SE 3XA3: Test Plan
KingMe

Team 9, KingMe
Ardhendu Barge 400066133
Dylan Smith 001314410
Thaneegan Chandrasekara 400022748

March 6, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
March 5, 2021	1.0	Revision 0 (all members)

1 General Information

1.1 Purpose

The purpose of this document is to outline the testing plan for our project. The testing we will be performing will be done to ensure that the application being built meets all the functional and non functional requirements outlined in the SRS document.

1.2 Scope

The scope of this document is to show the type of tests that will be performed and explain the reasoning behind each. The test plan will cover what types of test plans will be performed, how they will be executed, plans for automated testing, PoC test case, the schedule for testing and what tools we will be using for code coverage. The test cases will outline the expected system behavior in response to Tester input for the game of checkers. The previous implementation of the project did not contain any test cases and thus our group will be creating new test cases that will specifically cover the functionality of our new implementation, including testing the GUI.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation2	**Definition**
FR	Functional Requirement
NFR	Non-functional Requirement
TP	Test Plan
SRS	Software Requirement Specification
POC	Proof Of Concept
GUI	Graphical User Interface

1.4 Overview of Document

In the rest of this document we will be discussing the approach we plan to take to test our application. We will introduce the tools we plan to use as

well as the various testing methods we will use to verify and validate that our requirements are being met.

2 Plan

2.1 Software Description

The software being tested is a re-implementation of a checkers application. It will offer the user the ability to play checkers against another player or against the computer. The game will adhere to all of the rules of checkers and will show users all the possible valid moves they can make when they select one of their pieces. The application will also include a tutorial outlining how to play the game and an option to restart the game at any time.

2.2 Test Team

Test team will consist of all three members of team namely Ardhendu, Thaneegan and Dylan. Each tester will have a specific area of testing based on their expertise. We also plan to have users test the system using an exploratory approach to help find bugs in the user interface, and help improve the usability of the application.

2.3 Automated Testing Approach

We will use automated testing to perform unit testing and integration testing. We will need to use unit testing to ensure the application adheres to all the rules of checkers. We will provide the various functions with input and ensure the correct output. We will use unittest to automate the process, and we will run the tests any time one of the functions are modified. We will also use integration testing to ensure that different functions and modules interact with one another correctly. They will also be automated using unittest and will be run when any changes are made to one of the components or when a new component is added.

2.4 Testing Tools

We will be using unittest and vscode for testing the application. We will also use many different testing techniques. We will use a combination of

whitebox, blackbox and exploratory testing to test the checkers logic and to test the GUI.

2.5 Testing Schedule

The Gantt chart illustrates the subtasks for tesing and the corresponding resource breakdown. Please see the Gantt Chart at the following url: Project Schedule.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Application Options

Homescreen

1. FR-AO-1

   Type: Functional, Manual

   Initial State: Game is not launched (Clarify with TA).

   Input: Run the game/application.

   Output: Window with home screen displays the game mode options.

   How test will be performed: Tester will launch the game application on the computer.

2. FR-AO-2

   Type: Functional, Manual

   Initial State: Game is launched and currently on the homescreen.

   Input: Nothing.

   Output: The homescreen waits for Tester input .

   How test will be performed: Tester will wait on the homescreen for a determined amount of time to ensure that the application remains on the homescreen.

New Game

1. FR-AO-3

   Type: Functional, Manual

   Initial State: Game is currently in progress.

   Input: Tester selects new game button.

   Output: Application returns to the homescreen.

   How test will be performed: Tester will begin a game. After varying amounts of moves the Tester will select the new game button to ensure that the applicaiton returns to the homescreen.

Game Modes

1. FR-AO-4 Type: Functional, Manual

   Initial State: Application is on the homescreen.

   Input: Tester selects 1-player mode.

   Output: When game commences the computer will move pieces on it's turn.

   How test will be performed: Tester will choose the 1-player game mode and begin a game. After the Tester makes their move check to see that the computer moves one of it's pieces.

2. FR-AO-5 Type: Functional, Manual

   Initial State: Application is on the homescreen.

   Input: Tester selects 2-player mode.

   Output: When game commences the application will alternate which pieces can be moved but will expect input from Testers to move the pieces.

   How test will be performed: Tester will choose the 2-player game mode and begin a game. After the Tester moves a red piece make sure the red pieces will not move again. Then tester will ensure they can move a black piece. Repeat for several moves.

Colour Choice

1. FR-AO-6 Type: Functional, Manual

   Initial State: Tester has chosen 1-player mode.

   Input: Tester selects red/black as their colour.

   Output: When game commences the Tester will be able to move the red/black pieces and the computer will move the black/red.

   How test will be performed: Tester will choose the red pieces. When the game begins the Tester will move the red pieces to ensure the game allows it. This will be tested with FR-AO-4 to ensure that the computer then moves the black pieces. The Tester will make several more moves ensuring this pattern hold. The Tester will then start a new game (FR-AO-3) and choose the black pieces and repeat the process.

3.1.2 Gameplay

Valid Moves

1. FR-GP-1 Type: Functional, Manual

   Initial State: Game is in progress and it is the testers turn.

   Input: Tester selects one of their pieces.

   Output: The game will highlight the selected pieces and indicate all the possible moves that piece can make with small dots.

   How test will be performed: Tester will play a game and select one of their pieces to ensure that the piece is highlighted and that all the possible moves are indicated.

2. FR-GP-2 Type: Functional, Manual

   Initial State: Game is in progress and tester has promoted a piece to a king.

   Input: Tester chooses to move the king piece.

   Output: The game will highlight the selected pieces and indicate all the possible moves that piece can make with small dots.

   How test will be performed: Tester will play a game and select a king piece and make sure that the valid moves include moves in both directions.

3. FR-GP-3 Type: Functional, Manual

   Initial State: Game is in progress and tester has selected a piece that is one move from the end of the board.

   Input: Tester chooses move the piece to the end of the board.

   Output: The game will update the state of the board to reflect the tester's move and the piece will be promoted to a king.

   How test will be performed: Tester will play a game and select one of their pieces that is one move from the end of the board. The tester will then choose to move the piece to the end of the board. The board will update to reflect the testers move, this includes promoting the piece that was moved to a king.

4. FR-GP-4 Type: Functional, Manual

   Initial State: Game is in progress and tester has selected a piece.

   Input: Tester chooses to take one of the valid moves.

   Output: The game will update the state of the board to reflect the tester's move.

   How test will be performed: Tester will play a game and select one of their pieces. The tester will then choose a valid move to make and ensure that the board updates to reflect the move.

5. FR-GP-5 Type: Functional, Manual

   Initial State: Game is in progress and tester has selected a piece.

   Input: Tester chooses to take one of the valid moves that captures an opposing piece.

   Output: The game will update the state of the board to reflect the tester's move and the opposing piece is removed from the board.

   How test will be performed: Tester will play a game and select one of their pieces. The tester will then choose to capture an opposing piece to ensure that the board updates to reflect the move, and that the opposing piece is removed from the board.

Invalid Moves

1. FR-GP-6 Type: Functional, Manual

   Initial State: Game is in progress and tester has selected a piece.

   Input: Tester chooses to move to an invalid square.

   Output: The game will notify the tester that the move was invalid, and the board will remain in the same state.

   How test will be performed: Tester will play a game and select one of their pieces. The tester will then choose to move to an invalid square and see if the they are notified that the move is invalid and that the board remains in it's current state.

2. FR-GP-7 Type: Functional, Manual

   Initial State: Game is in progress and it is the computer's turn to make a move.

   Input: Tester chooses a piece to move.

   Output: The game will notify the tester that it is not their turn and will not change the state of the board.

   How test will be performed: Tester will play a game while the computer is making a move the tester will attempt to move a piece. The tester will then note if the game notified them that is was not their turn and that the board did not change state.

End of Game

1. FR-GP-8 Type: Functional, Manual

   Initial State: Game is in progress.

   Input: Tester chooses a piece to move, and that move prevents their opponent from having any more valid moves.

   Output: The game will notify the tester that the end of game has been reached.

   How test will be performed: Tester will play a game until the computer has no more valid moves. They will then check to make sure the game notifies them that the end of game has been reached.

2. FR-GP-9 Type: Functional, Manual

   Initial State: Game is in progress.

   Input: Tester moves a piece that will force them to lose the game.

   Output: The computer will win the game. The game will notify the tester that the end of game has been reached.

   How test will be performed: Tester will play a until they have no more valid moves. They will then check to make sure the game notifies them that the end of game has been reached.

3.2 Tests for Nonfunctional Requirements

3.2.1 Usability and Humanity

1. NFR-UH-1

   Type: Non-functional, Manual

   Initial State: Game application is launched with default colour scheme.

   Input/Condition: Change colour scheme to the preference.

   Output/Result: Colour scheme is changed to one chosen by the user.

   How test will be performed: Tester will execute the steps to change colour scheme people .

2. NFR-UH-2

   Type: Non-functional, Manual

   Initial State: Game application is launched.

   Input/Condition: Select "Tutorial" option on main screen.

   Output/Result: A pdf will open with all the instructions in it.

   How test will be performed: Tester will manually perform the steps to open the instructions.

3.2.2 Performance

1. NFR-P-1

   Type: Non-functional, Automated

   Initial State: There are two possible initial states. 1) User has launched application but not started any game yet. 2) User is in game but wishes to start a new game.

   Input: User clicks on new game button in menu.

   Output: The board is reset to the initial state.

   How test will be performed: A timer will be used to compute the time taken from when the user clicks on the spot they wish to move their piece to and when the board displays the updated state.

2. NFR-P-2

   Type: Non-functional, Automated

   Initial State: The checkers game has started and it is the user's turn to make a move.

   Input: User clicks on one of their checker's pieces and then clicks on the spot on the board they wish to move the piece to.

   Output: The board reflects the move by displaying the selected piece in the user's desired spot.

   How test will be performed: A timer will be used to compute the time taken from when the user clicks on the spot they wish to move their piece to and when the board displays the updated state.

3. NFR-P-3

   Type: Non-functional, Automated

   Initial State: The checkers game has started and it is the AI's turn to make a move.

   Input: No input from user.

Output: The AI computes the next move to be made and the board reflects the move accordingly.

How test will be performed: A timer will be used to compute the time taken from when the user's turn is completed and when the board displays the updated state.

4. NFR-P-4

   Type: Non-functional, Automated

   Initial State: A checkers game has started and is active.

   Input: User or AI moves a piece.

   Output: FPS and time printed in test log.

   How test will be performed: Test case will check FPS periodically and compare to see if it is above or equal to 60 at all times.

3.2.3 Operational and Environment

1. NFR-OE-1

   Type: Non-functional, Manual

   Initial State: Game application is not launched

   Input/Condition: Disconnect from internet as well as remove any ethernet from computer/laptop and launch the game.

   Output/Result: Game launches and user is play is able to it.

   How test will be performed: Tester will manually perform the steps to open the instructions.

3.2.4 Maintainability and Support

1. NFR-MS-1

   Type: Non-functional, Manual

   Initial State: N/A

   Input/Condition: Tester can view the documentation.

Output/Result: Tester is able to view the documentation and understand it.

How test will be performed: Tester will manually perform code inspection and review.

2. NFR-MS-2

   Type: Non-functional, Manual

   Initial State: N/A

   Input/Condition: Tester will check the code and do code inspection as well as code review.

   Output/Result: Tester can understand the code logic and cohesion between different code modules.

   How test will be performed: Tester will manually perform code inspection and review.

3. NFR-MS-3

   Type: Non-functional, Manual

   Initial State: N/A

   Input/Condition: Tester will perform checks on the code in addition to code inspection as well as code review.

   Output/Result: Tester will be able to find any discrepancies in code style after analyzing the code.

   How test will be performed: Tester will manually perform code inspection and review.

3.2.5 Security

1. NFR-SR-1

   Type: Non-functional, Static, Manual

   Initial State: N/A

   Input: Source code will be reviewed by tester through code inspection.

Output: Tester will determine if there are any requests for input that is not related to making a move on the checkers board or accessing a functionality on the menu.

How test will be performed: Tester will conduct a code inspection to identify any spots in the source code where they may be a security vulnerability. Tester will ensure no requests for external input is being made, outside of 1) making a move on the checkers board and 2) clicking on of the menu functionalities.

3.3 Traceability Between Test Cases and Requirements

Table 3: **Requirement Traceability Matrix**

Requirement ID	**Requirement Description**	**Priority**	**Test Case ID**
FR1	Home Screen	High	FR-AO-1 FR-AO-2
FR2	Allow user to choose piece colour	Low	FR-AO6
FR3	Allow only valid moves	High	FR-GP-6
FR4	Show valid moves	Medium	FR-GP-1 FR-GP-2
FR5	Invalid move notification	Medium	FR-GP-6
FR6	Display is updated after each move	High	FR-GP-4 FR-GP-3 FR-GP-5
FR7	Start new game	Medium	FR-AO-3 FR-AO-6
FR8	Win state	High	FR-GP-8
FR9	Loss state	High	FR-GP-9
FR10	Win/Loss state notification	Medium	FR-GP-8
FR11	Remove jumped pieces	High	FR-GP-5
FR12	User's turn	High	FR-GP-7
FR13	Black pieces have first turn	High	FR-AO-6
FR14	King pieces can move both ways	Medium	FR-GP-2
FR15	Piece Promotion	High	FR-GP-3
FR16	Highlight selected piece	Low	FR-GP-1
FR17	Different game modes	Low	FR-AO-4 FR-AO-5
NFR1	New game (response time)	Medium	NFR-P-1
NFR2	User makes a move (response time)	High	NFR-P-2
NFR3	AI makes a move (response time)	High	NFR-P-3
NFR4	FPS of application	Low	NFR-P-4
NFR5	Documentation of code	Low	NFR-MS-1
NFR6	Modularized code	Medium	NFR-MS-2
NFR7	Coding style	Low	NFR-MS-3
NFR8	Game should not compromise security	High	NFR-SR-1
NFR3	Game has different colour schemes	Medium	NFR-UH-1
NFR5	Game should have a tutorial	High	NFR-UH-2
NFR11	Game does not need internet	High	NFR-OE-1

4 Tests for Proof of Concept

For our proof of concept we wanted to demonstrate that we were able to integrate the original project with a GUI to increase the ease of use of the application. The testing we did consisted mainly of Functional/Manual/Dynamic testing as it required a tester to observe the output of the application on the screen when the user interacts with the application.

4.1 Display

State of Board

1. POC-T-1

   Type: Functional, Dynamic, Manual

   Initial State: GUI has not been given a board to display.

   Input: Random board state.

   Output: The GUI displays the board state given, correctly, on the screen.

   How test will be performed: Tester will create a random board state. Tester will call the GUI display with the random board state. Tester will verify that the board state displayed is the same as what was entered.

Homescreen

1. POC-T-2

   Type: Functional, Dynamic, Manual

   Initial State: Application has not been launched.

   Input: Tester launches the application

   Output: The GUI displays the home screen, with options to start a new game or to view a tutorial.

   How test will be performed: Tester will launch the application and verify that the home screen is displayed with options to start new game and view tutorial.

4.2 User Interactions

Get location of Clicks

1. POC-T-3

   Type: Functional, Dynamic, Manual

   Initial State: Game is in progress.

   Input: Tester clicks on the board.

   Output: Function receives mouse click, calculates position of click on board and prints the board coordinates of which square was clicked.

   How test will be performed: Tester will begin a game and click on the board. They will verify that the square printed corresponds to the square that was clicked.

2. POC-T-4

   Type: Functional, Dynamic, Manual

   Initial State: Game is in progress.

   Input: Tester clicks on a button.

   Output: Function receives mouse click, calculates position of click and prints the the name of the button that was clicked.

   How test will be performed: Tester will begin a game and click on a butotn. They will verify that button printed corresponds to the button they clicked.

Integrating Clicks

1. POC-T-5

   Type: Functional, Dynamic, Manual

   Initial State: Game is in progress and it is the tester's turn.

   Input: Tester clicks on a one of their piece and clicks on a destination.

   Output: Game displays the new board state after the move has been applied.

   How test will be performed: Tester will begin a game and select one of their pieces. They will then select a destination square and verify that the game updates the display to reflect the move they made.

2. POC-T-6

   Type: Functional, Dynamic, Manual

   Initial State: Game is in progress.

   Input: Tester clicks the new game button.

   Output: Game resets the state of the board and indicates that a new game has begun.

   How test will be performed: Tester will start a game and make several moves. They will then click the new game button and verify that the board is reset and a message is displayed indicating the new game has begun.

3. POC-T-7

   Type: Functional, Dynamic, Manual

   Initial State: Application has been launched.

   Input: Tester clicks the tutorial button.

   Output: Game displays a pop-up outlining how to use the application.

   How test will be performed: Tester will launch the application and at any time will select the tutorial button. They will verify that a tutorial window is displayed outlining how to use the application.

4. POC-T-8

   Type: Functional, Dynamic, Manual

   Initial State: Game is in progress and it is the tester's turn.

   Input: Tester clicks on an opposing piece.

   Output: Game displays an error message saying that the piece does not belong to them. The state of the board remains the same and the tester is able to select one of their pieces.

   How test will be performed: Tester will play a game and select and opposing piece to move. The tester will then verify that an appropriate error message is displayed and the board state is not updated. The tester will then attempt to move one of their own pieces to check that the game hasn't skipped their turn.

5 Comparison to Existing Implementation

In the original implementation of the Checkers game, there was no documents/files relating to testing of the program. There were numerous bugs and errors within the program including game ending early, pieces being able to be moved anywhere and more. Testing should have been done to ensure these errors did not exist. Based on their implementation, it can be seen that unit testing could've been used to effectively test the 3 classes (MinMax, Board, Main) and their respective functions to ensure the game logic worked correctly. The unit testing could have been completed via automated testing and would not have required much manual testing. On the other hand, our new implementation of the game requires both unit testing and integration testing. Unit testing will be created to test the game logic since the original implementation did not have any. Integration testing will now be required because the addition of a GUI introduces the possibility for new issues. Integration testing will provide validation that the game logic and the GUI work successfully together. Since the GUI is difficult to test with automation, this will require manual testing.

6 Unit Testing Plan

6.1 Unit testing of internal functions

1. IF-T-1

   Type: Functional, Dynamic, Automated

   Initial State: Game is in progress.

   Input: Player moves a piece and is not a win move

   Output: AI makes a optimized move

   How test will be performed: Automated test will check if the move is in the possible moves list

2. IF-T-2

   Type: Functional, Dynamic, Automated

   Initial State: Game is launched.

   Input: New game is intitated.

Output: All black and white pieces spawn in their intial position

How test will be performed: List of initial positions of all pawns would be compared to position of pawns in the game

3. IF-T-3

   Type: Functional, Dynamic, Automated

   Initial State: Game is in progress

   Input: Move is made by either AI or player

   Output: State of the board is changed

   How test will be performed: List of initial positions of all pawns would be compared to position of pawns in the game

4. IF-T-4

   Type: Functional, Dynamic, Automated

   Initial State: Game is in progress

   Input: Move is made by either AI or player

   Output: State of the board is changed

   How test will be performed: The turn boolean variable will compared with current turn to determine if each player is getting turns consecutively

7 Appendix

N/A

7.1 Symbolic Parameters

N/A

7.2 Usability Survey Questions

To test the usability of our application we plan to have users test the application and answer several survey questions. We will also ask for general feedback of the game. We will ask the users the several questions and will ask

them to choose one of the following answers: Stronly agree, agree, neutral, disagree, strongly disagree. The questions we plan to include in the survey are as follows:

- The application was very easy to use.
- I found the tutorial was very descriptive and made using the app much easier.
- When moving a piece the dots representing the valid moves made it easier to choose a move.
- When selecting a piece it is highlighted this helped assure me that the application registered my choice.
- The game responded in a timely fashion.
- The application was visually appealing.
- The buttons and options were self explanatory, and caused no confusion.
- I was pleased with the experience of playing checkers.