Software Testing Report

Group 8 Members

Zac Challis

Viktor Atta-Darkua

Dandi Harmanto

Harsh Mohan

Danny Burrows

Test Plan

We plan to attempt to make tests to cover all the requirements from our requirements deliverable. These tests will be run after each code update, which is frequent due to our use of continuous integration, to make sure that it still works and meets our project requirements and allow us to find any bugs our updated code has when tests fail.

We are going to perform black-box and white-box testing on our code. Our code will include a mixture of Unit tests, integration tests, system tests and manual tests as needed:

- Unit tests test single 'units' of code and are fast to write but cannot test certain bugs and complicated features
- Integration tests test multiple system components and their interaction together and can solve more complex problems than unit tests
- System tests test the whole system, are slow and difficult to write and slow to run but most of our project's requirements can be tested using them
- Manual tests do not need to be coded and are simple to implement but can take longer to run than all previous test types and cannot be automated
- Unit tests take the least time but have the greatest limits while system and manual tests take the most time but can test all of our project requirements. We are using a mixture of testing approaches because:
    - It is a waste of time to use System or manual testing for things that can be solved using Unit tests
    - If Unit tests cannot test a certain feature of our project due to it being too complicated, we can move up to integration, system and manual testing as required.
- Black-box because the code originates from another team and we don't have perfect understanding of it, so it's easier to just look at inputs and outputs
    - We will include a traceability matrix to make sure no requirements are missed and attempt to reduce redundancy
- Both black & white box because they are complementary

JUnit is a common Java testing framework that can enable us to automate our tests. This will allow us to run them quickly after every update to our code..

When implementing black box testing, we will try to cover all of our requirements. At each requirement we will see if we need to split the test into multiple partitions and boundaries depending on the possible outputs; if we do, then we will test multiple inputs leading to their respective equivalence classes and boundaries to make sure our program has no bugs

Unit Testing

Unit testing was adopted to accomplish our white box testing. Prior to the implementation of any new features, unit tests were written and run to establish all currently satisfied requirements. Succeeding this, unit tests were used to complete our integration and system testing of new game features, as the communication between the various interacting components and managers was highly traceable, the application was able to be tested as a whole.

Tests were written once a testable component was completed and considered to be successful from a functional perspective, in order to guarantee new code complied with the code base and unmodified code were not affected by changes. It was critical tests were written for any functionality where multiple components are called and interact with one another, such as : a user purchasing a power-up. These tests also benefited from a large amount of testing data. This measure ensured the corresponding code was reliable, simplifying future debugging. Testing complied with our method of continuous integration, automatically running on every push to the repository.

Out of 192 unit tests covering all testable lines of code, there is a 100% pass rate showing a complete satisfaction of the requirements capable of creating tests for. These tests covered 67% of all classes and 45% of all lines. These tests are documented in a testing table where each record encompasses : the name of the test method, the functional requirements associated with it, objective of the test and result.

Due to the limitation of testing in a headless application, a large number of lines limited to rendering had to be omitted, as tests relating to them would be impossible to write and unable to run.

Link to the unit testing table : Unit testing table
Code coverage table : Code coverage table
URL to testing report : Testing Report

Manual Testing

Completeness of tests circumscribed the fulfilment of all user requirements To make certain our testing was as complete as possible, the traceability matrix was utilised to identify the requirements unable to be automatedly unit tested; where unit tests were considered unfeasible or strenuous, appropriate manual black box testing was elicited.

Link to traceability table : Traceability matrix

A wide range of manual play tests were run whenever a component was completed to certify there were not problems from the users perspective, that unit tests were unable to identify; these would typically be graphic based such as for responding user input. These tests were repeated multiple times with a wide range of testing data, examining acceptable, invalid and

boundary values. Manual testing was the simpler and even exclusive method of testing non functional requirements such as the testing of colourblind compatibility.

Out of 40 manual play tests, there is a 100% pass rate showing complete satisfaction of the tests unaccounted for by unit testing. These tests were documented in a testing table where each record encompassed : the name of the test method, requirements associated with it, the objective of the test, test data, expected result and the actual result. Some attributes were unable to be assigned test data such as the testing of non-functional requirements.

Link to the manual testing tabe : Manual testing table

Evaluation

Through our 100% test pass rate, we can make certain all requirements have been fulfilled with only certain exceptions. Such as UR_FAMILY_FRIENDLY, which cannot be tested for, due to its subjective nature. Attributable to the time constraint the team experienced, it was not feasible to test every possible scenario bringing into question the completeness of all our tests.