Test plan for “Hillel Auto” application

1. Introduction

This document is a test plan for the Hillel Auto web and mobile application, which will undergo full testing by the QA engineers team. Hillel Auto is a modern digital tool designed for car owners who want to have complete control over the maintenance of their car.
Key features of the application include:

1. Monitoring of parts and fluid replacement schedules;
2. Planning maintenance costs;
3. Access to over 100 step-by-step instructions for self-diagnosis and repair.

Hillel Auto’s target audience is car owners of the following brands and models:

- Audi: TT, R8, Q7, A6, A8
- BMW: 3 Series, 5 Series, X5, X6, Z3
- Ford: Fiesta, Focus, Fusion, Mondeo, Sierra
- Porsche: 911, Cayenne, Panamera
- Fiat: Palio, Ducato, Panda, Punto, Scudo

The application is optimized for use in popular browsers: Safari, Google Chrome, Opera, and Mozilla Firefox, which allows a cross-browser compatibility and accessibility for a wide range of users

This test plan defines the test items, approach, entry/exit criteria, environments needs, responsibilities, risk and contingencies

The QA engineers team will be responsible for the full testing cycle, including functional, regression, UX/UI, and cross-browser testing.

2. Test items

The QA engineers team will test the following key elements of functionality and user interface:

- Access types: Guest Mode; User Mode
- Registration and Authorization
- User Profile
- Vehicle Management
- Instruction library

- Interface and Navigation
- Cross-Browser Compatibility
- Security

3. Approach

Our team will use the following testing types and techniques:

- Functional Testing
- Cross-browser Testing
- UI/UX Testing
- Regression Testing
- Performance Testing
- Basic Security Testing
- Pairwise Testing
- Boundary value analysis
- Exploratory Testing

4. Entry/Exit criteria

Entry criteria:

- Requirements and documentation are clearly defined, approved, and available to the QA team.
- Test environment is set up and accessible
- Test cases are written, reviewed, and approved by the QA Lead.
- All necessary third-party tools are configured and operational.
- A working version of the app is available on all supported browsers (Safari, Chrome, Opera, Firefox).

Exit criteria:

- All planned test cases (functional, UI/UX, regression, cross-browser) have been executed.
- All critical and high-priority bugs are fixed, verified, and closed.
- A detailed test summary report has been created and shared with interested party.
- Product meets the defined acceptance criteria for release.

5. Environments needs

Browsers:

- Safari (latest vers.)
- Chrome (latest vers.)
- Opera (latest vers.)
- Firefox (latest vers.)

Devices:

- HP ProBook 450 G10 FHD i5-1334U/16GB/512GB (Windows 11 Professional)
- Dell Inspiron 15 3530 IntelCore i7/16GB/1TB (Windows 10 Professional)
- MacBook Air 13 Retina, 256GB, 8 CPU / 8 GPU, 16GB RAM with Apple M2 (macOS Sonoma 14.7.3)

Tools:

- Testcaselab (Writing test cases)
- Jira (Tracking and managing Bugs, Tasks)
- Lightshot (Defects visualization)

6. Responsibilities

QA Lead
Responsible for the overall test strategy, approving the test plan and test cases, monitoring testing progress, managing the QA team, and communicating with stakeholders about the quality status of the product.

QA Engineers
Develop, execute, and update test cases. Responsible for functional, UI/UX, regression, and cross-browser testing. Log bugs, retest them after they are fixed, and provide feedback on the usability and stability of the product.

Developers
Responsible for fixing bugs identified by QA, providing technical explanations, supporting QA in reproducing complex defects, and ensuring the stable integration of changes into the product.

Project Manager
Formulate functional requirements, prioritize tasks, monitor deadlines, and make the final release decision based on testing results.

UX/UI Designer

Provide design mockups, advise QA on interface compliance, and review design comments made during testing.

7. Risk and contingencies

- Delay in providing a stable version for testing
- Incomplete or outdated documentation
- Limited resources
- Unstable test environment
- A large number of changes in the final stages of development
- Difficulty in maintaining cross-browser compatibility