**Go Local**

**Test Strategy and Plan for Agile**

**Table of Contents**

**1.0 Purpose of This Document 4**

**2.0 Introduction 4**

**2.1 Program/ Project Overview 4**

**2.2 Application Overview 5**

**3.0 Testing Objectives 5**

**4.0 Testing Scope 6**

**4.1 Scope of testing 6**

**4.2 In Scope 7**

4.2.1 Program/Application / Module 7

4.2.2 Type of Testing 7

4.2.3 Other In-Scope Items 7

**4.3 Out of scope 8**

4.3.1 Program/Application / Module 8

4.3.2 Type of Testing 8

4.3.3 Other Out-of-Scope Items 8

**5.0 Testing Approach & Strategy 8**

**5.1 Feature Testing Approach & Coverage 9**

**5.2 Regression Approach & Coverage 9**

**5.3 Integration Testing Approach & Coverage 9**

**5.4 Performance & Security Testing Approach & Coverage 10**

**5.5 Test Automation Approach & Coverage 10**

**5.6 Test Data Management 10**

**5.7 Testing Phases, Scope, Deliverables, Acceptance Criteria, Impacted Applications 10**

**6.0 Overall Testing approach 11**

**6.1 Test Design Approach 11**

6.1.1 Design Methodology 12

6.1.2 Review Process 12

6.1.3 Entry Criteria for Test Design 12

**7.0 Test Execution Approach 12**

**7.1 Execution Methodology 12**

**7.2 Test Data Management 12**

**7.3 Execution Cycles 12**

**8.0 Tools & Hardware / Software Requirements 13**

**8.1 Tools 13**

**8.2 Test Management Tool 13**

**9.0 Assumption 14**

**10.0 Dependencies 14**

**11.0 Defect Management 14**

**11.1 Defect Process Steps 14**

**12.0 Defect Severity & Priority Definition or Classification Guidelines 16**

**1.0** **Purpose of This Document**

The purpose of this document is to decide and define the testing approach for Go Local. This document is intended for project stakeholders, namely Business, Scrum Teams, Architecture Team, NFR Team, Release team, Scrum Master, Agile Coach and so on.

This document will aim to:

· Define the processes, approach & Methodology to be followed for this project

· Define and communicate the scope and boundaries of testing

· Outline the definition and entry and exit criteria, assumptions, dependencies, schedule and also roles and responsibility of the resources

· Provide Reference on Definition of Ready and Definition of Done as applicable

· Define deliverables across all Sprints / Iteration/ Releases and phases within it

· Identify the Infrastructure, Tools and Test Environments for all types of testings

· Identify key risk issues and their respective mitigation/contingency plan

# **2.0** **Introduction**

This Test Plan outlines the QA testing strategy adopted to ensure the high-quality delivery of the Go Local web application. The objective is to validate that all features and business rules are implemented correctly and meet the defined business requirements prior to release. This document details the testing practices and procedures to be followed, including the scope of testing, tools, resources, and timelines.

The Go Local platform is designed to connect users with trusted local service providers across various categories such as electricians, plumbers, tutors, beauticians, and more. The application aims to enhance transparency, trust, and convenience in service discovery and booking, ultimately improving user satisfaction and supporting local businesses.

## **2.1** **Program/ Project Overview**

The Go Local project is a location-based community service platform aimed at simplifying the process of discovering and engaging with nearby service providers. The program supports multiple user roles—customers, service providers, and administrators—each with distinct functionalities.

**Estimated Benefits and Outcomes:**

* Improved accessibility to verified local services
* Enhanced user experience through intuitive search and booking features
* Empowerment of service providers with tools to manage profiles and bookings
* Community engagement via reviews and ratings
* Scalable architecture for future expansion

**Timelines:**

The project is expected to follow an agile development cycle with iterative testing phases.

## **2.2** **Application Overview**

Go Local is a web-based application that facilitates location-based service discovery and booking. It includes the following core modules:

* Landing Page: Highlights top-rated service providers and customer testimonials.
* Sign-Up/Login: Separate registration flows for customers and service providers.
* User Dashboard: Personalized dashboard post-login with search and navigation features.
* Service Listings: Filtered view of service providers based on city and profession.
* Profile Pages: Detailed profiles for both customers and service providers.
* Admin Dashboard: Tools for monitoring platform activity and managing content.

The application is designed to be responsive and accessible across devices, with a focus on usability, security, and performance.

# **3.0** **Testing Objectives**

* Ensure all features meet the acceptance criteria defined by the business team.
* Confirm that the application behaves correctly for different types of users (customers, service providers, and admins).
* Check that the system is stable, secure, and performs well under normal usage.
* Make sure the user interface is clear, consistent, and works across different devices and browsers.
* Support the development team by providing feedback and helping to improve the quality of the product.

# **4.0** **Testing Scope**

4.1 Scope of testing

< Include details on scope of testing to define the Required Functional Testing levels, Supplemental Functional Tests, Supplemental Non-Functional Tests, etc. Grouping under supplemental functional and Non-functional could be modified under the project need>

| **Test** | **Primary Contact** | **Comments** |
| --- | --- | --- |
| **Required Functional Testing levels:** |  |  |
| Unit | Development Team | Each module will be tested individually during development. |
| System/ Integration | QA Team | End-to-end testing of workflows and data flow between modules. |
| User Acceptance | Business Analyst | Final validation by stakeholders to ensure the system meets business needs. |
| Production Verification | QA Lead | Quick checks after deployment to confirm system stability. |
| **Supplemental Functional Tests:** |  |  |
| Security Testing | QA Team | Ensure user data is protected and access is properly controlled. |
|  |
|  |
| Usability Testing | QA Team | Check if the app is easy to use and navigate. |
|  |
|  |
| Data Verification and Application Conformity Testing | QA Team | Validates that data is displayed and stored correctly. |
| Beta Testing | Product Team | Testing by a small group of real users before full release. |
| Pilot Testing | Business Analyst | Limited release to test in a real-world environment. |
| Parallel Testing | QA Team | Comparing results between old and new systems (if applicable). |

## **4.2** **In Scope**

This section outlines the overall scope of testing activities for the Go Local application. It includes the specific modules, types of testing, and other relevant items that will be covered during the QA process.

### **4.2.1** **Program/Application / Module**

**Program:** Go Local – A location-based service platform connecting users with local service providers.

**Application:** Web-based application accessible via desktop and mobile browsers.

**Modules in Scope:**

* Landing Page
* Sign-Up and Login (for Customers and Service Providers)
* User Dashboard
* Service Listings and Search
* Profile Pages
* Booking and Review System
* Admin Dashboard

### **4.2.2** **Type of Testing**

* Functional Testing: To validate all user flows and business logic.
* System Integration Testing: To ensure modules work together as expected.
* User Acceptance Testing (UAT): Final validation by stakeholders.
* Security Testing: To check for data protection and access control.
* Usability Testing: To ensure the application is user-friendly and intuitive.
* Data Verification: To confirm accuracy and consistency of stored and displayed data.
* Production Verification Testing: Quick checks post-deployment to confirm stability.

### **4.2.3** **Other In-Scope Items**

* Cross-browser and cross-device testing
* Basic performance checks under expected load
* Accessibility checks (basic ADA compliance)
* Feedback loop during beta or pilot testing (if conducted)

## **4.3** **Out of scope**

This section outlines the areas that are not included in the current testing scope for the Go Local project. These exclusions are based on the current phase of development, available resources, and project priorities.

### **4.3.1** **Program/Application / Module**

* Mobile App (Native): Only the web version of the application is in scope. Any native Android or iOS app versions are not part of this testing cycle.
* Third-party Integrations: External systems or APIs not yet integrated or finalized are excluded from testing at this stage.

### **4.3.2** **Type of Testing**

* Performance Load Testing at Scale: While basic performance checks will be done, full-scale load testing is not included in this phase.
* Penetration Testing: Deep security testing by external experts is not part of this cycle and will be handled separately.
* Disaster Recovery Testing: Not applicable for this release.

### **4.3.3** **Other Out-of-Scope Items**

* Legacy System Comparison: There is no existing system to compare with, so parallel testing is not applicable.
* Multi-language Support: Internationalization and localization testing will be considered in future phases.
* Voice or AI-based Features: Not part of the current feature set.

# **5.0** **Testing Approach & Strategy**

The testing approach for the Go Local project follows an agile methodology, where testing is integrated into each sprint cycle. The QA team collaborates closely with developers and business analysts during story grooming, sprint planning, and daily stand-ups to ensure clarity on requirements and early identification of risks. Acceptance criteria are defined for each user story, and test cases are created accordingly. Functional, integration, and user acceptance testing are conducted for every release, with regression testing performed before major deployments to ensure system stability.

To manage changes in scope or requirements, the QA team participates in backlog refinement and reprioritization discussions. Any scope creep is addressed through impact analysis and updated test planning. Regular sprint demos and retrospectives help track progress and improve processes. The team also supports DevOps by validating builds in the CI/CD pipeline and ensuring that quality checks are in place before deployment. This strategy ensures that testing remains aligned with business goals and timelines while maintaining high product quality.

## **5.1** **Feature Testing Approach & Coverage**

The QA team will follow a structured approach to test each feature of the Go Local application, ensuring that all functionalities are validated against their acceptance criteria. Testing will include functional testing, integration testing, browser compatibility checks, and basic API validation where applicable. Manual testing will be the primary method in this phase, with regression testing conducted in each sprint to ensure new updates do not affect existing features. The goal is to provide thorough coverage across all user roles and workflows to ensure a smooth and reliable user experience.

## **5.2** **Regression Approach & Coverage**

Regression testing for the Go Local application will be conducted in every sprint to ensure that newly added features or fixes do not negatively impact existing functionality. The QA team will maintain a regression suite that covers all critical workflows such as user registration, login, service search, booking, and profile management. As the application evolves, the suite will be updated to reflect changes in features and business rules. Regression tests will be executed manually for now, with plans to automate high-priority scenarios in future phases to improve efficiency and consistency.

## **5.3** **Integration Testing Approach & Coverage**

Integration testing for the Go Local application will focus on verifying the interaction between different modules such as user registration, login, service listings, booking, and profile management. The goal is to ensure that data flows correctly between components and that combined functionalities work as expected. The QA team will test key user journeys that span multiple modules, such as signing up, searching for a service, and completing a booking. Any dependencies between frontend and backend systems, or between user roles (e.g., customer and service provider), will be tested to confirm smooth integration and consistent behavior across the platform.

## **5.4** **Performance & Security Testing Approach & Coverage**

Performance and security testing for the Go Local application will focus on ensuring the system remains responsive and secure under expected usage conditions. Basic performance checks will be conducted to monitor page load times and responsiveness across key workflows. Security testing will include validation of user authentication, role-based access control, and protection of sensitive data such as personal information and login credentials. While advanced load and penetration testing may be handled in later phases or by specialized teams, the QA team will ensure that core security and performance aspects are covered during regular sprint cycles.

## **5.5** **Test Automation Approach & Coverage**

The automation strategy for the Go Local application will focus on improving test efficiency and reducing manual effort over time. Initially, manual testing will be used to validate new features, while automation will be introduced gradually for stable and repetitive test cases such as login, registration, and booking workflows. The QA team plans to use standard automation tools to build a regression suite that can be run during each sprint and before releases. Automation coverage will expand as the application matures, helping ensure faster feedback, consistent results, and smoother CI/CD integration.

## **5.6** **Test Data Management**

Effective test data management is essential for validating the features of the Go Local application. The QA team will use a mix of real-world scenarios and dummy data to simulate user interactions such as registrations, bookings, and profile updates. Data sets will be created to cover various user roles (customers, service providers, and admins) and edge cases. Sensitive data will be masked or anonymized where needed to maintain privacy and security. Test data will be refreshed regularly to ensure consistency across test cycles and to support accurate and repeatable results.

## **5.7** **Testing Phases, Scope, Deliverables, Acceptance Criteria, Impacted Applications**

**Testing Phases**

* Unit Testing
* Integration Testing
* System Testing
* User Acceptance Testing (UAT)
* Regression Testing

**Scope**

* Validation of all functional and non-functional requirements
* Coverage of end-to-end business processes
* Verification of system interfaces and integrations

**Deliverables**

Test Strategy and Test Plan - Defines the overall approach and implementation for testing activities across the project.

Test Cases - Developed based on requirements and acceptance criteria, ensuring comprehensive coverage and traceability.

Test Suites - Organized collections of test cases for structured execution.

Test Execution Reports - Summarizes the progress and results of test execution at defined intervals.

Defect Reports - Tracks and categorizes identified issues by priority, type, and resolution status.

Test Summary Report - Consolidates outcomes from all testing phases, highlighting requirement coverage, defect status, and overall system readiness.

**Acceptance Criteria**

* Successful execution of all critical and high-priority test cases
* No unresolved critical or high-priority defects
* Formal approval from business stakeholders
* Compliance with defined performance and quality benchmarks

# **6.0** **Overall Testing approach**

This section outlines the comprehensive approach to testing across the project lifecycle, ensuring alignment with business requirements, quality standards, and delivery timelines.

## **6.1 Test Design Approach**

The test design will be guided by the selected methodology and aligned with agile practices. The following aspects will be considered:

**6.1.1 Design Methodology**

The approach may include Acceptance Test Driven Development (ATDD), Behaviour-Driven Development (BDD), or other relevant frameworks. Feature files and structured scenarios will be created where applicable.

**6.1.2 Review Process**

All test design artifacts will undergo peer review to ensure completeness, accuracy, and alignment with requirements.

**6.1.3 Entry Criteria for Test Design**

* Finalized and approved user stories or requirements
* Defined acceptance criteria for each user story
* Availability of test data or test data strategy
* Access to test environments
* Completion of design walkthroughs and reviews

## **7.0** **Test Execution Approach**

The execution strategy will ensure thorough validation of all systems and interfaces, using a combination of manual and automated testing techniques.

**7.1 Execution Methodology**

* Manual testing will be used for exploratory, ad-hoc, and UI validations.
* Automation will be leveraged for regression, repetitive tasks, and API validations.
* Virtualization and mock services may be used where dependent systems are unavailable.

**7.2 Test Data Management**

* Test data will be created or masked from production sources as per compliance guidelines.
* Data sets will be refreshed and maintained for each cycle.

**7.3 Execution Cycles**

* Multiple cycles of test execution will be planned, including dry runs, formal cycles, and regression cycles.
* Each cycle will include defect logging, resolution, and revalidation.

# **8.0** **Tools & Hardware / Software Requirements**

This section outlines all tools, software, and hardware/devices required for successful execution of the project.

## **8.1** **Tools**

| **Testing Type / Phase** | **Tool** | **Version** | **Licensed / Open Source / Cognizant Proprietary** |
| --- | --- | --- | --- |
| Development Unit Testing |  |  |  |
| Functional / End to End / Regression / UAT/ Prod |  |  |  |
| Continuous Integration |  |  |  |
| Test Data Management |  |  |  |
| Test Automation |  |  |  |
| Test Management |  |  |  |
| Performance testing |  |  |  |
| Security testing |  |  |  |
| Release & Deployment |  |  |  |
| Others |  |  |  |

## **8.2 Test Management Tool**

The test management tool used in this project supports comprehensive quality assurance activities throughout the testing lifecycle. Key features include:

· Test case repository

· Traceability and test coverage

· Test execution

· Test Reports

· Defect Tracking

# **9.0** **Assumption**

The following assumptions are made in defining and executing the test strategy for this engagement:

· Dedicated testing environment will be made available for testing

· Appropriate Subject matter experts (SMEs) and Product Owners will be made available to clarify requirements and resolve any requirements ambiguities

· Each story will be developed and unit tested by the development team before it’s made available for the feature testing team to start feature testing

· All identified defects in a particular sprint will be fixed within the timelines of the same sprint. If not, they will be added back to the product backlog and resolved in any subsequent builds / Sprints based on the severity and priority of the defects

· Emergency build will be made available as a fix for any showstopper defects>>

# **10.0** **Dependencies**

The successful execution of the test strategy is dependent on the following factors:

· Test Environments will be available during the onsite and offshore timings except the planned downtime window and adequately staffed for support

· Test Environment integrated with needed integrations and downstream for Integration Testing and System Testing

· All queries/ambiguities are clarified before the test design starts

# **11.0** **Defect Management**

Defect management ensures that all identified issues are tracked, prioritized, and resolved efficiently to maintain product quality. The following process outlines the steps for managing defects during the project lifecycle:

**11.1 Defect Process Steps**

**Initial Validation**

When QA identifies a discrepancy between the requirement and the application build, it is first validated with the Developer or SME to confirm whether it is a defect. This step helps reduce unnecessary entries and overhead.

**Clarification Communication**

Clarifications are exchanged via email with a standardized subject format:

Project\_Sprint\_StoryID\_Description

(e.g., BC\_Sprint4\_US1234\_LoginValidationIssue)

**Defect Logging**

Upon confirmation, QA logs the defect in the Agile project management tool (e.g., Rally, Jira).

Status: Submitted

Schedule State: Defined

**Severity Assignment**

QA assigns a severity level based on the impact and urgency. Refer to the Test Strategy document for severity definitions and defect status lifecycle.

**Reproduction Steps**

QA documents detailed steps to reproduce the defect, ensuring developers can replicate the issue in their environment.

**Developer Assignment**

The defect is assigned to the developer responsible for the related user story or task.

**Follow-Up and Communication**

QA follows up daily with the developer until the defect is resolved. Updates are shared during daily scrum meetings and reminders are sent via email if needed.

**Fix and Deployment**

* Once the developer fixes the defect, the updated code is deployed to the QA environment.
* Developer updates the defect status to Fixed
* Notification is sent to the defect submitter
* Verification and Closure
* QA verifies the fix.
* If resolved, the defect status is changed to Closed
* If not resolved, the status is reverted to Open and the Schedule State is updated to In Progress

## **12.0** **Defect Severity & Priority Definition or Classification Guidelines**

| **Defect Severity** | **Definition / Classification Guideline** |
| --- | --- |
| Critical | Fatal error or “Show Stopper”  No Work Around - Cannot continue testing |
| High | Prevents the system from meeting business requirements  Major Functionality Failure  Workaround is complex and and/or time consuming,  Testing can continue in another area and slow down the execution progress |
| Medium | Affects functionality (not critical)  Simple Workaround is available  Testing can proceed |
| Low | Cosmetic issue  Does not affect any functionality  Testing can proceed without interruption |