|  |
| --- |
| Four Walls Software |
| Test Summary & Defect Report |
| Landfill Labs Worker Service Prototype |

|  |
| --- |
| Husnain Ahmed (21308666)  1-8-2025 |

Contents

[Test Summary 3](#_Toc183187117)

[System Overview 4](#_Toc183187118)

[Testing Scope 4](#_Toc183187119)

[Test Environment & Tools 8](#_Toc183187120)

[Statistics 8](#_Toc183187121)

[Recommendations & Conclusion 9](#_Toc183187122)

[Defect Report 10](#_Toc183187123)

# Test Summary

[More generally, this document should serve as a guide, you may wish to change the headings/style to be more consistent with the Four Walls Software Branding Guidelines, see the link below [6G6Z0035\_1CWK30\_2024/Supplementary documents/Brand guidelines at main · samattwood9/6G6Z0035\_1CWK30\_2024](https://github.com/samattwood9/6G6Z0035_1CWK30_2024/tree/main/Supplementary%20documents/Brand%20guidelines). Additionally, you may wish to add sections and improve the presentation of this document (i.e., adding page numbers to make it easier to navigate). In short, don’t let this template constrain you in any way – it is here for guidance purposes]

This is a summary

## System Overview

## 

The Landfill Labs Waste Management System, developed by Four Walls Software for The Council, is a user-friendly program that will allow the council to be more efficient when managing and disposing of any recyclable waste from historic landfill sites.

It allows different waste management scenarios to be configured, model these scenarios accordingly, and output results that The Council can use to empty historic landfill sites.

**System Logic:**

* **RunModel**:
* This is the core method for modelling waste scenarios.
  + Returns key results: travel duration, process duration, and total duration.
* **‘EstimateWasteSplit’**:
* Estimates the split of waste based on total volume.
* If total waste is less than 1250 cubic meters:
  + 50% plastic/glass, 50% paper.
* If total waste is greater than 1250 cubic meters:
  + 30% plastic/glass, 50% paper, and 20% metallic.
* **‘FindViableCentres’**:
* Identifies viable recycling centres based on waste type and distance.
* For metallic waste: includes centres within 3 hours travel time.
* For no metallic/paper waste: only Alpha and Beta centres are viable.
* **‘FindOptimalCentre’**:
* Orders viable centres by proximity and priority.
* **Priority order:**
  + Nearest centres.
  + If multiple centres are equally near, the one with the highest generation (Gamma > Beta).
  + If generation is equal, the centre with fewer years of activity is prioritized.
* In case of ties, the optimal centre is chosen randomly.
* **‘CalculateTravelDuration’**:
* Calculates travel time based on the number of round trips needed for waste transport.
* Each truck can carry up to 20 cubic meters of waste per trip.
* **‘CalculateProcessDuration’**:
* Calculates processing time based on centre type and waste type.
* **Type Alpha:** 1 cubic meter per hour.
* **Type Beta:** 1.5 cubic meters per hour.
* **Type Gamma:**
  + 1.5 cubic meters per hour for plastic.
  + 2.0 cubic meters per hour for metallic.
  + 3.0 cubic meters per hour for paper.

Calculates travel and processing durations using specific formulas.

Orders recycling centres by priority, considering proximity, type, and operational age.

**Key features:**

* **The ability to model scenarios:**
  + The user can input data to create Historic Sites (Landfills) and Recycling centres.
  + The system uses this data to calculate the amount of time it will take to fill the recycling centre with waste and to process the waste.
  + It then outputs these times to the user as well as the total time.
* **Command Line Interface (CLI):**
  + The user can navigate through the CLI to interact with the system.
  + The CLI has a specific page (About page) to give the user information on how the system works and how to use it.
* **System constraints:**
  + Only 1 user at a time can currently use this system.
  + Only 1 transport truck is available at any time.
  + Each transport truck can only carry a maximum of 20 cubic metres of waste at a time.
* **Security:**
  + Only 1 user at a time can currently use this system as it is in a initial stage, this will be changed in the future and we will need to take steps to ensure that each users data remains secure and only accessible to them and them alone.
  + Data is not stored locally or on the cloud at this moment in time so it cannot be accessed my malicious third parties, in future iterations of this system we will need to ensure data privacy and protection by decentralising data and/or encrypting it.
  + The system follows the Single Responsibility Principle (SRP) and each function is responsible for a single task each.
  + The system will have to follow well established best security practices.
* **Error handling:**
  + The system is designed to fail gracefully in most cases and should prompt users to retry, proper messages should be displayed for invalid or missing inputs.
  + Numeric values aren't always formatted to 2 decimal places as they should be.
  + Some invalid scenarios result in exceptions or crashes.

## Testing Scope

The scope of testing for this project is designed to ensure that the system requirements are met and the developed features function as expected. The testing is focused on the key system logic outlined previously in this document and will be executed to verify the proper operation of the critical features.

**In-scope**

The following features and aspects of the system are considered in-scope for testing:

* **Worker Service**:
  + The system must be able to model waste scenarios.
  + Estimate waste splits and identify viable recycling centres.
  + Order recycling centres based on proximity or the highest waste generation.
  + Calculate travel duration and processing time for each selected recycling centre.
* **Security Criteria**:
  + Ensure that the system adheres to established best practices for security.
  + Protect against privacy breaches and ensure secure handling of user data.

**Out-of-scope**

The following aspects are considered out-of-scope for testing as they are not critical to the current implementation of the system:

* **System Performance**:
  + Scalability and responsiveness under stress.
  + Load testing and determining response times under heavy usage are not required at this stage.

**Types of testing used:**

* **Manual Testing:** This was used to manually test the CLI. The data provided in the ‘TechMemo.md’ provided a source of information to check against the system. I used this data to check for any spelling errors in the text that the system outputs to the user. I also tested if the system handled errors gracefully without throwing exceptions and whether the outputted numbers were concatenated to 2 decimal places.
* **Unit tests:** These were used to test individual methods like ‘findViableCentres’ to ensure that they work as expected.
* **Integration tests:** These were used to Test functions that call other functions and classes like ‘findOptimalCentres’ which calls 3 functions, or other functions that use instances of the historic and recycling classes.
* **Parameterised tests with CSV data:** These were used to save time and improve code readability, I could pass in extra data for more code coverage and to test different scenarios for functions such as null values, empty values, different waste splits and distributions of waste types.

## Test Environment & Tools

lWTIER ABOUT MANUAL TESTS HERE

Below are the tools I have used for my automated tests:

|  |  |
| --- | --- |
| **Tool** | **Explanation** |
| Laptop | My personal laptop (Razer Blade 14, 2014 model) that I have used to run the SUT on and write tests on. This uses windows 10 as an operating system. |
| IntelliJ | The IDE of choice for testing the SUT. |
| JUnit5 (Java) | The testing framework I have used to write tests. |
| CSV data | I have used this to parameterise some of my tests and reduce code redundancy / repetition. |
| TestRecycling | A subclass of Recycling I have used for better ease of testingas the original class is abstract. |
| Test data | The following mock data was used for testing:**centre1**: Location.B, 5 years active, generation "Alpha", rates [1.0, 1.0, 1.0]**centre2**: Location.B, 10 years active, generation "Beta", rates [1.5, 1.5, 1.5]**centre3**: Location.C, 15 years active, generation "Gamma", rates [1.5, 2.0, 3.0] This data was set up in a void setup() with the ‘@BeforeEach’ notation, this allowed me to save space and stop repetitive code by not having to recreate this code for every test that required it. |

Table 1 — Tools used for testing

I used the information in the following documents to write my tests:

* **Spec\_v1.docx:** This is the design spec for the system.
* **TechMemo.md:** This is an explanation of the SUT and provided some test data in then form of examples of spelling error free text that I could use for my manual tests.
* **MemorandumOfUnderstanding.md:** An explanatory email that briefed me on the system and its requirements, this included some examples of how the system should work.

## Statistics

[Here you may wish to calculate and include some summary statistics: how many tests passed/failed, how many were automated/manual, how many defects were considered to be of a critical severity vs. a trivial severity, and so on. You may even like to consider and document the line coverage provided by your unit tests.]



Figure 1 — Tests run

A screenshot of a computer program

Description automatically generated

Figure 2 — Test coverage report

## Recommendations & Conclusion

[You may want to make some recommendations and draw some conclusions here based on the testing you have done. Did the experiment of having a junior developer using generative AI as a mentor work with regards to the Landfill Labs project?]

# Defect Report

Table 1: [Insert a caption for the table here. Populate the table with defects!]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Defect ID** | **Description** | **Steps to Reproduce** | **Expected Behaviour** | **Actual Behaviour** | **Priority** | **Severity** | **Status** | **Source** |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |