



**Translantik**  
SIMPLY FLEET MANAGEMENT

# TRANSLANTIK TEST PLAN

John Doe

Johndoe@cybertekschool.com



## Table of Contents

### **1** 3

#### **1.1** 3

#### **1.2** 3

### **2** 4

#### **2.1** 4

#### **2.2** 4

#### **2.3** 4

#### **2.4** 5

##### 2.4.1 5

##### 2.4.2 5

##### 2.4.3 5

##### 2.4.4 5

#### **2.5** 5

##### 2.5.1 5

##### 2.5.2 6

##### 2.5.3 6

##### 2.5.4 6

#### **2.6** 6

### **3** 7

#### **3.1** 7

#### **3.2** 7

#### **3.3** 7

##### 3.3.1 7

##### 3.3.2 7

##### 3.3.3 7

##### 3.3.4 7

### **4** 8

#### 4.1.1 8

#### 4.1.2 8

#### 4.1.3 8

#### 4.1.4 9



**5** 9

5.1.1 9

5.1.2 9

5.1.3 9

5.1.4 9

**TEST PLAN APPROVAL**

**11**

---

## 1 INTRODUCTION

### 1.1 PURPOSE OF THE TEST PLAN DOCUMENT

This document serves as a test plan. It describes the testing approach and automation framework that will test the application under test. This document describes:

- Application under test overview
- Testing strategy
- Test management
- Test Risks
- Deliverables

Each section is divided into several subsections that serve to provide more detailed insight on the description and the goals of that section.

This document focuses on the functional testing of the application. Providing details on the strategy and management of performance testing and the security testing is not in the scope of this document.

### 1.2 APPLICATION UNDER TEST OVERVIEW - *TRANSLANTIK*

**Translantik** is an online fleet management solution that is highly specialized for companies that handle large vehicle fleets. **Translantik** makes the complete process of handling vehicle fleets fully transparent from expense and procedures point of view. **Translantik** enables smooth operations between the company, its clients who lease company services and vehicles, and drivers. **Translantik** offers highly customizable solutions for tracking vehicle lifecycle and expenses, lead management, driver management, as well as solutions for reporting and analytics.

Main modules of the **Translantik** application are Dashboard, Fleet, Customers, Sales, Activities, Marketing and Calendar. More information about the functionalities of the application is given in the requirements documentation of the application.

## **2 TESTING STRATEGY**

### **1.**

The company utilizes Behavior Driven Testing approach to ensure that the automated testing brings business value. BDD model puts the highest priority to business values and return on investment rather than chasing a number of test cases. The company will work to identify priorities based on the value they bring. Tools like Cucumber BDD, Selenium WebDriver will be used to implement the BDD testing approach.

### **2.1 TEST OBJECTIVES**

The objective of the test is to verify that the functionality of the application works according to the requirements. Automated testing focuses only on functional, smoke, end-to-end, and regression testing of the application. Tests will be integrated with the existing CI environment of the application.

Automated tests will be developed based on the requirements provided and executed to verify the functionality. Tests can be triggered manually as well as run automatically based on the certain schedule.

Automated testing framework includes reports with detailed description and the screenshots of the issues detected during the execution.

As the result of this project, reusable automated tests will be developed. Those tests can be periodically executed with some maintenance even after the end of the project.

### **A.**

### **2.2 ASSUMPTIONS**

Requirements are available for the application under test. Feature files will be written based on the requirements.

Business Analyst is available to work with test engineers at all times. The role of the BA in automated testing includes to participate in writing and verifying feature files, verify or reject reported bugs and defects, assign priority to tests cases and reported issues. BA will always be available for clarifying requirements and test cases.

Features of the application targeted for automated testing are stable enough for developing and executing tests. Any functionality that will be tested using automated tests must be stable otherwise automated testing will not have the expected return on invested time and effort.

Testing environment has the necessary test data. Test data reflects the production environment.

### **2.3 PRINCIPLES**

Testing reporting processes will be defined well ahead, yet flexible, with the ability to change as needed in accordance with the agile principles.

Gherkin and Cucumber will be used to write feature files that validate the acceptance criteria.

Feature files will be written focusing on meeting the business objectives, cost efficiency, and quality.

Testing environment and data will emulate a production environment as much as possible.

There will be clearly defined entrance and exit criteria for each test case.

Testing will be integrated with the CI/CD process and smoke tests will run for every build to find problems as early as possible.

Tests will be executed in different browsers to verify that application works that same way in all browsers.

## **2.4 SCOPE**

### **2.4.1 Functional testing**

Functional testing will be performed to verify if the application features are developed according to the specifications. Functional testing will be done manually and using automation. BA's will provide the scenarios for functional tests. However, testers also will need to write scenarios for functional tests when required.

Functional testing will be carried out by the functional testing team.

### **2.4.2 Performance testing**

Performance testing will be carried out in order to measure scalability, reliability and resource usage of the application under test. Performance testing will be carried out by the performance testing engineer. **Performance testing strategy and management details will be described in a separate document. Performance testing is not in the scope of this document.**

### **2.4.3 Security testing**

Security testing will be conducted to check for vulnerabilities, threats, risks in the application in order to make the application highly secure and prevent malicious attacks from intruders. **Security testing will be carried out by the performance testing engineer. Security testing strategy and management details will be described in a separate document. Security testing is not in the scope of this document.**

### **2.4.4 Layers of testing**

Functional testing will be on three layers of the application: UI, API, and Database. In each layer tests can be executed manually or using automation.

## **2.5 TYPES OF TESTS**

Automated tests are quite flexible and can be used to carry out several types of testing. However, Selenium tests are primarily used to do certain types of functional testing. All types of tests will be developed based on the same framework and generate the same type of reports.

### **2.5.1 Functional tests**

Automated testing will primarily focus on the functional testing of the application under test. Tests will be executed in the test environment. Functional tests verify functionality of a specific feature based on the requirements for that feature. Functional tests will be carried out by testing the User

Interface, the back end and web services of **Translantik** application. For automated functional tests, priority will be given to tests cases that bring most business value.

### **2.5.2 Smoke tests**

Smoke tests will be developed and executed periodically. Smoke testing will be used to identify the general stability of the **Translantik** application. Testing scenarios for the smoke test will be approved by the BA or SME(**Subject Matter Expert**). Smoke tests can run against multiple environments.

Smoke tests will be integrated to the CI environment and can be executed based on a schedule as well as be triggered after every deployment. Smoke tests reports will be emailed to the whole team in case of a failure. Typically, smoke test failure indicates a major issue with an application under test and requires immediate attention.

### **2.5.3 End-to-end tests**

The purpose of End-To-End testing is to ensure entire business processes that the system supports per requirements. The business flows for End-To-End testing will be identified in collaboration with the BA and feature files will be developed specifically for these types of tests. Typical End-To-End test may include testing multiple systems as part of the single test.

### **2.5.4 Regression tests**

Regression tests will be carried out after a major code change as well as before certain milestones. Functional automated tests and End-To-End which are stable and most value will be added to the Regression. Regression test suite will grow after every sprint iteration. Regression tests will be run as often as possible to keep the Regression suite up to date and find issues early. Maintaining the regression suite will be included in the estimation of the work of test engineers.

## **2.6 TEST DATA**

For functional testing, test data need to be generated to ensure proper testing of scenarios that depends on preset data input. Test data will be generated in several ways depending on the needs of specific test scenarios.

Methods to generate test data include

- Spinning up test environments in docker that come with preloaded test data
- Using database script to insert data directly to database
- Using JDBC and API to create test data before or during the automation test execution
- Using Selenium WebDriver to create the test data before or during the test execution

### 3 TEST MANAGEMENT

#### 3.1 TEST AND DEFECT MANAGEMENT

Jira will be used to reporting and tracking bugs and defects found by the automated tests. The procedure for reporting, verifying and prioritizing defects will be decided in cooperation with the stakeholders. Details that will be decided for the following processes:

- Tools used for defect tracking
- Reporting and verifying defects
- Defining and assigning priority and severity
- Fixing the reported defects
- Retesting and closing opened defects

#### 3.2 REPORTING

Comprehensive and detailed reports will be provided. The reports will include the business logic, test steps and screenshots. Reports for each test will be saved in the CI environment.

#### 3.3 FRAMEWORKS TOOLS

Automated testing framework will be developed by integrating different tools. This project will use Java based tools for automated testing. Tools are chosen based on their reliability, maintainability and business value. They also allow flexibility in testing the **Translantik** application in different browsers, allow AWS integration, provide detailed and comprehensive reporting.

##### 3.3.1 Cucumber

Cucumber is an open-source framework used for Behavior Driven Development. It brings the business side, developers, and the testers together. It allows doing testing with the business goal in mind.

##### 3.3.2 Selenium WebDriver

Selenium automated browsers. It allows writing and executing automated tests against multiple types of browsers. It is a widely used automation tool used for functional UI testing.

##### 3.3.3 JDBC

JDBC (Java Database Connectivity) is a java library that is used to establish connectivity between a java application and a database. It will be used testing the database related test cases, as well during test data preparation and clean-up.

##### 3.3.4 REST ASSURED

Rest Assured is a java-based library used to test RESTful web services. It makes API testing easy to understand and using the library for testing RESTful web services. Rest Assured is used both for testing web services of the application under test, as well as creating test data and posttest clean up.



## **4 TEST RISKS**

### **4.1.1 Lack of automated testing experience**

#### **4.1.1.1 Description**

The project does not have an automated testing process established. Introducing and implementing automated testing practices to current software development processes require significant changes to the current workflow. Integrating testing automated will not bring value in the initial stages and benefits will be seen over a certain time. Lack of experience and the wrong expectation may increase the customer dissatisfaction. The success of this project depends on proper initial planning.

This is a high impact risk.

#### **4.1.1.2 Mitigation Plan**

The test plan needs to be laid out and properly introduced in detail to the development team. Clear set of goals need to be established. Several sessions will be required to come up with the process of integrating automated testing to the current workflow. Timelines, testing strategy, defect management will be discussed.

### **4.1.2 Unstable test environment**

#### **4.1.2.1 Description**

Automated testing brings value when the application under test is stable. Having an unstable environment adds complications when it comes developing, executing automated tests. Executing automated tests on unstable environment creates unnecessary false flags.

This is a high impact risk.

#### **4.1.2.2 Mitigation Plan**

2. Ensure the stability of the environment by running the smoke tests often.
3. Coordinate with DevOps team and create flows to spin up new test environments using dockers on demand.
4. Write automated tests for the stable flows and functionalities only. Coordinate with BA in deciding what feature to automate.

### **4.1.3 Lack of detailed requirements**

#### **4.1.3.1 Description**

There needs to be clearly written requirements for functionalities that will be automated. Requirements will be used to write the feature files. Without the requirements, testers will not be able to write test cases.

This is a medium impact risk.

#### **4.1.3.2 Mitigation Plan**

Development team will provide requirements for features that will be tested. Engineers will work with the BA to write feature files.

#### **4.1.4 Technical implementation and resources**

##### **4.1.4.1 Description**

The benefit from automated tests increases over time. Automated tests have low ROI in the initial stages. It takes some time for setting up the architecture, automating test cases, integrating test cases. The fact that engineers are interns will add to this timeframe. Overall it may lead to customer dissatisfaction.

This is a low probability risk.

##### **4.1.4.2 Mitigation Plan**

Goals and timelines need to be clearly defined to the stakeholders to give a clear picture of the path ahead.

## **5 DELIVERABLES**

### **5.1.1 Test plan (this document)**

Test plan outlines the general testing strategy, objectives and risks. Developed by the test lead

### **5.1.2 Testing framework.**

At the end of the project automation will be built and delivered as a result of this project. Developed by the testing team.

### **5.1.3 CI integration**

At the end of the project automation will be built and delivered as a result of this project. Developed by the testing team in cooperation with the DevOps.

### **5.1.4 Duration**

Automated testing is part of agile development practice. To that end the automation project will continue as long as the development of the application continues. Once the development is completed, automated tests will still be maintained with a minimal team to support the production support efforts.

## 5. TEST PLAN APPROVAL

The undersigned acknowledge they have reviewed THE TEST PLAN document and agree with the approach it presents. Any changes to this Requirements Definition will be coordinated with and approved by the undersigned or their designated representatives.

Signature:	_____	Date:	_____
Print Name:	_____		
Title:	_____		
Role:	_____		

Signature:	_____	Date:	_____
Print Name:	_____		
Title:	_____		
Role:	_____		

Signature:	_____	Date:	_____
Print Name:	_____		
Title:	_____		
Role:	_____		