Testing

Old Team:

"Lucky" Team 13

Team 13

Yuxin Wu
Bailey Findlay
Elizabeth Edwards
Chenxi Wu
Alex McRobie
Lawrence Li

New Team:
Team 14
Joshua Beaven
Calum Feenan
Harry Hillman
Jonathon Snelgrove
Omar Alhosani
Rayan Butt

PART A: Summary of testing methods and approaches

To begin with, we made it clear that the code would have to be refactored to enable testing if the code that we had inherited was not testable. Following on from this, the team came to a conclusion on which testing method we should use and we decided on the Test pyramid method (which was discovered during our testing research). We wanted to focus on using unit testing where possible as this would allow us to use the most amount of automated testing, this came with the downside of placing the importance of tests and understanding what might be a reasonable test. However, if the feature needed to be tested was reliant on a human player, or the feature was too complex then we would instead have to rely on manual testing to ensure that the feature is still looked at. These manual tests would cover things like rendering of the game, where even though an assertTrue gave us true that the game was rendered, we the humans had to ensure that we could see the game on our monitors.

Having decided our testing structure, we moved on to discussing: which parts of the game needed testing, which method of testing we would use for said feature (Unit, Integration, E2E), what are expected inputs and outputs were and a way of comparing those with the actual input/outputs. Many of these decisions were made by looking at our requirements and seeing what testing would be suitable. We noted these down into a Testing Plan[2]. This testing plan would change as the game progressed in development as new parts of code/the implemented game would require tests.

Black-box testing and White-box testing were also critical in ensuring that our testing ran efficiently across the team. The black-box testing were tests that were based on requirements, and these were important for the people that didn't work on the code/havn't been filled in on how the code works. As the requirements were elicited beforehand, everyone could look at them and contribute. White-box testing was used in unison with black-box testing and allowed the people who had worked in the code/were filled in on the code were able to use the information about the structure of the code and create code from it.

Part B Report on Tests:
In total we conducted 8 automatic testing categories, where most contained multiple tests. These categories were:

- Assets
- Display
- HUD
- Ingredients
- Listener
- Orders
- PreGame Selection
- Recipes

Automatic Testing:
To complete our automatic testing we used Intellij built in coverage tool, and GitHub. Whenever a new test was completed we would ensure to run these tests on the newest version of the game that was pushed onto main. This kept us up to date with checking if new code or tests caused the game to crash/behave incorrectly or stop tests from passing. If an error did arise from the new test, this would be raised from either the testing team to the implementation team, or from the implementation team to the testing team, depending on which side the error took place. We implemented the use of a testing matrix[1] to show what our automatic tests had in relation to the testing matrix.

Manual Testing:
We conducted manual testing within the game environment. Again we used the testing matrix[1] to show the manual testing results and the requirements that they hit. The rightmost column on the testing matrix shows how the manual tests were carried out.

Test Results:
The testing matrix[1] displays the test results best, although we tried to hit every requirement through testing, some were missed, this is displayed through the number underneath the requirements. If the cell beneath the requirement is empty, that means none of our tests were able to hit this requirement. Many of our tests did overlap in terms of requirement hit, and that is also displayed in the testing matrix.

Testing Completeness:
Although all of our automatic tests passed, a number of the tests we initially set out to complete automatically were moved to testing manually due to the fact that the automatic tests conflicted with libgdx/we were not able to mock and set up an environment to configure them to work. Because of this, we made sure that every automatic test that we failed to set up was still tested through manual testing, and although this was less efficient than automatic testing, it still ensured that we hit all our tests.

In terms of the tests that we did set up correctly, we wanted to make sure that these tests did indeed test something about the code, and were not set up to pass. The testers and implementers would therefore, check the tests regularly to ensure that they are actually testing something.

As stated previously, we used intelliJ’s built in coverage tool to enable us to view the test coverage. When this tool was run on the io.tests folder (where the tests are located) the report came back with 60% class coverage, 45% method coverage, and 45% line coverage . As a team we viewed this as slightly low for test coverage, however due to the intricacy of the work, we understood that achieving a higher meaning coverage was beyond the automatic testing scope. However, due to our testing also including manual testing we believe that the 45% line coverage is not showing our coverage in full picture, and with all of our manual and automatic testing, we have covered a lot more of the requirements and code functions.

<u>Part C: Testing Material</u>
Link to Testing
Results:https://eng1team14assessment2.github.io/TestImages/TestResults.png
Link to Coverage Report:
https://eng1team14assessment2.github.io/TestImages/TestCoverage.png
Link to Test Matrix (Manual Tests are found at the bottom of the testing matrix):
https://eng1team14assessment2.github.io/TestImages/TestMatrix.png

<u>Appendix</u>
[1] Test Matrix:
https://docs.google.com/spreadsheets/d/1pg8gWZXW0eiTQ6ZKMQBdQKwz1In-4hFXL0LyMqQh0b4/edit#gid=0
[2] Test plan:
https://docs.google.com/document/d/1e8P7vwczdA-IjJ2W-wsdc3iAcWeTb-SZAmppihCxjYg/edit?usp=sharing