

Test Methods

Our tests are implemented via JUnit and associated libraries and are clearly indicated within the code, as any class ending with “test” contains tests only for its related main class. RoboticonTest contains all the tests for the Roboticon class. This structure allows for easy tracing of errors both in your Java IDE and continuous integration platform of choice (We have been using IntelliJ and TravisCI). Our tests were implemented such that:

- Every method (That is not a Setter or a Getter or part of the UI structure) has an associated test, where appropriate methods may have Valid, Invalid and Boundary value tests to ensure they operate and respond correctly.
- All the requirements expected at this point in the project are tested, be this with separate tests or the tests mentioned above.

Our Tests will be considered at an acceptable state when all unit tests are passing and all other tests of implemented functions are passing. Some usability tests are allowed to fail, provided the feature they relate to was not essential for Assessment 2. Any such failures should be documented in the report with details as to why they fail and/or what is required to fix them.

Functional Testing

Unit Testing, All methods and classes were unit tested as mentioned above. Unit tests are located in the classes ending with ‘Test’. These unit tests will always be applicable to the project as it develops and can be modified as necessary if the architecture develops.

Integration Testing, Some of the unit tests required multiple classes to be performed, this makes them essentially unit tests **and** integration tests due to the fact they’re testing multiple classes at once whilst checking the correctness of a single method. Dedicated integration tests separate from the unit test directory were deemed unnecessary as the system testing is the next logical step up for our use case.

System Testing, Whilst not fully implemented as true black box system testing, the system is regularly deployed to check the UI is displaying correctly. And whilst not formal or complete, the system is deployed on a regular basis.

Acceptance Testing, Not included as of yet due to the stage of development we currently sit at.

Non-Functional Testing

Performance, As with system testing, basic performance testing has informally occurred throughout development under the expected use case.

Usability, A small test suite drawn from the requirements will be answered qualitatively and located on the [website](#)

Compatibility, throughout development Travis CI has deployed the tests in a linux environment, and our team members have been informally deploying on both macOS and Windows environments.

The full test suite is provided alongside the the code and is clearly distinct thanks to our code structure.

Documentation for the Unit Tests is provided alongside each test. Essentially each test maps to a corresponding function it is testing the correctness of so test<Function name> in the class test<Class name> is testing that function in that class.

Sources:

All Functional and Non-Functional Testing definitions were sourced from:

<https://www.inflectra.com/Ideas/Topic/Testing-Methodologies.aspx>

Test Report

The Unit tests and Integration test results can be seen [here](#)

We ran 22 Unit tests, the majority being true unit tests, some being modified unit tests which tested a single method but required other objects to be correct to function properly. These tests were required to have 100% pass rate, otherwise internal logic would be broken potentially jeopardising the game.

Qualitative test link: <https://jm179796.github.io/SEPR/testing.html>

The qualitative test document only pertains to requirements fully or partially implemented at this point. The document will require being expanded for future testing of the system. In regards to the document, the following questions and therefore requirements did not pass: 5,14, 21, 22, 26, 29. Below we have detailed why the tests failed and what steps future developers should take to ensure they pass:

Not UI Supported yet:

5. Can you see the values of a tile you own after you've claimed it?

Whilst Tiles have private values for resources and methods to access them, The UI does not yet update to display these values after the Tile has been acquired by a Player. Future teams should consider maybe using the tooltip or the Tile display in the top right to display these values.

14. Is it possible to gamble to win or lose money?

The gamble method exists in Market however it was not requested for Assessment 2 and therefore the UI has not been included. In accordance with the requirements from the User, future teams should consider creating a separate screen from the main UI to allow gambling and not crowd the main game UI.

21. Does the game select the winner based on the highest score?

This usability test failed due to the fact that whilst the Game calculates the winner at the end of the game, the UI does not actually render it. But the winner is calculated in the final round of play and the output is sent to the console. A UI screen should be added to declare the end of the game.

22. Is it possible to see information about tiles you own, such as their resource counts and any roboticon assigned to the tile?

The roboticons are displayed on tiles but without their upgrade levels and without the tile's resource counts. The roboticon upgrades and tile resources have methods associated with them that the UI has not yet implemented.

Not Implemented:

26. Do the values on tiles represent vaguely the image on top?

Currently all tiles are implemented to have the same yields, future teams should declare the tiles to be representative of the map content on that tile .

29. Is the information described in requirement 17 clearly visible?

17 a. It will be possible to get the following information about the tile:

- i. The owner of the tile; The type of resources that can be gained from it; what kind of Roboticon is working on it (if one's working on the tile at all); options to assign Roboticons from each tile and an option to upgrade the Roboticon working on the tile*

Whilst the owner and Roboticon presence can be seen, options to assign and upgrade Roboticons. The Roboticon upgrade status and the Tile resource values are not implemented. Future teams should create a graphical method of displaying the roboticon level perhaps on the icon or as a tooltip on the icon and the Tile resource values should be considered as above.

Alongside this test suite it should be noted that some requirements were implicitly met, we will list them here so as to avoid any confusion and state why we believe them to be met:

2.a) The [map](#) displayed in the game meets the specification set

A more clear and visual representation of the state of the project and it's requirements can be found [here](#):

<https://jm179796.github.io/SEPR/Images/RequirementsMet.pdf>