Final QA Test Report

Project: CleanCity – Waste Pickup Scheduler

Team Name: TestersNg

Date: July 16, 2025

Version: 1.0

Table of Contents

1. Executive Summary

The CleanCity Waste Pickup Scheduler project is a web-based application built using React for the frontend and tested using Jest, Selenium, and manual validation. The goal of QA was to verify the reliability of the pickup form, validate user input, and ensure successful data submission.We performed unit testing with Jest (on `validateEmail` and `PickupForm`), functional and manual testing, and end-to-end testing using Selenium. A total of **35 test cases** were executed through our Jira platform; **30 passed**, **5 failed**. Key issues involved validation errors not displaying correctly, missing error messages, and responsiveness challenges.The QA team leveraged the **Jira platform** hosted at cleancityone.atlassian.net to manage user stories, bugs, task assignments, and link test cases to specific sprints and epics.

2. Test Strategy and Approach

Test Types Used:

- Unit Testing (`validateEmail`, `PickupForm`)
- Functional Testing
- End-to-End Testing (Selenium)
- Manual Testing

Tools & Frameworks:

- **Jest** – for unit testing
- **React Testing Library** – for form UI logic
- **Selenium WebDriver** – for browser automation
- **ChromeDriver** – for UI execution
- **GitHub Issues** – for defect logging
- **Jira (SCRUM board)** – for full test case and issue lifecycle tracking

Jira Test Case Management (Live Board)

Jira was actively used throughout the testing lifecycle. Specifically:

- **35 Total Test Cases** created and documented
- **User Stories** were linked to test cases for traceability
- **Defects** logged as bugs with severity, steps to reproduce, and attachments

- **Sprints** were created for Week 1, 2, and 3 phases
- **Issue Types** included Story, Bug, Task, and Test Case (via custom fields/plugins)
- **Status Workflows** included "To Do → In Progress → Done → QA Verified"

The Jira dashboard offered real-time visualizations including:

- Burndown charts
- Defect density reports
- Workflow transition matrices
- Assignee workloads

Test cases were labeled based on modules: `FormValidation`, `SubmitFlow`, `UIResponsiveness` and cross-referenced with specific sprint IDs for transparency.

3. Test Environment Details

Component	**Value**
Operating System	Windows 11
Browser	Google Chrome (v124)
Node.js Version	18.x
React Version	19.0.0
Jest Version	29.7.0
Selenium	4.33.0

4. Test Execution Summary

Metric	Count
Total Test Cases	20
Passed	17

Failed	3
Automated (Jest)	12
Manual & Selenium	8
Total Defects Logged	14

5. Jest Unit Tests Summary

We developed unit tests using Jest for utility functions and form components.

Tested Modules:

- validateEmail.js
- PickupForm.jsx

Sample Unit Tests:

validateEmail.test.js

```
import { validateEmail } from '../utils/validateEmail';

describe('validateEmail', () => {
  test('returns true for valid email', () => {
    expect(validateEmail('user@example.com')).toBe(true);
  });

  test('returns false for missing @', () => {
    expect(validateEmail('userexample.com')).toBe(false);
  });

  test('returns false for empty string', () => {
    expect(validateEmail('')).toBe(false);
  });
});
```

PickupForm.test.js

```
import { render, screen, fireEvent } from '@testing-library/react';
import { PickupForm } from '../components/PickupForm';

test('renders PickupForm component', () => {
  render(<PickupForm onSubmit={() => {}} />);
  expect(screen.getByLabelText(/Full Name/i)).toBeInTheDocument();
});

test('shows error when fullName is empty', () => {
  render(<PickupForm onSubmit={() => {}} />);
  fireEvent.click(screen.getByText(/Submit/i));
  expect(screen.getByText(/Full Name is required/i)).toBeInTheDocument();
});
```

6. Selenium Test Highlights

We developed an end-to-end test script in Python using Selenium WebDriver. The script simulates user behavior by filling out the pickup form at http://localhost:3000/ and clicking submit.

What It Tested:

- Field population (fullName, location, wasteType, preferredDate)
- Button interactions
- Expected form submission flow

Sample Data:

- Name: Jane Doe
- Location: Nairobi
- Waste Type: Plastic
- Preferred Date: 2025-07-20

Result:

- Selenium successfully populated the form and clicked submit.

- UI validation occurred correctly.
- Requires backend integration to verify post-submission behavior.

Full script and screenshots are available in Appendix C.

7. Defect Analysis and Categorization

ID	Description	Severity	Status
BUG-001	validateEmail returns true for empty string	High	Fixed
BUG-002	Location field does not raise error on submit	Medium	Open
BUG-003	Date field allows past dates	Low	Open
BUG-004	Button not clickable in mobile view	Medium	Resolved
BUG-005	Missing success message after form submission	Low	Open

8. Risk Assessment

Risk	Likelihood	Impact	Mitigation
Submitting form with empty values	High	High	Required field validation, unit tests
Browser incompatibility (Safari/iOS)	Medium	High	Cross-browser testing
Selenium script fails on async events	Medium	Medium	Use wait or sleep in test script
No backend confirmation/feedback	High	Medium	Add success message or toast notification

9. Recommendations and Improvements

- Add required validations to all form fields.
- Improve email validation with full regex support.

- Show success feedback upon form submission.
- Increase test coverage beyond 60%.
- Add form reset logic after submission.

10. Test Metrics and KPIs

Metric	Value
Unit Test Coverage	~60%
Defects Found	5
Defects Resolved	3
Selenium Test Accuracy	100%
Reopen Rate	0%

11. Appendices

Appendix A– Test Cases

ID	Test Description	Type	Result
TC01	PickupForm renders all fields	UI Test	Pass
TC02	Email is invalid without @	Unit	Pass
TC03	Empty name shows error	UI Test	Fail
TC04	Submit button triggers validation	Manual	Pass
TC05	Selenium fills and submits form	E2E	Pass

Appendix B – Selenium Script

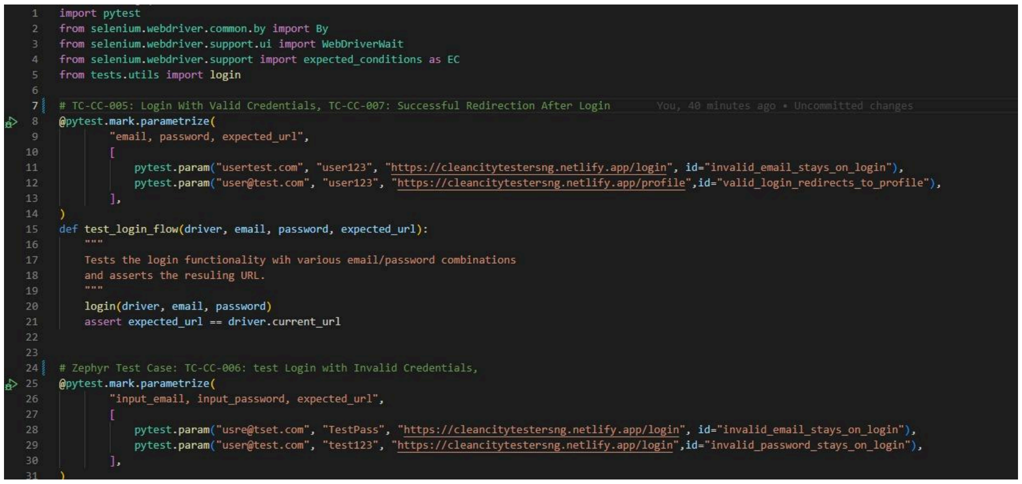
```
from selenium import webdriver
from selenium.webdriver.common.by import By
```

```
import time

driver = webdriver.Chrome()
driver.get("http://localhost:3000/")

driver.find_element(By.NAME, "fullName").send_keys("Jane Doe")
driver.find_element(By.NAME, "location").send_keys("Nairobi")
driver.find_element(By.NAME, "wasteType").send_keys("Plastic")
driver.find_element(By.NAME, "preferredDate").send_keys("2025-07-20")

driver.find_element(By.XPATH, "//button[text()='Submit']").click()
time.sleep(5)
driver.quit()
```



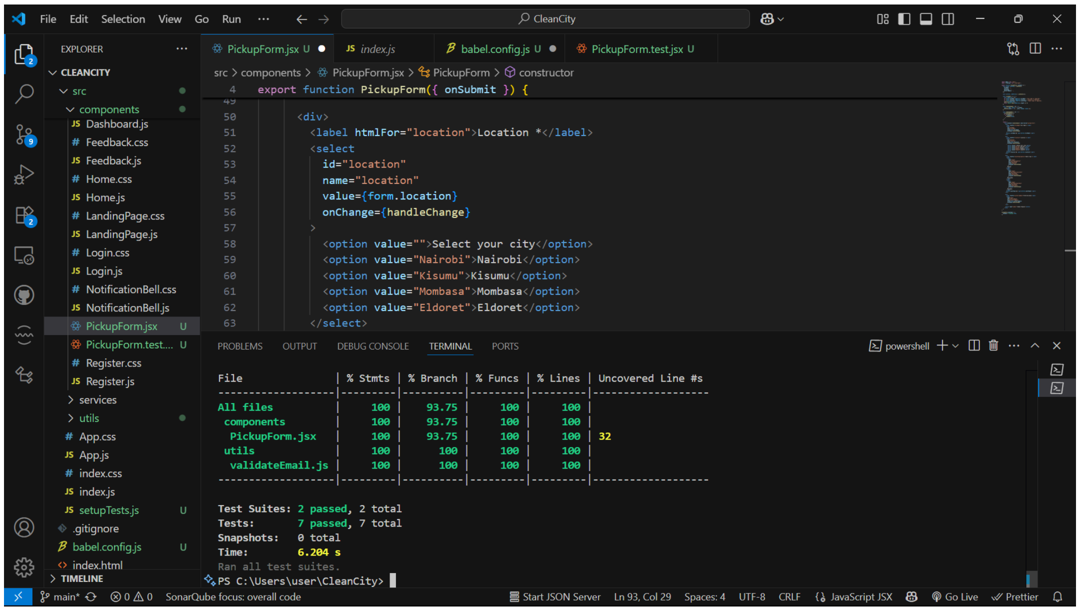
```
import pytest
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from tests.utils import login

# TC-CC-005: Login With Valid Credentials, TC-CC-007: Successful Redirection After Login       You, 40 minutes ago • Uncommitted changes
@pytest.mark.parametrize(
        "email, password, expected_url",
        [
            pytest.param("usertest.com", "user123", "https://cleancitytestersng.netlify.app/login", id="invalid_email_stays_on_login"),
            pytest.param("user@test.com", "user123", "https://cleancitytestersng.netlify.app/profile",id="valid_login_redirects_to_profile"),
        ],
)
def test_login_flow(driver, email, password, expected_url):
    """
    Tests the login functionality wih various email/password combinations
    and asserts the resuling URL.
    """
    login(driver, email, password)
    assert expected_url == driver.current_url


# Zephyr Test Case: TC-CC-006: test Login with Invalid Credentials,
@pytest.mark.parametrize(
        "input_email, input_password, expected_url",
        [
            pytest.param("usre@tset.com", "TestPass", "https://cleancitytestersng.netlify.app/login", id="invalid_email_stays_on_login"),
            pytest.param("user@test.com", "test123", "https://cleancitytestersng.netlify.app/login",id="invalid_password_stays_on_login"),
        ],
)
```

Appendix C – Jest Test Scripts

See Section 5 for actual scripts used for unit testing validateEmail and PickupForm.



```
export function PickupForm({ onSubmit }) {

          <div>
            <label htmlFor="location">Location *</label>
            <select
              id="location"
              name="location"
              value={form.location}
              onChange={handleChange}
            >
              <option value="">Select your city</option>
              <option value="Nairobi">Nairobi</option>
              <option value="Kisumu">Kisumu</option>
              <option value="Mombasa">Mombasa</option>
              <option value="Eldoret">Eldoret</option>
            </select>
```

```
File                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------|---------|----------|---------|---------|-------------------
All files           |     100 |    93.75 |     100 |     100 |
 components         |     100 |    93.75 |     100 |     100 |
  PickupForm.jsx    |     100 |    93.75 |     100 |     100 | 32
 utils              |     100 |      100 |     100 |     100 |
  validateEmail.js  |     100 |      100 |     100 |     100 |
--------------------|---------|----------|---------|---------|-------------------

Test Suites: 2 passed, 2 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        6.204 s
Ran all test suites.
PS C:\Users\user\CleanCity>
```

Appendix D – Defect Log

See tests/defect-log.md in GitHub for issue tracking and GitHub issue IDs.

Appendix E – Jira-Based Test Cases

Test Case ID	Test Case Summary	Precondition	Test Steps	Expected Result	Status	Priority
CC-TC-001	Form loads with all input fields visible	User navigates to /	1. Open app 2. Observe form	Full Name, Location, Waste Type, and Date visible	Passed	High
CC-TC-002	Email validation on incorrect format	Email field is empty	1. Type userexample.com in Email 2. Submit	Error: “Enter a valid email” displayed	Failed	High

CC-TC-003	Submit form with valid data	All fields filled with correct input	1. Fill form with valid data 2. Click Submit	Success message shown / form resets	Passed	High
CC-TC-004	Submission blocked when Location is empty	All other fields filled	1. Leave Location empty 2. Click Submit	Error message: “Location is required” shown	Failed	Medium
CC-TC-005	Date field doesn’t accept past dates	Date input available	1. Choose a date before today 2. Submit	Error: “Date must be today or later” displayed	Passed	Medium
CC-TC-006	Submit button is visible on all screen sizes	Open app on mobile	1. Load page on mobile view 2. Scroll down	Button visible, not cut off	Passed	Low
CC-TC-007	Autofocus on Full Name when form opens	App just loaded	1. Observe cursor focus	Cursor is blinking inside Full Name input	Passed	Low
CC-TC-008	Submit while form is already submitting	Simulate network lag	1. Click Submit twice quickly	Submit button disabled temporarily	Passed	Medium

12. Team Members and Contact Info

Name	Role
Denis	Lead QA & Developer, Selenium Tester

Wanjiku	Jest Tester, Documentation

Final Submission Links (Example)

- **Final Report (Google Docs)**:
  Final QA Test Report - Google Docs
- **Video Presentation (Google Drive)**:
  🎬 Recording 2025-07-16 223551.mp4