**Test Strategy Document  
DISTRICT**

V 0.1

Author: ProtestEr

Date: 17/07/2025

Table of Contents

[1.0 Introduction 2](#_Toc203670025)

[1.1 Purpose 2](#_Toc203670026)

[1.2 Target Audience 2](#_Toc203670027)

[2.0 System Description 3](#_Toc203670028)

[3.0 Test Deliverables 3](#_Toc203670029)

[4.0 Test Types 4](#_Toc203670030)

[5.0 Multi-Browser Testing 4](#_Toc203670031)

[6.0 Test Data 5](#_Toc203670032)

[7.0 Resources Required 5](#_Toc203670033)

[7.1 Hardware 5](#_Toc203670034)

[7.2 Software 5](#_Toc203670035)

# Introduction

The PROTESTER-HACKATHON-PROJECT is focused on building a comprehensive automation framework for the district.in web application. The project leverages Java, Selenium, and Cucumber to automate end-to-end testing of critical business workflows. By implementing modular and reusable test components, the project aims to reduce manual testing effort, accelerate release cycles, and improve overall product quality. The framework is designed to be scalable, maintainable, and easily extensible for future requirements.

## 1.1 Purpose

The primary purpose of this project is to provide a reliable and efficient automated testing solution for district.in. Automation will help in early detection of defects, ensure consistent validation of functionalities, and support rapid regression testing during development and deployment. The project also aims to establish the best practices in test automation, facilitate collaboration among team members, and deliver detailed reporting for informed decision-making.

## 1.2 Target Audience

This project is intended for QA engineers, automation testers, developers, project managers, and business stakeholders involved in the district.in platform. It serves as a reference for understanding the automation framework, test coverage, and reporting mechanisms. The audience includes both technical and non-technical members who are interested in the quality assurance process and project outcomes.

# 2.0 System Description

District.in is a feature-rich e-commerce platform that enables users to search for products, manage their shopping cart, complete purchases, and maintain personal profiles. The system includes modules for user authentication, product search, cart management, checkout, order history, and profile updates. The automation framework interacts with the web UI and underlying business logic to validate user journeys, data integrity, and system responses across different browsers and devices.

# 3.0 Test Deliverables

| Project Phase | Deliverables | Description |
| --- | --- | --- |
| Requirement Analysis | Test Requirement Document | List of testable requirements and acceptance criteria |
| Test Planning | Test Strategy & Test Plan | Approach, scope, resources, schedule, and risk analysis |
| Test Case Development | Automated Test Scripts, Feature Files, Step Definitions | Scripts and BDD files for all identified scenarios |
| Test Environment Setup | Environment Setup Guide, Configuration Files | Instructions and files for setting up test environment |
| Test Execution | Test Execution Reports, Screenshots | Detailed reports of test runs with evidence of failures |
| Test Cycle Closure | Summary Report, Defect Log, Lessons Learned Document | Final summary, defect analysis, and improvement recommendations |

# 4.0 Test Types

* Functional Testing: Validates individual features and user workflows for correctness
* Regression Testing: Ensures new changes do not break existing functionalities
* Data-Driven Testing: Tests application behaviour with multiple data sets and edge cases
* Cross-Browser Testing: Verifies consistent performance and appearance across Chrome, Firefox, and Edge
* Integration Testing: Checks interactions between different modules and third-party services
* UI Validation: Confirms that user interface elements render and behave as expected

# 5.0 Multi-Browser Testing

The automation framework is configured to execute tests on multiple browsers, including Chrome, Firefox, and Edge. This ensures that district.in delivers a consistent and reliable user experience regardless of the browser used. Selenium WebDriver is utilized for browser automation, and test results are compared to identify any browser-specific issues. Multi-browser testing also helps in detecting compatibility problems early in the development cycle.

# 6.0 Test Data

Test data is managed using structured Excel files, allowing for easy updates and scalability. The framework integrates Apache POI to read and utilize test data for various scenarios, including valid, invalid, and boundary cases. Data-driven testing enhances coverage by simulating real-world user inputs and conditions. Proper test data management ensures repeatability and reliability of automated tests.

# 7.0 Resources Required

## 7.1 Hardware

* Development PCs or Laptops

## 7.2 Software

* Java JDK for developing and running automation scripts
* Maven for project build and dependency management
* Selenium WebDriver for browser automation
* Cucumber for BDD scenario implementation
* Apache POI for Excel-based data-driven testing
* Latest versions of Chrome, Firefox, and Edge browsers
* Git for source code management and collaboration
* Visual Studio Code as the primary development environment
* Extent Reports for generating detailed test execution reports
* Jenkins or GitHub Actions for CI/CD integration and automated test execution