A | 10/05/23 | Added tests about check, checkmate, castling, and en-passant. Minor grammar and spelling fix. | wyn |