SE 3XA3: Test Report
Test Report

Team 25, N.L.E
Zihao Chen and chenz87
Ruoyuan Liu and liur19
Gundeep Kanwal and kanwalg

Decemer 6, 2017

Contents

List of Tables

List of Figures

This document is a specific test report corresponding to the test plan of the project Spiritual Jumper.

1 Functional Requirements Evaluation

FR1: FR1 is validated in two parts. When Score Testing(refer to Test Plan 3.1.2 for more detail) is performed over the game, the score should perform correctly to have the test passed for scoreboard. Secondly, when manual testing is performed across the testing process, the score never appears to have abnormal behavior. From the perspective of source code, the score is generated based on the highest height the character with a particular factor to make the score fancier. The requirement is already satisfied even before the implementation of test.

FR2: FR2, shares some similarity as FR1, validates by performing Score Testing over the game. Other than the manual testing(manually check the scoreboard everytime after a game-over), unit testing using JUnit is implemented to make sure that the scoreboard is in the correct state.

FR3: FR3 is tested manually by Area of Testing 1 and Area of Testing 2(see Test plan 3.1.1 and 3.1.2 for more detail). Every mouse movement and keyboard input while manual testing is corresponding to the Area of Testing 1 and 2. One of the issue jumped out during the testing process is that when the game is in the menu state, the mouse hover is not responded correctly by the game. For example, when the mouse is on the icon LEADERBOARD, the correct response should be the changing in menu background to highlight LEADERBOARD option. However, the game stuck at the initial menu and would not change. The issue was resolved after we navigated the part of code where monitors mouse position and found a bug that would cause game not reading mouse position correctly

Table 1: **Revision History**

Date	**Version**	**Notes**
December 6th	0.0	Document Skeleton created
December 6th	1.0	Document Finished

FR4: FR4 should be satisfied during implementation process, but when we performing other tests, the music would not stop after we press escape button to exit to main menu. Although the escape button is not a functional keyboard input in our game, it is set to escape game state by system default. We later fixed the issue by override the escape button functionality. FR5: FR5 is considered to be one of the most challenging requirement that we perform test on. As in the test plan, the platform test is tested by Run Testing. First of all, some of the platform only appears in high difficulty level, which lead us to temporarily changed game difficulty to level 5(the most challenging one) at the start of our game. The up arrow key was implemented to have the character jump infinitely high when pressed to test different platform without game-over. The function of different platform is tested to be in right behavior after hours of testing.

FR6: While performing testing on FR5, the spring function is tested as well. No significant error observed.

FR7: FR7 is considered validated as a result of that our game compiles successfully.

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel

Description: This test is graphic measurments of user interface and in-game environment. It is important to make appropriate and appealing graph aiming our main users group, who are in age between 14-18.

Test: - Pass

Results: According to the survey, 75% of the users were fond of our graphics design. The develop team believes the graphic satisfied the requirements and require no further change.

2.2 Usability

Description: This test is used to check easiness in terms of learning the game, and also check if the game control is smooth enough to play. The goal is all users can learn how to play under 5 minutes.

Test: - Pass

Result: 83% of the testers completed the task within the given time frame surpassing the requirement of 80%.

2.3 Performance

Description: This test is time measurement for the program performance. Our game is expected to start functioning in 1 second after compiling.

Test: - Pass

Result: 95% Our game can run in each mode under 1 second, it is not slow while performing every each functions.

2.4 Operational and Environmental Requirements

Description: This test is portability measurement of the system

Test: - Fail

Result: Our game cannot run outside of IDE environment due to the characteristics of Java Applet.

2.5 Maintainability and Support Requirements

Description: This test is measurement of easiness of update or other further change to the system.

Test: - Pass

Result: Our game is compatible with the change we made.

2.6 Security Requirements

Description: Whenever the system is retrieving data, it will ask for authorization before any further action.

Test: - Pass

Result: Our game does not retrieve data from users.

2.7 Cultural Requirements

Description: The content and purpose of this game should not cause any ambiguity or offence to various culture of users.

Test: - Pass

Result: Our game does not have any discrimination or causing any ambiguity to any group.

2.8 Legal Requirements

Description: The content and purpose of this game should not violate any law or protocol.

Test: - Pass

Result: Our game does not violate any law or protocol.

2.9 Health and Safety Requirements

Description: This test is making sure there is no explicit and inappropriate content for users groups.

Test: - Pass

Result: Our game does not contain any explicit or inappropriate content.

3 Comparison to Existing Implementation

The existing implementation does not have a 'theme'. Our project offer a space theme including the space background picture and background music.

4 Unit Testing

In terms of unit testing, the team used the Junit test classes within eclipse in order to carry out the unit testing which were mentioned in the test plan. The leader board was tested by unit testing the readscores method within a Junit Test Class named LeaderboardTest.java. Within that class, the method was tested by inserting 6 different scores within the array list used by the code. The list of scores are sorted in order and are then tested to check if the scores are in fact in order. The test case is successful, highlighting the accuracy of the leaderboard. Edge cases such as equaivalent scores, and zero scores were handled.

5 Changes Due to Testing

In the section functional requirements evaluation, it gives out a few examples on how the testing impact our source code and decisions. This section will continue discuss about changes we made after and during testing.
In the origina

6 Trace to Requirements

Let AOT as numbered in Area of Testing in Test Plan, then AOT1.1 is Mouse Testing. Also, for non functional testing, Area of Testing would start with N as NAOT.

Table 2: **Test-Requirement Trace**

File	**Related Requirement(s)**
AOT1.1	FR.3, FR.4
AOT1.2	FR.3
AOT2.1	FR.2, FR.3, FR.4
AOT2.2	FR.2, FR.3, FR.4
AOT2.3	FR.1, FR.2
AOT3.1	FR.1, FR.3, FR.4, FR.5, FR.6, FR.7
AOT3.2	FR.5, FR.9, FR.26
AOT3.3	FR.3
AOT3.4	FR.1
NAOT1	NFR.1, NFR.2, NFR.3
NAOT2	NFR.4
NAOT3	NFR.4, NFR.5
NAOT4	NFR.7,NFR.8
NAOT5	NFR.6

7 Trace to Modules

Table 3: **Test-Module Trace**

File	**Related Requirement(s)**
AOT1.1	
AOT1.2	
AOT2.1	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31
AOT2.2	FR.5, FR.31
AOT2.3	FR.5
AOT3.1	FR.5, FR.13, FR.14, FR.15, FR.25, FR.30 FR.31
AOT3.2	FR.5, FR.9, FR.26
AOT3.3	FR.16-19
AOT3.4	FR.16-19
NAOT1	Runs Tests
NAOT2	FR.37, FR.38, FR.39
NAOT3	FR.35-39
NAOT4	FR.9-15, FR.26-34, NFR.5
NAOT5	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31

8 Code Coverage Metrics

As in the table in the section above, Trace Tables provided a specific view on how the test covers the functional and non-functional requirements. It also