SE 3XA3: Test Report
Title of Project

Team #, Team Name
Harry Fu, fuh6
Mengshan Cui, cuim2
Xingjian Ke, kex1

December 4, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Dec 14, 2018	1.0	Test report Rev1

Test report is an important deliverable which is prepared at the end of a Testing project, or rather after Testing is completed. The prime objective of this document is to explain various details and activities about the Testing performed for the Project, to the respective stakeholders like Senior Management, Client etc. This document provides a report of the results of the testing performed on the Platform Perils application. Both system testing and quality testing are covered. Traceability between testing and both requirements and modules is given in the

nal section.

1 Functional Requirements Evaluation

Description of Tests: The purpose of these tests is to ensure that the user is able to use the software according to the given requirements. These tests will include mouse click testing, algorithm testing, graphical user interface testing and data storage testing.

FR-MSC-1
Result: The user is able to click a piece and move this piece to another cell

FR-MSC-2
Result: The user is able to see the highlighted cells(possible move of this piece) when the user clicked on this piece.

FR-MSC-3
Result: The user is able to unselected this piece by click it twice. User click this piece and click another piece this program gives no response for that.

FR-ALG-1
Result: The checkmate panel showed and game over when checkmate happened(there is no possible move for all pieces in one side.)

FR-GUI-1
Result: During the game is in progress, there is a timer showed at the top of chessboard

FR-GUI-2
Result: After user click the start game button, the game will start and there will be a panel at the top of chessboard shows current player state(white turn/black turn).

FR-GUI-3
Result: After user click the resgin button, a pop-up window showed up and say the current player is lost the game.

2 Nonfunctional Requirements Evaluation

2.1 Usability

2.1.1 GUI Testing

Description of Tests: Usability of the Graphical User Interface (GUI) was tested by a small test group of ten McMaster students who are not in Software Engineering to better reflect the technological experience of the potential users for this program. Participants were monitored to observe the time it took them to perform the requested task. After the participants were finished with all tasks, each participant filled out a feedback form, evaluating the system on its usability.

Test Name : NF-1
Results: All participants were able to complete the task of launching the program successfully on operating systems Windows, Mac OS or Linux, and outputing a setup menu. The majority of participants would be able to perform this within 3 minutes.

Test Name : NF-2
Results: All participants were able to complete the task of choosing the role and setting the timer within 2 minutes, exceeding the requirement that the majority of participants would be able to perform this within 2 minutes.

Test Name : NF-3
Results: The average rating was Satisfactory or above on each criterion of

ease of download, ease of use, understandability of text and symbols, and overall satisfaction. Feedback was received that the system might be improved if it allowed the user to set the screen size for chess board. Additionally, feedback was received that if the user guide can be placed in the game dialogue.

Test Name : NF-4
Results: The average rating from the test group on each category from test NF-3 was equal to or greater than the average rating from the test group on the corresponding category for this test, except for Overall Satisfaction. This indicates that the reimplementation is at least as usable or has higher usability than the Existing Implementation on Usability.

2.2 Performance

Description of Tests: Generating a chess board on the screen was initiated repeatedly on default settings and the time between initiation of the action and commencement of display the setup page was timed. The performance of the program is tested during participants complete the following tasks.

Test Name : NF-5
Results: The time delay was under the required processing time of 10 seconds on all iterations of the test.

Test Name : NF-6
Results: Once the timer reached 0 between two valid moves, the game ended and a dialogue displayed that the current player lose the game.

Test Name : NF-7
Results: All participants were able to complete the task of moving the pieces of the pre-selected side during a chess game. The majority of participants

Test Name : NF-8
Results: All participants were able to complete the task of moving the selected valid square, and displayed correctly on the screen. The majority of participants would be able to perform this which ensures the chess game can be executed successfully.

3 Comparison to Existing Implementation

3.1 NF-4

NF-4 in Tests for Nonfunctional Requirements(Usability):

3.1.1 criteria of ease of download:

ChessOOP: 4
ChessAce: 4

3.1.2 ease of use:

ChessOOP: 3
ChessAce: 4

3.1.3 understandability of text and symbols:

ChessOOP: 4
ChessAce: 5

3.1.4 overall satisfaction:

ChessOOP: 3.5
ChessAce: 4.5

3.2 POC

POC test cases are about two basic movements not included in the existing implementation. Because ChessAce is more user-friendly and more useable than ChessOOP, so for performance part ChessAce is better than ChessOOP. Then because ChessAce has more features than ChessOOP such as promotion, castling, resign button and pause button. So for completeness ChessAce is better than ChessOOP.

4 Unit Testing

Each piece, its sub-classes and main module in this program associates with a relatively huge datapool, an 8*8 matrix board strcture, so constructing an automated test will require to repeated large-scale data but using GUI, and console output to do integration testing could do a same checking performance with a higher time eciency. So after the reversion 0 team MIF just decided to use only integration test based on GUI and console output to verify that it can avoid requirement omission, violation, inconsistency and other unexpected operation.

5 Changes Due to Testing

After the integration testing, there are three main parts has been changed:

5.1 Path for Queen and Rook

After the first integrating (GUI based) testing, there were several problems found in Queen and Rook's possible movement especially for the diagonal movements. The algorithm has been changed and some bugs were fixed.

5.2 Checkmate algorithm

Our initial implementation for the checkmate algorithm is to check if two Kings have possible movement, but after first integrating test we found that the piece which checked the king could be eliminated by another pieces and will not check this king anymore. So we decide to implement this algorithm by checking if there are any possible move for other pieces. This will make sure the correctness of the checkmate.

5.3 Is_king_in_danger algorithm

Changed the checking king is in danger algorithm after the first integrating testing. Removed the conflict between the real and the virtual pieces.

6 System Testing

Test Name	NF-9
Initial State	The chess game is operating on screen.
Input	The King and Rook have not been moved yet and there is no pieces between them.
Expected Output	Castling will be performed as Rook is moved to King's left side.

Table 2: NF-9

Test Name	NF-10
Initial State	The chess game is operating on screen.
Input	During the game, the pawn reaches its eighth rank.
Expected Output	Promotion will be performed as pawn is immediately replaced by a queen.

Table 3: NF-10

7 Trace to Requirements

Test	Requirements
Functional Requirements Testing	
FR-MSC-1	FR1, FR3, FR15
FR-MSC-2	FR2, FR12, FR15
FR-MSC-3	FR16
FR-ALG-1	FR6, FR17
FR-GUI-1	FR4, FR14
FR-GUI-2	FR4, FR5, FR7, FR8
FR-GUI-3	FR4, FR13
Non-functional Requirements Testing	
NF-1	3.3.1, 3.4.2, 3.4.3, 3.5.2
NF-2	3.3.3
NF-3	3.1, 3.3.1, 3.7
NF-4	3.3.
NF-5	3.2
NF-6	3.3.2
NF-7	3.4.1
NF-8	3.3.4
System Testing	
NF-9	3.2, 3.3.4
NF-10	3.2, 3.3.4

Table 4: Trace Between Tests and Requirements

8 Trace to Modules

Test	Modules
Functional Requirements Testing	
FR-MSC-1	M1, M4
FR-MSC-2	M1, M4, M5, M8
FR-MSC-3	M1, M4
FR-ALG-1	M4, M7, M8
FR-GUI-1	M2, M4
FR-GUI-2	M4, M5, M7
FR-GUI-3	M3, M4
Non-functional Requirements Testing	
NF-1	M1
NF-2	M1, M2
NF-3	NULL
NF-4	NULL
NF-5	M1
NF-6	M2, M3, M4
NF-7	M1, M8
NF-8	M1, M5, M7, M8
System Testing	
NF-9	M1, M4, M5, M7, M8
NF-10	M1, M4, M5, M7, M8

Table 5: Trace Between Tests and Modules

9 Code Coverage Metrics

The ChessAce group has managed to produce roughly 92 percent code coverage through our tests. This number is based o the fact that all of the modules have been covered in integrating testing based on GUI and console output. Please refer to the trace to modules section. There it is clearly shown that we have covered every module multiple times, once again confirming the 92 percent coverage rate.