SE 3XA3: Test Report
Mastermind

Team 204, Trident Inc
Justin Prez, prezj
Justin Rosner, rosnej1
Harshil Modi, modih1

April 6, 2020

Contents

1 Functional Requirements Evaluation **1**
- 1.1 User Input . . . 1
- 1.2 Game Environment . . . 2
- 1.3 Game Logic . . . 3

2 Nonfunctional Requirements Evaluation **3**
- 2.1 Look and Feel . . . 3
- 2.2 Usability and Humanity Requirements . . . 3
- 2.3 Performance . . . 4
- 2.4 Operational and Environmental . . . 4

3 Comparison to Existing Implementation **4**

4 Unit Testing **5**

5 Changes Due to Testing **5**
- 5.1 Functional Requirements Evaluation . . . 5
- 5.2 Look and Feel Requirements Evaluation . . . 5
- 5.3 Usability and Humanity Evaluation . . . 6
- 5.4 Performance Evaluation . . . 6
- 5.5 Operation and Environmental Evaluation . . . 6

6 Automated Testing **6**

7 Trace to Requirements **6**

8 Trace to Modules **7**

9 Code Coverage Metrics **9**

List of Tables

1 **Revision History** . . . ii
2 Trace Between Test Cases and Requirements . . . 7
3 Trace Between Test Cases and Requirements . . . 8

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
2020-04-05	1.0	Initial version of Test Report

This document provides a report of the testing that was performed on the Mastermind App. The doc also provides changes performed as a result of testing, a traceabiliy matrix for the test results, and code converage metrics.

1 Functional Requirements Evaluation

1.1 User Input

- Test Name: UI1 - Test selection of blue colour peg
  Result: This test was successful as as a blue peg was placed in the next available location upon clicking on the blue coloured peg from the list of pegs.

- Test Name: UI2 - Test selection of yellow colour peg
  Result: This test was successful as as a yellow peg was placed in the next available location upon clicking on the yellow coloured peg from the list of pegs.

- Test Name: UI3 - Test selection of red colour peg
  Result: This test was successful as as a red peg was placed in the next available location upon clicking on the red coloured peg from the list of pegs.

- Test Name: UI4 - Test selection of green colour peg
  Result: This test was successful as as a green peg was placed in the next available location upon clicking on the green coloured peg from the list of pegs.

- Test Name: UI5 - Test selection of white colour peg
  Result: This test was successful as as a white peg was placed in the next available location upon clicking on the white coloured peg from the list of pegs.

- Test Name: UI6 - Test selection of purple colour peg
  Result: This test was successful as as a purple peg was placed in the next available location upon clicking on the purple coloured peg from the list of pegs.

- Test name: UI7: Test simultaneous button presses
  Result: This test was successful as the device would not register the

event when two buttons were pressed at the same time. No errors were generated.

- Test name: UI8: Test undo button
  Result: This test was successful as the user is able to undo their latest changes by pressing the undo button. The undo button also does not work if hints have been provided for a row. This is because then the user will be able to cheat the game.

- Test name: UI9: Test game instructions are displayed to the user upon start up of the game
  Result: This test was successful as the user is able to see the game instructions as soon as the game start or the user goes to the start screen.

- Test name: UI10: Test close game button
  Result: This test was successful as the game screen goes to the start menu upon clicking on the 'X' button in the top left corner of the game.

- Test name: UI11: Test new game button
  Result: This test was successful as upon clicking on the new game button the user is directed to the game screen.

1.2 Game Environment

- Test name: GE1: Test load start menu upon startup
  Result: This test was successful. When the user click on the Mastermind app, the user is directed to the start menu.

- Test name: GE2: Test load help menu upon startup
  Result: This test was successful. When the user click on the Mastermind app, the user is directed to the start menu which also contains the help menu.

- Test name: GE3: Test load game board
  Result: This test was successful. When the user click on the play button the user the game board is loaded to the screen.

1.3 Game Logic

- Test name: GL1: Test full peg row
  Result: This test was successful. When the user clicked on a next peg to be added, the next peg was added to the next row when the previous row contained 4 pegs already.

- Test name: GL2: Test incorrect guess (that doesn't finish the game)
  Result: This test was successful. When a row is full and the game has not been won, the hints are added to show the number of correct guess and the user is able to continue adding pegs to the next row.

- Test name: GL3: Test win game
  Result: This test was successful. When the user correctly guesses the combination, a win state message is displayed to the user and the user is directed to the main menu.

- Test name: GL4: Test lose game
  Result: This test was successful. When the user guesses 40 pegs and the game as not been won, the user gets a message saying that they lost the game and the correct combination is displayed to the user.

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel

- Test name: LF1: Test that the game board resembles the classic game board from the 70's
  Result: The testers believe that the overall feel of the game is a newer version of the 70's game with a wooden background.

2.2 Usability and Humanity Requirements

- Test name: UHR1: Test that the game shall be playable by user MIN_AGE and up
  Result: The developers asked the their younger sibling, parents and grandparents to follow the instructions and play the game. The tests was successful as everyone was able to play the game without any problems.

- Test name: UHR2: Test that the user will remember how to play the game
  Result: This test was successful. When the game was given to a player who played the game a week ago, the user was still able to remember how to play the game without any instructions.

2.3 Performance

- Test name: P1: Speed and latency
  Result: When carrying out the following test no visual latency was observed. Furthermore, Flutter build in features did not identity much latency either.

- Test name: P2: Precision and Accuracy
  Result: When playing the game multiple times and changing the options, the game performed as intended. No in-game glitches were experienced during the game.

- Test name: P3: System Stress Test
  Result: This test passed. When the tester played the game and spam clicked a peg the game was able to update faster then the user was able to click.

2.4 Operational and Environmental

- Test name: OE1: The game can be loaded on an Android system
  Result: The test passed as the game was successfully able to be downloaded on an Android device.

3 Comparison to Existing Implementation

This section does not apply as tests were not provided in the original implementation and our test changes could not be tested on the original implementation.

4 Unit Testing

Unit testing was only performed on the game logic section as Unit testing the physical UI is not possible. Each function within the this section was tested using flutter test framework. Unit tests was performed for both returned value and exception handling. See MIS for more information and unit_test.dart for our unit testing.

All 4 of the unit tests passed. Within each unit test - multiple tests were performed to test both regular and boundary conditions. Overall, all the functions produced the correct output to the provided input.

5 Changes Due to Testing

5.1 Functional Requirements Evaluation

Few small changes were made to Mastermind due to tests corresponding to the functional requirements. With the recent revision of our test plan (Rev 1) we designed new functional tests for features of our game. Specifically, tests UI7 and UI8 were introduced to test the buttons handling multiple inputs and the functionality of the undo button. The UI7 test passed as the game did not error when simultaneously pressing the coloured buttons. UI8 tested for a feature not yet implemented, thus we had to adjust the Mastermind game board to accommodate for the new button. After adding the undo button, UI8 was tested and passed. All other functional tests for user input, menu navigation, game environment, and game logic passed.

5.2 Look and Feel Requirements Evaluation

Changes were made to the aesthetics of the menu and game board screen due to test LF1. Approximately 20 people were shown the initial demonstration of the game (Rev 0) and were to figure out how to play on their own, and comment on their experience using the app. Almost all users could play without any assistance, but many commented on the need for better application visuals. Our team redesigned the entire menu page, and change the background appearance of the application to resemble a fresh installment of the Mastermind from the 1970's.

5.3 Usability and Humanity Evaluation

There have been no changes made due to tests corresponding to the usability and humanity requirements test cases.

5.4 Performance Evaluation

There have been no changes made due to tests corresponding to the performance requirements test cases.

5.5 Operation and Environmental Evaluation

Changes were made to application to accommodate different screen sizes including phones and tablets. Although test OE1 does not specifically require us to use more than 1 type of emulator, we decided to experiment with emulators of different screen sizes to see how it would affect the appearance of our application. To ensure that the components of the application were not distorted we needed to align the components correctly so that they would appear in the correct position relative to each screen.

6 Automated Testing

Automated testing was performed throughout the development process to ensure that all versions of the mastermind worked as intended. Automated testing was done using Flutter's in-built feature which runs each unit test, widget test, and integration test when the "Run button" is pressed. Unit testing was done to insure that the game logic was not changed as in flutter the back-end and front-end code is intertwined. Widget testing was done to ensure that each widget was functional with its built in features. Finally, integrating testing was done to make sure that the system works when combined and that the app is functional on both ios and android.

7 Trace to Requirements

This section shows the Traceability Matrix between the test cases and their corresponding requirements.

Test Case	Req.
UI1	REQ2, REQ8
UI2	REQ3, REQ8
UI3	REQ4, REQ8
UI4	REQ5, REQ8
UI5	REQ6, REQ8
UI6	REQ7, REQ8
UI7	REQ8
UI8	REQ14
UI9	REQ9
UI10	REQ10
UI11	REQ9, OE4, OE2
GE1	REQ9, OE4, OE2
GE2	OE4, OE2
GE3	OE4, OE2
GL1	REQ11, REQ12, REQ8
GL2	REQ12, REQ8
GL3	REQ10, REQ11, REQ12
GL4	REQ13
LF1	LF1
UHR1	UH1
UHR2	UH7
P1	PR1
P2	PR3, PR4
P3	PR8, PR7, PR3
OE1	REQ1, OE1, OE2

Table 2: Trace Between Test Cases and Requirements

8 Trace to Modules

This section shows the Traceability Matrix from the Test cases to the Modules outlined in the MIS and MG. For further reference to the modules please see the MG and MIS documents.

For reference the modules are as follows:

- M1: Android Emulator

- M2: Game Board Module
- M3: Menu Module
- M4: Button Module
- M5: Game Board Controller
- M6: Types Module

Test Case	**Modules**
UI1	M2, M4, M6
UI2	M2, M4, M6
UI3	M2, M4, M6
UI4	M2, M4, M6
UI5	M2, M4, M6
UI6	M2, M4, M6
UI7	M4, M2, M5
UI8	M2, M5
UI9	M1, M3
UI10	M2, M3
UI11	M2, M3
GE1	M1, M3
GE2	M1, M3
GE3	M2
GL1	M5
GL2	M5
GL3	M3, M5
GL4	M3, M5
LF1	M2
UHR1	M1, M2, M3
UHR2	M1, M2, M3
P1	M1
P2	M1, M5
P3	M2, M4, M5, M6
OE1	M1

Table 3: Trace Between Test Cases and Requirements

9 Code Coverage Metrics

Trident Inc. has managed to get 98% statement and branch coverage through testing the Mastermind application. This number was generated by running the command ‘flutter test –coverage‘ in the command line in an existing flutter project with flutter tests and proper dependencies installed. Branch coverage is much more difficult to obtain than statement coverage due to the majority of the branches being dependent on a unique sequence of user inputs. The resulting code coverage metrics give the development team confidence in their testing suite, and that they have amply tested the implementation.