Ensuring quality of the platform while meeting the customer requirements and stability of the product by providing automated test.

Smart Garage

Testing Web Application (REST API & Web UI)

Andela Micic  
Danail Zahariev  
Nikola Dimitrov  
  
**Document Version**: 2.0  
**Last Updated**: 16.09.2025

Contents

[**1. Overview** 3](#_Toc208924370)

[**Introduction** 3](#_Toc208924371)

[**Purpose** 3](#_Toc208924372)

[**Objectives** 3](#_Toc208924373)

[**Tasks** 3](#_Toc208924374)

[**2. Testing Types** 4](#_Toc208924375)

[**2.1 Functional Testing** 4](#_Toc208924376)

[**2.2 API Testing** 4](#_Toc208924377)

[**2.3 Integration Testing** 4](#_Toc208924378)

[**2.4 Regression Testing** 4](#_Toc208924379)

[**3. Scope of Testing** 4](#_Toc208924380)

[**3.1 Functionalities to be Tested** 4](#_Toc208924381)

[**3.2 Functionalities NOT to be Tested** 6](#_Toc208924382)

[**4. Entry Criteria** 6](#_Toc208924383)

[**5. Exit Criteria** 6](#_Toc208924384)

[**6. Resources** 6](#_Toc208924385)

[**Roles & Responsibilities** 6](#_Toc208924386)

[**Tools & Technologies** 6](#_Toc208924387)

[**7. Schedule** 7](#_Toc208924388)

[**8. Test Environment** 7](#_Toc208924389)

[**Configuration** 7](#_Toc208924390)

[**Test Accounts** 7](#_Toc208924391)

[**9. Metrics & Reporting** 8](#_Toc208924392)

[**Test Coverage Metrics** 8](#_Toc208924393)

[**Execution Metrics** 8](#_Toc208924394)

[**Defect Metrics** 8](#_Toc208924395)

[**Reporting Deliverables** 8](#_Toc208924396)

[**10. Risk Analysis** 8](#_Toc208924397)

[**11. Defect Management** 9](#_Toc208924398)

[**Severity Levels** 9](#_Toc208924399)

[**Defect Workflow** 9](#_Toc208924400)

[**12. Test Deliverables** 9](#_Toc208924401)

[**Documentation** 9](#_Toc208924402)

[**Test Artifacts** 9](#_Toc208924403)

[**Reports** 9](#_Toc208924404)

[**13. Dependencies** 10](#_Toc208924405)

[**Dependencies** 10](#_Toc208924406)

**1. Overview**

**Introduction**

Smart Garage is a comprehensive web application designed for auto repair shop owners to efficiently manage daily operations. The system supports two user roles: **Customers** who can manage their vehicles and view service history, and **Employees** who handle administrative functions including customer management, vehicle registration, and service operations. The application includes a REST API for third-party integration.

**Purpose**

This Test Plan defines the systematic approach, methodology, and deliverables for ensuring Smart Garage meets all functional requirements, maintains high quality standards, and provides a stable production-ready experience for end-users.

**Objectives**

* **Functional Verification**: Ensure all critical features work according to requirements
* **Quality Assurance**: Achieve >95% test pass rate for core functionality
* **Automation Coverage**: Implement automated testing for regression prevention
* **Defect Prevention**: Identify and resolve issues before production release
* **Documentation**: Provide comprehensive testing evidence and reports

**Tasks**

1. Test planning and strategy development
2. Test case design and documentation
3. Manual exploratory testing of all modules and features
4. Automated test suite development
5. Defect management and tracking
6. Final reporting and metrics analysis

**2. Testing Types**

**2.1 Functional Testing**

**Objective**: Verify all features work according to application requirements

* **UI Testing**: Manual and automated testing using Java + Selenium
* **Business Logic**: Validation rules, calculations, workflows

**2.2 API Testing**

**Objective**: Validate all features work correctly on API level

* **Tools**: Postman for manual testing, Java + RestAssured for automation
* **Scope**: All CRUD operations for Users, Vehicles, Services
* **Validation**: Status codes, response formats, data validation
* **Documentation**: Swagger API documentation verification

**2.3 Integration Testing**

**Objective**: Ensure system components work together correctly Discover edge cases and usability issues in unscripted scenarios and patterns

* **UI-API Integration**: Data consistency across layers

**2.4 Regression Testing**

**Objective**: Ensure new changes don't break existing functionality

* **Automated Suite**: Core functionality coverage
* **Scope**: Happy paths + critical scenarios

**3. Scope of Testing**

**3.1 Functionalities to be Tested**

**Entity Validation**

* **License Plate Validation**: Bulgarian format compliance
* **VIN Validation**: 17-character requirement, forbidden characters
* **Year Validation**: Range 1886-present
* **Model/Brand Validation**: Length constraints (2-50 characters)

**Access Control & Security**

* **Authentication**: Login, logout, session management
* **Authorization**: Customer vs Employee access rights
* **Password Management**: Change password

**Vehicle Management**

* **CRUD Operations**: Create, Read, Update, Delete vehicles
* **Customer Linking**: Vehicle-customer relationships
* **Filtering**: By owner, make, model, year
* **Sorting**: Multiple column options
* **Auto-creation**: New make/model handling

**Service Management**

* **Service CRUD**: Complete lifecycle management
* **Pricing Validation**: Non-negative values
* **Filtering**: By name and/or price
* **Service History**: Visit tracking and reporting

**Customer Management**

* **Profile Management**: CRUD operations
* **Search Functionality**: By name, email, phone, vehicle
* **Visit Management**: Service history tracking
* **Report Generation**: PDF reports, currency conversion

**3.2 Functionalities NOT to be Tested**

* Optional features (loyalty program, job applications)
* Payment gateway integration (bank/credit card)
* Minor CSS/styling issues
* Third-party API internals
* Easter eggs and hidden features

**4. Entry Criteria**

* Test data prepared (users, vehicles, services)
* Access credentials provided for all roles
* Testing tools configured (Selenium, Postman, GitHub)
* Database scripts for test data ready
* API documentation (Swagger) available

**5. Exit Criteria**

* All Critical and High priority test cases executed
* Test pass rate >90%
* Test execution report delivered
* Defect metrics documented

**6. Resources**

**Roles & Responsibilities**

**QA Engineer Trainees**

* Test planning and documentation
* Manual test execution
* Automation development
* Defect reporting and tracking
* Test report generation

**Tools & Technologies**

* **Test Management**: JIRA
* **Automation**: Java + Selenium WebDriver, Junit5
* **API Testing**: RestAssured, Postman
* **Version Control**: GitHub
* **Reporting**: Allure Report, Jira(Xray)
* **Browsers**: Chrome (latest), Firefox (latest)
* **IDE**: IntelliJ IDEA

**7. Schedule**

| **Week** | **Activities** |
| --- | --- |
| **1** | • Test planning & strategy  • Test case design  • Manual exploratory testing |
| **2** | • API testing with Postman  • Integration testing |
| **2** | • Automation framework setup  • Core test automation  • Regression suite creation |
| **3** | • Complete automation suite  • Final testing & reporting |
| **4** | • Documentation completion  • Project defense preparation |
|  |  |

**8. Test Environment**

**Configuration**

* **Application URL**: http://localhost:8081/
* **API Endpoint**: http://localhost:8081/api/
* **Database**: MariaDB

**Test Accounts**

| **Role** | **Username** | **Password** | **Purpose** |
| --- | --- | --- | --- |
| Customer | felix\_jackson | password123%D | Admin functions |
| Employee | alex\_rider | password123%D | Customer view |
| Mechanic | kevin\_brown | password123%D | Mechanic view |

**9. Metrics & Reporting**

**Test Coverage Metrics**

* **Requirements Coverage**: 100% of MUST requirements
* **Code Coverage**: 85%+ for service layer
* **UI Coverage**: 70% of critical user paths
* **API Coverage**: 85% of endpoints

**Execution Metrics**

* **Executed**: Target 100%
* **Pass Rate**: Target >90%
* **Automation Rate**: 25% of test cases

**Defect Metrics**

* **By Severity**: Critical/High/Medium/Low
* **By Module**: Vehicle/Service/User/UI/API/REST
* **Defect Density**: Bugs per module

**Reporting Deliverables**

1. **Final Test Report**: Complete analysis
2. **Defect Report**: All issues with status

**10. Risk Analysis**

| **Risk** | **Impact** | **Probability** | **Mitigation** |
| --- | --- | --- | --- |
| Limited testing time | High | High | Prioritize critical tests, parallel execution |
| Environment instability | Medium | Low | Backup environment, deployment scripts |
| Requirement changes | Medium | Low | Regular sync meetings, change management |
| Third-party API downtime | Medium | Low | Mock services for testing |
| Resource availability | Medium | Low | Cross-training, documentation |
| Complex test data setup | High | High | Automated data scripts |

**11. Defect Management**

**Severity Levels**

* **Critical**: Major feature broken
* **High**: Blocking issue
* **Medium**: Feature partially working
* **Low**: Minor issue or cosmetic defect

**Defect Workflow**

1. **Discovery**: Found during testing
2. **Logging**: Create JIRA bug
3. **Triage**: Assign severity, priority and mark as defect if needed

**12. Test Deliverables**

**Documentation**

* Test Plan
* Test Cases
* Test Execution Report
* Defect Report
* Automation Code Repository

**Test Artifacts**

* Selenium automation framework
* Postman API collection
* Test data scripts

**Reports**

* Coverage analysis
* Pass/fail metrics
* Defect analysis
* Final presentation

**13. Dependencies**

**Dependencies**

* Access to all system modules
* API documentation completeness
* Third-party service availability
* Database access for verification