***Test Plan Template: Smart Garage TI***

**Prepared by:**  
Viktoria Spasova  
Nikolay Vlasenko

## **TABLE OF CONTENTS**

**1.0 Introduction  
2.0 Objectives and Tasks**  
 2.1 Objectives  
 2.2 Tasks  
**3.0 Scope**  
**4.0 Testing Strategy** 4.1 Unit Testing  
 4.2 System and Integration Testing  
 4.3 Performance Testing  
 4.4 Smoke Testing  
 4.5 Automated Regression Testing  
 4.6 Acceptance Testing  
**5.0 Hardware Requirements  
6.0 Environment Requirements**  
 6.1 Main Frame  
 6.2 Workstation  
**7.0 Test Schedule  
8.0 Control Procedures  
9.0 Features to Be Tested  
10.0 Features Not to Be Tested  
11.0 Entry Criteria  
12.0 Resources/Roles & Responsibilities  
13.0 Schedules  
14.0 Significantly Impacted Departments (SIDs)  
15.0 Dependencies  
16.0 Risks/Assumptions  
17.0 Tools  
18.0 Exit Criteria  
19.0 Approvals**

## **1.0 INTRODUCTION**

Smart Garage is a web application designed for auto repair shop owners to manage day-to-day operations efficiently. It serves both employees and customers, providing functionality for vehicle management, service tracking, report generation, email notifications, and loyalty management. The purpose of this test plan is to ensure that all features meet functional, technical, and business requirements through structured manual and automated testing.

## **2.0 OBJECTIVES AND TASKS**

### 2.1 Objectives

* Ensure all functional requirements are correctly implemented.
* Validate input, authentication, and authorization mechanisms.
* Verify filtering, sorting, report generation, PDF creation, currency conversion, and email notifications.
* Confirm REST API endpoints are functional and documented via Swagger.
* Identify defects and report them for resolution.

### 2.2 Tasks

* Design detailed manual test cases for all functional modules.
* Execute test cases and document results.
* Create Automated suites that cover at least the happy paths.
* Conduct integration testing for workflows like customer registration → service → report generation → email.
* Perform smoke testing for core features.
* Run automated regression tests on critical paths.
* Prepare test summary and defect reports.

## **3.0 SCOPE**

**In Scope:**

* Unit Testing
* Integration Testing
* Smoke Testing
* API Testing
* **Out of Scope:**
* Performance Testing
* Security Testing
* Non-functonal Testing

## **4.0 TESTING STRATEGY**

### 4.1 Unit Testing

* Focus on input validation, business rules, and exception handling.
* Target at least 80% code coverage in the service layer with BDD-style unit tests.

### 4.2 Integration Testing

* Test full workflows: customer registration → vehicle/service creation → PDF report generation → emails.

### 4.4 Smoke Testing

* Ensure core functionality works: login, user registration, service addition, PDF/email generation.

### 4.5 API Testing

* Verify all main endpoints.
* Test CRUD Operations for core entities – Users, Vehicles and Services.
* Check staus codes, request/response bodies.

## **5.0 HARDWARE REQUIREMENTS**

* Server: 8GB RAM, 4-core CPU, 250GB SSD
* Database server (MariaDB)
* Workstations: 8GB RAM, dual-core CPU, 100GB HDD

## **6.0 ENVIRONMENT REQUIREMENTS**

### 6.1 Main Frame

* Docker
* Database server (MariaDB)

### 6.2 Workstation

* Browsers: Chrome, Firefox, Edge
* Test tools: Java, Selenium (for automation), Postman and Rest-Assured (API testing),

## **7.0 TEST SCHEDULE**

* Test Case Design: 3–4 days
* Manual Test Execution: 1 day
* Smoke Testing: 1 day
* Regression Testing: 2 days
* Test Report & Documentation: 1–2 days

## **8.0 CONTROL PROCEDURES**

* Track defects in Jira.
* Review test cases before execution.
* Maintain version control for test artifacts.

## **9.0 FEATURES TO BE TESTED**

### ****Public Part****

* Login with valid/invalid credentials
* Register with valid/invalid credentials
* Forgotten password functionality
* Anonymous access to public pages

### ****Customer Part****

* Service list and filtering by vehicle/date
* Detailed visit reports
* PDF report generation
* Currency selection for reports
* Password change
* Personal Details change

### ****Employee Administrative Part****

* Vehicle management: viewing, filtering, sorting and creating
* Service management: browsing, creating, updating and deleteting
* Customer management: browsing, filtering, deleting and updating
* Report management: viewing, currency selecting and generating PDF reports

## **10.0 FEATURES NOT TO BE TESTED**

* Apply to Mechanic Job
* Visit Rating
* Remote Visit Repair
* Loyalty Program Discount
* Spare Parts
* Service Visit Calendar

## **11.0 ENTRY CRITERIA**

* Application deployed in test environment.
* Test data prepared for customers, vehicles, services.
* REST API deployed with Swagger documentation.

## **12.0 RESOURCES / ROLES & RESPONSIBILITIES**

* **Viktoria Spasova:** Test case design, manual testing, defect reporting
* **Nikolay Vlasenko:** Test execution, smoke testing, regression testing

## **13.0 SCHEDULES**

* Week 1: Test case creation, environment setup
* Week 2: Manual testing, defect logging
* Week 3: Regression testing, acceptance testing, final report

## **14.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)**

* IT / Development Team
* Auto repair shop administration
* Customer support
* HR (for job applications)

## **15.0 DEPENDENCIES**

* Database server availability
* Third-party API access (currency conversion, car models)
* REST API deployed and documented

## **16.0 RISKS / ASSUMPTIONS**

* Risk: Email or PDF failures due to network or server issues
* Risk: Currency conversion service may be unavailable
* Assumption: Test data accurately represents production scenarios

## **17.0 TOOLS**

* Jira – Defect tracking, test management
* Selenium – Automated regression tests
* Postman – REST API testing
* Browser developer tools – UI inspection
* PDF viewer – Report verification

## **18.0 EXIT CRITERIA**

* All high-priority test cases executed and documented
* All critical defects resolved or logged with workarounds
* 80% test coverage of unit tests

## **19.0 APPROVALS**

| **Role** | **Name** | **Signature** | **Date** |
| --- | --- | --- | --- |
| QA Tester | Viktoria Spasova | \_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_ |
| QA Tester | Nikolay Vlasenko | \_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_ |