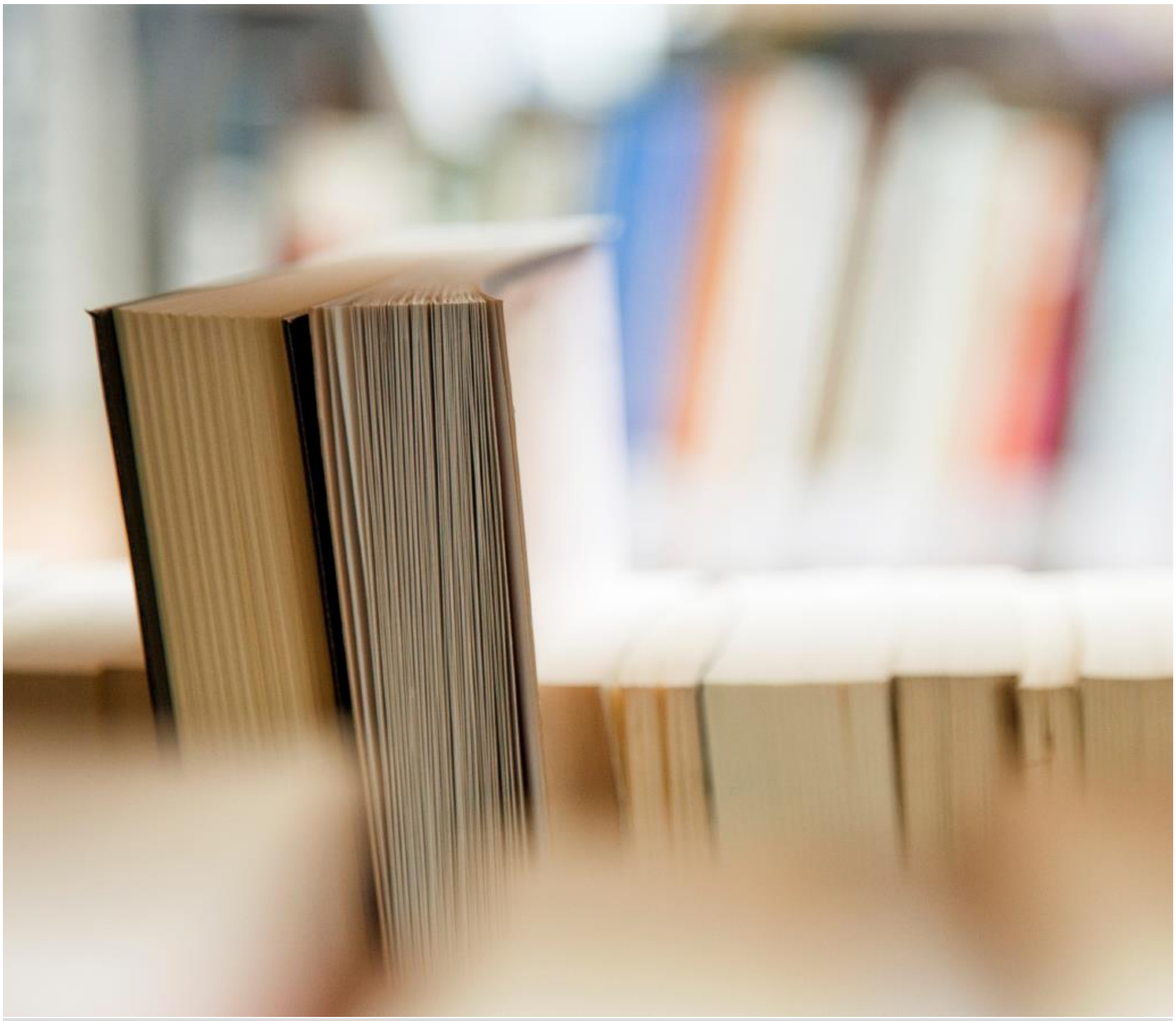

Document name: CLEANCITY TEST REPORT

Project: CleanCity - Waste Pickup Scheduler

Group Name: QA Commanders

Group Members: Nosipho Mdakane, Steven Odhiambo, Pathiswa Dlulane

Testing done by: Nosipho Mdakane, Steven Odhiambo, Pathiswa Dlulane

Submission date: July 16, 2025

Table of Contents

1. Project Overview 2

2. Testing Scope 2

3. Executive Summary 2

3.1 Key Findings 2

3.2 Recommendations 2

4.Test Strategy and Approach 3

5.Test environment Details 3

5.1 Device and System Testing 3

5.2 Operating Systems 3

5.3 Browser Testing... 3

5.4 User Interface (UI) and User Experience (UX) Testing 3

6. **Test Execution Summary** 3

7. Defect Analysis and Categorisation 3

8. Risk Assessment 4

8.1 Risk Analysis 4

8.1.1 Functional Risks 4

8.1.2 Non-Functional Risks 5

8.1.3 Business Risks 5

8.1.4 Organizational Risks 5

8.2 Risk Assessment 5

9. Recommendations and Improvements 5

9.1 Testing Enhancements 5

9.2 Error Handling Improvements 5

9.3 Accessibility Compliance (WCAG 2.1 Level AA) 5

9.4 Date Field Validation 6

10. Test Metrics and KPIs 6

10.1 Defect Density 6

10.2 Defect Removal Efficiency 6

10.3 Test Coverage 6

10.4 Test Case Effectiveness 6

10.5 Defect Leakage 6

10.6 Mean Time to Repair 6

10.7 Test Execution Time 7

10.8 Test Case Pass Rate 7

10.9 Defect Severity 7

10.10 Automation 7

10.11 Build Failure Rate 7

. **Appendices** 8

1. **Project Overview:**

- Project Name: Clean City software project.
- Version: #1
- Date: 16 July 2025
- Group Members: Nosipho Mdakane, Steven Odhiambo, Pathiswa Dlulane
- Environment: Development Stage

2. **Testing Scope:**

- In-Scope Modules: Waste collection scheduling and public transit route planning.
- Test Types: Unit Testing, Functional Testing, Performance Testing and UI Testing.
- Test Coverage: Defects, Issues and Bugs

3. **Executive Summary**
   - CleanCity software testing process successfully validated the core functionality of the waste management system, including route optimization and citizen reporting.
   - Further testing is recommended after changes have been implemented to the coding so to assess the impact of these changes.
   - The changes to be implemented as per the key findings and below:

   a. **Key findings**

- Summarize the results of the testing process.
- Highlight areas where the software met expectations and performed well.
- Identify any bugs and defects encountered.
- Provide issues found and their severity.

   b. **Recommendations**

- Add more unit tests for form validation and business logic in `script.js`.
- Consider automated end-to-end tests (e.g., Cypress, Playwright) for UI flows.
- Improve error handling for network failures.

4. **Test Strategy and Approach**
   - Test strategy incorporated various testing types focusing on functionality, performance, security, compatibility and usability.
   - Test Aproach focused on:

- Unit Testing
- User Acceptance Testing (UAT)
- Performance Testing
- Security Testing
- Usability Testing

5. Test environment Details

a. Device and System Testing

- Mobile Devices - Test on different smartphones and tablets, considering various screen sizes, processing power and operating system versions.

5.2 Operating Systems - Ensure compatibility with different operating systems that might be used by city officials or waste management personnel.

5.3 Browser Testing: Test on different types of Browsers to ensure compatibility.

5.4 User Interface (UI) and User Experience (UX) Testing:

- Accessibility
- User Friendliness
- Security Testing and Data Protection
- Data Integrity:

6. Test Execution Summary

- Total Number of Test Cases: 11
- Number of Passed: 10
- Number of Failed: 1
- Number of Blocked: 0
- Pass/Fail Rate: 90%

7. Defect Analysis and Categorisation

No	Description	Severity	Status	Distribution
1.	The System accepts a schedule duplicate	High	Open	Admin Portal
2.	No alt text on awareness images	Medium	Open	Awareness Page
3.	Missing alt Attributes on Images	Medium	Open	Google Chrome\Dev Tools\Issues Tab
4.	Form Accepts Past and Blank Dates	Medium	Open	date input field
5.	Dashboard filter by 'Eldoret' does not return results	High	Open	Dashboard page, filter
6.	Admin status update does not refresh dashboard	High	Open	Admin Page

7.	No validation for special characters in name fields	Medium	Open	Login Page
8.	Password field accepts less than 6 characters	Medium	Open	Login Page
9.	Feedback form accepts invalid Request IDs	Medium	Open	Feedback Form
10.	XSS vulnerability in feedback/comments fields	High	Open	Feedback/comments field
11.	No error message for invalid login credentials	Medium	Open	Login Page
12.	Tab order skips key form fields	Low	Open	All forms
13.	Color contrast fails accessibility standards	Low	Open	All Pages
14.	No confirmation dialog before deleting a request	Low	Open	Login as Admin
15.	Local Storage not cleared on logout	Medium	Open	Logout Tab
16.	Responsive layout breaks on iPhone SE	Low	Open	All pages
17	No feedback if network is offline	Low	Open	Form Submission

8. Risk Assessment

8.1 Risk Analysis

8.1.1 Functional Risks:

Core features of the software:

- Incorrect waste collection routes leading to missed pickups.
- Failure to notify residents about emergencies.
- Inaccurate data captured.

8.1.2 Non-Functional Risks:

Software performance and usability:

- Slow response times during peak usage.
- Data handling
- Accessibility issues for citizens with disabilities i.e Blind users.
- Failure to identify duplicates might cause delays in service delivery.

8.1.3 Business Risks:

Overall project and its impact on operations:

- Budget overruns due to unexpected development costs.
- Delays in implementation impacting city services.

- Depending on a single vendor for critical software components.

8.1.4 Organizational Risks:
Internal processes and workforce:

- Lack of trained personnel to operate the software.
- Data security breaches.

8.2 Risk Assessment:

Risk Analysis	Severity
Functional Risks	High
Non-Functional Risks	High
Business Risks	High
Organizational Risks	High

9. Recommendations and Improvements

Recommendations for Development Team:

To improve the overall reliability, accessibility, and user experience of the application, the following actions are recommended:

9.1 Testing Enhancements:

- Add more unit tests** for `form validation` and core `business logic` in `script.js` to improve code coverage and catch edge cases early.
- Implement automated end-to-end tests using tools such as **Cypress** or **Playwright** to validate critical UI flows and user interactions across various scenarios.

9.2 Error Handling Improvements:

- Enhance **network failure handling** by implementing proper error messages, retry logic (if applicable), and fallback behaviours to ensure a smoother user experience during connectivity issues.

9.3 Accessibility Compliance (WCAG 2.1 Level AA)

- To improve accessibility and align with international standards:

  Ensure all **images** include **descriptive** `alt` attributes that

  accurately convey their purpose or content. This is crucial for screen reader compatibility.

- Use `alt=""` for **decorative images** to avoid unnecessary noise in screen reader output.

  - For icon **buttons or links**, include `aria-labels` or visible **text equivalents** to ensure the functionality is communicated clearly.

  - Regularly audit the application using tools like **Lighthouse** and **axe DevTools** to identify and resolve accessibility regressions proactively.

9.4 Date Field Validation:

Ensure robust validation for any **date input fields:

- The field should **reject past dates** and only allow current or future selections.
- Mark the field as required on both the client side (`required` attribute) and server side.
- Use additional constraints like `min` in the HTML markup and validate in JavaScript or backend logic to guard against manipulation.

10. Test Metrics and KPIs

10.1 Defect Density:

- Definition: Number of defects found per unit of code.
- Purpose: Measures defects in different parts of the software, highlighting areas requiring more attention.

10.2 Defect Removal Efficiency:

- Definition: The percentage of defects found and fixed during testing before release.
- Purpose: Measures the effectiveness of the testing process for early defect detection.

10.3 Test Coverage:

- Definition: The extent to which the software functionality is covered by test cases.
- Purpose: Ensures that all critical parts of the software are tested.

10.4 Test Case Effectiveness:

- Definition: The ability of test cases to find defects.
- Purpose: How well the test cases are designed to uncover potential issues.

10.5. Defect Leakage:

- Definition: to identify defects found in production that were missed during the testing phase.
- Purpose: Indicates the effectiveness of the testing process.

10.6 Mean Time to Repair:

- Definition: Turnaround time to fix a defect.
- Purpose: efficiency of the bug fixing process.

10.7 Test Execution Time:

- Definition: Turnaround time to execute a set of test cases.
- Purpose: Optimizes the testing process and assist on increasing testing speed.

10.8 Test Case Pass Rate:

- Definition: Identify how many test cases passed.
- Purpose: Indicates the reliability of the software.

10.9 Defect Severity:

- Definition: Identifies the impact of defects on user experience.
- Purpose: Assist to prioritize bug fixes.

10.10. Automation:

- Definition: Identify automated tests.
- Purpose: Increase testing speed process.

10.11 Build Failure Rate:

- Definition: The number of times a build fails during testing.
- Purpose: Indicates the stability of the build process.

Appendices

Supporting documentation

https://github.com/dlulanep/PLP-Database-DEPT-CleanCity/issues

https://github.com/dlulanep/PLP-Database-DEPT-CleanCity/blob/main/docs/test-data.md

https://github.com/dlulanep/PLP-Database-DEPT-CleanCity/blob/main/tests/test_plan.md

Screenshots

https://github.com/dlulanep/PLP-Database-DEPT-CleanCity/tree/main/tests/Screenshots

Test Cases

https://github.com/dlulanep/PLP-Database-DEPT-CleanCity/blob/main/tests/test_cases.md