Trashmunchers Testing Strategy

Malik Besta, Ellie Keleman, Vignesh Mohanarajan,
Owen Lee, Scarlett Parker, William Liversridge

Contents

Version History

Version	Purpose	Author	Date
0.1	Draft	Malik Besta & Vignesh Mohanarajan	09/03/2023
0.2	Draft	Vignesh Mohanarajan	14/03/2023
0.3	Draft	Malik Besta	21/03/2023

Test Objectives and Scope

For our testing strategy we incorporate

1. unit tests,
2. API tests,
3. automation of tests,
4. cross browser tests,
5. security tests,
6. integration tests,
7. acceptance testing,
8. and system tests.

However, we will not be covering time scales, effort, cost, and performance tests as they are not crucial to the minimum viable product. They would be nice to have, but they aren't extremely important.

To guarantee the success of our project, we need to ensure that our project makes these objectives so that it is as close as possible to a minimum viable product. Our testing objectives are as follows:

1) Maximise code coverage in our tests - Ensures all aspects of the code that should be tested are tested.
2) Support continuous deployment – Ensures that the product can be built and successfully deployed when changes are pushed to the repository.
3) Identify the types of testing to be used.
4) Guarantee the robustness of the programs.
5) Ensure the API endpoints function as expected.

By achieving these objectives, we should be able to attain the benefits of doing so. Below is a list of benefits from reaching our goals.

i. Handle more exception cases, mitigating risk.
ii. Provide a superior user experience, increasing user experience.
iii. Identification of bugs and logic errors earlier, reducing time spent fixing it down the line.
iv. Encourages communication between team members so they understand how each aspect of the code works.
v. Easier allocation of who completed which task.

Types of testing

As stated previously, we are utilising many different methods in order to hit our objectives. Below is an outline of how, when, and who will be carrying out each test.

Testing Type	When	How	Who
Unit Tests	Implemented on 11th March 2023	tests.py in every Django app that includes tests for all the functions in its viewset(s).py list.	Malik, Vignesh
API Tests	Implemented on 11th March 2023	Unit tests written in tests.py should correspond to the APIs in the viewsets, to ensure they are functioning as intended	Malik, Vignesh
Automation of Tests	Implemented on 7th March 2023	Using GitHub actions, we can confirm whether the most recent commit can be	Ellie, Vignesh, Owen

		built to ensure that it works.	
Cross Browser Tests	Ongoing	Viewing the webpage on many different devices (computer, laptop, mobile, etc.) and seeing if the website functions the same across all devices.	Scarlett, Owen, Ellie, William
Security Tests	Ongoing	Ensuring that the website doesn't allow weak passwords and relays this information the user.	Scarlett, Vignesh
Integration Tests	Ongoing	Checking to see if actions carried out on one webpage/API endpoint is reflected throughout the others that use elements from the original webpage/API endpoint.	Scarlett, Vignesh, Ellie, Owen, William, Malik
System Tests	Ongoing	Checks to see if the system is fully operational and working as intended, with a test player, gamekeeper and developer.	Scarlett, Vignesh, Ellie, Owen, Malik, William

Acceptance Tests	Ongoing	Continuous consultation with the client, allowing them to view the current progress of the system and see where changes can be made to fit their requirements.	Scarlett, Vignesh, Ellie, Owen, Malik, William

Tracking and reporting issues/defects

Currently, we are looking into two different options.

Option 1 – Report Portal: Report Portal is an open-source online test reporting tool that keeps reports on the results of tests (e.g, who wrote it, what caused the test to fail, etc.). Whilst it may provide us with useful information and an ideal way to keep track of issues, there is the issue of training the whole team to use this tool. For a longer project, it would be more suitable. For now, it will only be considered unless plans change.

Option 2 – Trello Bug list: Easiest option out of the three. The team is already very familiar with the software and wouldn't need to be trained in using this as it is a requirement before joining this team. A drawback of doing so is that the list could become cluttered and would take too long to trace back issues to where problems are occurring.

As a team, we have decided to go with the Trello Bug List due to its simplicity and how it requires less time to for our team members to familiarise themselves with it. In addition to this, it could work in accordance with the Validation done/doing lists on the Trello board as these three lists can act as a test report for who is carrying the test out and the result of it (failure, success), highlighting the Trello board's advantages to Report Portal.

Test environments

Environment	Requirements
Local	Latest snapshot of the code taken from branch daily. Ensure you have installed latest dependencies from the pipfile. Run python manage.py test to carry out all Django app tests
Regression	Same as local. Rerun old variants of the code to ensure that the latest code is behaving as expected.

Testing Tools

Tool	Purpose
Django	Backend unit tests
Django Rest Framework	API unit tests
TBD	Security tests
TBD	UI automation

Risk Assessment

Currently, we do not have a tool to use for security testing. Because of this, there is a possibility of an attacker carrying out an SQL injection or exposing a XSS vulnerability. To ensure that this is addressed, we have encouraged our developers to stray away from string concatenation when adjusting the database. This is to ensure that an attacker cannot attach a payload to trigger the injection, leading to our project ultimately failing. If this does become a reality, we will investigate sanitising user input to prevent an attack from being possible.

Another risk that might occur is whether the webpage will display properly across many different devices. Users would be constrained to

playing our game on a single device format, preventing them from being able to use it on the move. A consequence of this would be that they would not be able to view the status of the map on the move, unless they have access to campus computers. Increased mobility of the product benefits the user as they will be able to view the game from the device of their choice, without having to worry about it feeling unsatisfying on their end. Enabling this is ensuring the CSS we have used allows for this to work and viewing the page on different devices. On the off chance we are unable to allow it to work across all devices, we will try to at least guarantee it works on mobile and computers/laptops so users can at least play on the move. This does, however, ultimately be disadvantageous to tablet users, so we will try to find ways to ensure they can use it in future updates.

Our last risk that we considered was user exploits. To ensure that players couldn't exploit the game mechanics, we looked into ways to prevent them from being able to, for example, send the same images multiple times to get points for their team and simulate their impact their recycling has on the environment. To prevent this, we have location authentication to ensure that the user was no more than 50m away from a recycling location when they submitted their image. In addition to this, we have developed a gamekeeper account that can approve user images to ensure that the same image hasn't been used multiple times, covering up this potential exploit. If a user finds a way around this (e.g., editing the image to show different bottles/larger amounts), we will reconsider how we approve images and reflect this on the website, so users are aware of the changes.