# Software Testing Report

*Assessment 2*


*Group 3:*

Ben Howard <bh1219@york.ac.uk>
Cai Hughes <cabh500@york.ac.uk>
Harry Richardson <hr1040@york.ac.uk>
Ivan Ndahiro <in597@york.ac.uk>
James Sutton <jis509@york.ac.uk>
Yuzhao Liu <yl5164@york.ac.uk>

*Group 6:*

Igor Smollinski <is942@york.ac.uk>
Pranshu Dhungana <pd861@york.ac.uk>
Zack Tyler-Kyle <ztk500@york.ac.uk>
Phoebe Russell <pbr508@york.ac.uk>
Sanjna Srinivasan <ss3264@york.ac.uk>
Sam Savery <sgs527@york.ac.uk>
Ewan Hutcheson <@eh1776york.ac.uk>

Testing the game has been a major topic of discussion throughout Assessment 2. We have a key responsibility to test the game to ensure that it functions as it should and more importantly, meets the requirements mapped out for us in the requirements section. We decided to test the game using autonomous unit tests. For each Java class we created a test class that has a test for each method in the original class, these tests certify that the methods function as they should and return the correct values.
We took the approach of autonomous unit testing as it allowed us to develop tests alongside new classes and methods during implementation or after, allowing us to view individual methods and how they should run. To make sure our implementation works, doing it in this modular fashion allows for ease in debugging if a test fails we can identify the exact method that is not functioning as intended and how that affects the game as a whole by tracking that method/class used throughout the project to then fix the class and/or method. For the autonomous side of testing we have created numerous test classes that test different classes through the game and the methods inside those classes.

**Autonomous Unit Testing Results and Links to Requirements**

| Tests | What is being tested | Test result | Related Requirement | Comments |
|---|---|---|---|---|
| CookAssetTests | Cook assets are present and accessible | All tests passed | FR_COOKS | Tests the cook assets exist |
| CustAssetTests | Customer assets are present and accessible | All tests passed | FR_NUMBER_OF_CUSTOMERS | Tests the customer assets exist |
| GameAudioAsset Tests | Game audio assets are present and accessible | All tests passed | n/a | Tests the game audio assets exist |
| MusicAssetTests | Music assets are present and accessible | All tests passed | n/a | Tests that the music asset exits |
| CookTests | Tests cook class methods | All tests passed | FR_COOKS<br>FR_COOK_CONTROLLER<br>FR_COOK_ACTIONS<br>UR_CONTROL_SYSTEM<br>UR_ITEMS | This test class makes sure the |
| CustomerTests | Tests customer class methods | All tests passed | FR_NUMBER_OF_CUSTOMERS<br>UR_DEMANDS<br>FR_DEMANDS | Tests the flow of customers and how long they wait |
| CustomerControllerTests | Tests customer controller class tests | All tests passed | FR_NUMBER_OF_CUSTOMERS<br>UR_DEMANDS<br>FR_DEMANDS | Tests that customers correctly load and save |
| CollisionTileTests | Tests collision tile | All tests passed | n/a | Checks that the collision tiles rectangle is drawn properly |
| PowerUnitTests | Tests power unit | All tests | UR_POWER_UPS | Checks that the |

| | | | | |
|---|---|---|---|---|
| | class methods | passed | UR_REPUTATION_POINTS<br>FR_INCREASE_SPEED<br>FR_REPUTATION_DECREMENT<br>FR_POWERUP_GENERATION<br>FR_ADD_POINTS<br>FR_INCREASE_REPUTATION_POINT | correct powerup is selected and is visible |
| OrderCardTests | Tests order card class and methods | All tests passed | UR_DEMANDS<br>FR_ITEM_DESCRIPTIONS | Checks if order cars are shown on screen with the correct size and information |
| BakingStationTests | Tests the baking station class and it's methods | All tests failed | FR_PREPARE_STAGE<br>UR_COOKING_STATIONS<br>UR_ITEM_STATION<br>FR_ACTION | Functionality cannot be tested due to some graphics components not being mockable |
| CookingStationTests | Tests the cooking station class and it's methods | All tests failed | FR_PREPARE_STAGE<br>UR_COOKING_STATIONS<br>UR_ITEM_STATION<br>FR_ACTION | Functionality cannot be tested due to some graphics components not being mockable |
| CuttingStationTests | Tests the cutting station class and it's methods | All tests failed | FR_PREPARE_STAGE<br>UR_COOKING_STATIONS<br>UR_ITEM_STATION<br>FR_ACTION | Functionality cannot be tested due to some graphics components not being mockable |
| OvenStationTests | Tests the oven station class and it's methods | All tests failed | FR_PREPARE_STAGE<br>UR_COOKING_STATIONS<br>UR_ITEM_STATION<br>FR_ACTION | Functionality cannot be tested due to some graphics components not being mockable |
| PrepStationTests | Tests the prep station class and it's methods | All tests failed | FR_PREPARE_STAGE<br>UR_COOKING_STATIONS<br>UR_ITEM_STATION<br>FR_ACTION | Test cannot instantiate ingredients as it is a game screen render and so the test fails |
| StationTests | Tests the station class and it's methods | All tests failed | FR_PREPARE_STAGE<br>UR_ITEM_STATION<br>UR_COOKING_STATIONS<br>UR_PANTRY_STATION<br>UR_COUNTER<br>FR_ACTION | Test cannot instantiate ingredients as it is a game screen render and so the test fails |

| | | | | |
|---|---|---|---|---|
| GameScreenTests | Tests saving | All tests failed | UR_SAVE_STATE | Test cannot instantiate (render) game screen and so the test fails |
| IngredientTests | Tests cook, slice and flip funtionality. Tests if a powerup is active when using an ingredient. Tests equality of 2 ingredients. | All tests failed | FR_SALAD<br>FR_PIZZA<br>FR_BURGER<br>FR_PIZZA<br>FR_JACKETPOTATO | Test cannot instantiate ingredients as it is a game screen render and so the test fails |
| RecipeTests | Tests if the current held stack is equal to a recipe in order to form the final food item. | All tests failed | FR_SALAD<br>FR_PIZZA<br>FR_BURGER<br>FR_PIZZA<br>FR_JACKETPOTATO | Test cannot instantiate ingredients as it is a game screen render and so the test fails |

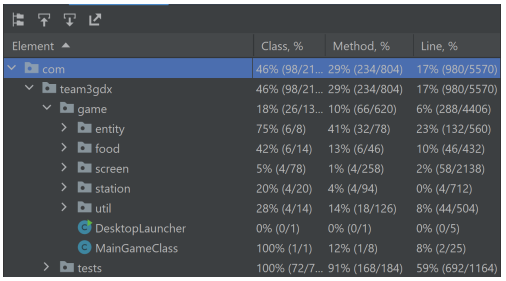
As you can see a number of test cases have failed in the autonomous unit testing. This functionality cannot be tested as it involves graphics components which are unable to be mocked using Mockito (and GdxTestRunner.java). As a response to this we have opted to do some manual testing. This allows for implementation to be done a lot easier and testing to work to still go ahead. We have created a manual testing table below to identify what each test is testing and what requirement it relates to.

**Manual Testing Results and Links to Requirements**

| Tests/What is being tested | Test result | Related Requirement | Comments | Related Figure |
|---|---|---|---|---|
| Tests that recipe is formed from the correct ingredients | Test passed - Correctly formed ingredient | UR_RECIPES | Testing in game that recipes are formed correctly from the correct ingredients | *Fig.2, Fig.3* |
| Test that button for scenario mode is present and loads scenario mode when clicked | Test passed - option to go into scenario mode and functions | UR_SCENARIO_MODE | Testing in game that there is an option to play scenario mode and that it goes into it. | *Fig.4* |
| Test that button for endless mode is present and loads endless mode when clicked | Test passed - option to go into endless mode and functions | UR_ENDLESS_MODE | Testing in game that there is an option to play endless mode and that it goes into it. | *Fig.4* |
| Test that the leaderboard is present and can be viewed and added to | Test passed - option to view and save to leaderboard | UR_LEADERBOARD | Testing in game that there is a leaderboard and it can be changed by playing the game and setting high scores. | *Fig.5* |
| Test that there is a main game screen | Test passed - both the ability | UR_ACCESSIBILITY | Testing in game that there is the ability to use | *Fig.6, Fig.7* |

| | | | | |
|---|---|---|---|---|
| and a menu while playing | to use the main menu and in game menu are present | | a main menu and an in game menu. | |
| Test play the game to earn enough points to unlock new stations | Test passed - two stations were able to be unlocked with enough points | UR_LOCKED_ RESOURCES | Testing in game to unlock stations by earning enough points and unlocking them | *Fig.8, Fig.9* |
| Test that has two different computers with different OSs running the game | Test passed - runs on both Windows and MacOS | UR_PLATFOR M | Testing in person to show that both OSs can run the game | *Fig.10, Fig.11* |

## Coverage




## Figure Section:

Fig.1 - Showing that the flip option is available and works.

Fig.2 - showing that items combine as a recipe to make the final product.

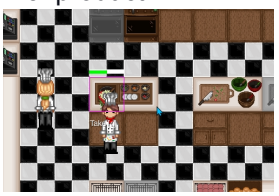


Fig.3 - Show that items combine as a recipe to make the final product.

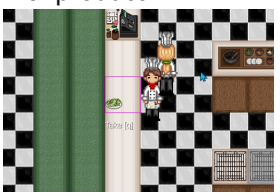


Fig.4 - Endless mode and Scenario mode.

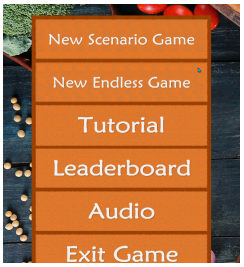



Fig.5 - Leaderboard.


Fig.6 - Showing the main menu is present.


Fig.7 - showing there is an in game menu.

Fig.8 - Shows a locked station which has the ability to be unlocked.

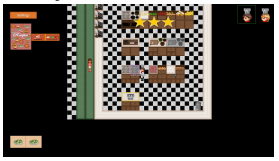


Fig.9 - Shows the station now unlocked.

Fig.10 - Shows the game running on a MacBook Pro.


Fig.11 - Shows the game being played on a Windows machine (as shown by windows task manager).

**C) URL:** [https://neves6.github.io/PiazzaPanic2/deliverables/TestResults.html](https://neves6.github.io/PiazzaPanic2/deliverables/TestResults.html)