CleanCity Waste Pickup Scheduler – Test Report

Document Version: 1.3
Application Version: 1.0.0
Document ID: TR-CC-2025-011
Date: November 17, 2025
Prepared by: QA Team

1. Executive Summary

The CleanCity Waste Pickup Scheduler underwent extensive functional, UI/UX, and security testing. Jest was used exclusively for **unit testing** at the component level, while Selenium was used for **automated end-to-end UI testing** across all critical workflows.
Jest component tests recorded a 100% pass rate (30/30). However, Selenium revealed failures in registration validation, admin CRUD operations, pickup persistence, notifications, gamification, and responsiveness. Manual testing validated these failures and uncovered additional issues related to language support and the payment module.

Overall Release Status: NOT READY for production

2. Areas Covered in Testing

2.1 Functional Testing

• Authentication (login, registration)
• Waste pickup scheduling
• Notifications module
• Admin dashboard workflows
• Gamification (points, badges, leaderboard)
• Feedback form and blog module

2.2 Non-Functional Testing

• UI/UX consistency
• Accessibility and usability
• Security checking (XSS, CSRF behavior, password checks)
• Cross-browser responsiveness

2.3 Unit Testing (Jest)

Component-level tests validated:
• Input validation logic
• Rendering of core components
• NotificationBell, Login, Register, App routing

2.4 Automated UI Testing (Selenium)

Selenium validated end-to-end workflows:
• Login & logout
• Registration workflows
• Scheduling requests
• Canceling requests
• Admin CRUD operations
• Gamification and notifications

3. Scope of Testing

In Scope

• Form validation (login, register, scheduling)
• User and admin workflows
• UI/UX and accessibility
• localStorage persistence validation
• Jest unit testing
• Selenium automated UI testing

Out of Scope

• Backend API/database testing
• High-load performance testing
• Live SMS, email, payment, maps integrations

4. Test Approach

4.1 Testing Methodology

A hybrid testing model was used:

• Manual exploratory testing
• Structured test case execution
• Unit tests for components (Jest)
• Automated UI tests (Selenium)

4.2 Testing Levels

• Unit Testing – Jest
• Integration Testing – Component interaction
• System Testing – User workflows
• Regression Testing – After bug fixes

4.3 Browsers Tested

• Chrome (latest)
• Firefox (latest)
• Safari (iOS)
• Edge (latest)

4.4 Devices Tested

Device	OS / Platform	Browser
Nothing Phone 2a	Android 14	Chrome
Redmi Note 14	Android 14	Chrome
Lenovo ThinkPad T490s	Windows 11	Chrome / Firefox / Edge
HP Spectre	Windows 11	Chrome / Edge
HP ZBook G6	Windows 11	Chrome / Edge / Firefox

5. Test Execution Summary

Status	Count	Notes
Pass	30	Jest unit tests, basic authentication flows
Fail	38	Admin CRUD, notifications, persistence, gamification
Draft/Pending	62	Multilingual, responsiveness, performance

6. Defect Analysis

A total of 38 defects were identified. All remain open and require resolution before production release.

Severity	Count
Critical	10
High	12
Medium	8
Low	8

6.1 Critical Defects

• Admin CRUD operations failing
• Payment module errors
• Missing 2FA
• Pickup persistence failure
• Notification triggers not firing

7. Risk Assessment

Risk Area	Impact	Likelihood	Risk Level
Authentication Failures	High	Medium	High
Data Loss (Persistence)	High	High	Critical
Admin CRUD Failures	High	High	Critical
Notification Issues	Medium	High	High

8. Recommendations

• Fix all critical & high defects before deployment.
• Add missing authentication features (2FA, confirm password).
• Improve admin dashboard stability.
• Strengthen notification and persistence logic.
• Complete multilingual, responsive, and offline testing.
• Expand Selenium regression automation.

9. Approvals

Role	Name	Status
Test Manager	Mercy Melody Chemutai	Pending
Risk Analyst	Lorraine Bwayo	Pending
Test Executor	Susan Mwangi	Pending

Powered by: