Software Test Plan

tankerkoenig

Mahmood Odeh

contents

1. Introduction

a. Purpose

The purpose of this Software Test Plan (STP) is to define the approach, resources, and schedule for testing the Tankerkoenig API. This document outlines the objectives, scope, and methodology for conducting API testing to ensure the quality and reliability of the Tankerkoenig API.

b. Project Overview

The Tankerkoenig API provides access to fuel price data for gas stations in Germany. It is a crucial component for applications that offer fuel price comparison or location-based services for drivers in Germany. The API is expected to be reliable, efficient, and secure to meet the needs of developers and users who rely on fuel price information.

c. Website Overview

The Tankerkoenig website (https://creativecommons.tankerkoenig.de/) offers information about fuel prices and gas stations in Germany. It provides a web interface for users to search for gas stations, view fuel prices, and access historical data. The website aims to help users find the best fuel prices and plan their refueling stops efficiently.

d. Key Features

- Fuel Price Data: Access current fuel prices for gas stations in Germany.
- Station Details: Retrieve details about gas stations, including location and services offered.
- Historical Data: Access historical fuel price data for analysis and comparison.
- Search Functionality: Search for gas stations based on location, fuel type, and other criteria.

2. Test Strategy

a. Test Objectives

- Verify the functionality and reliability of the Tankerkoenig API.
- Ensure that the API returns accurate and up-to-date fuel price data for gas stations in Germany.
- Validate the API's response times and performance under various loads.
- Verify the API's security measures to protect against unauthorized access.

b. Test Assumption

key assumption:

- The Tankerkoenig API documentation accurately reflects the API's functionality and behavior.
- The test data provided for testing accurately represents real-world fuel price scenarios in Germany.
- The testing environment accurately mirrors the production environment of the Tankerkoenig API.

General assumptions:

- General Assumption: Testers have access to the necessary tools and resources for testing the Tankerkoenig API.
- General Assumption: Testers have a basic understanding of API testing principles and methodologies.

Functional and Non-Functional Testing:

- The testing team will perform functional and non functional tests at the most of the API website.

User Acceptance Testing:

- Stakeholder Involvement: Involve stakeholders in the testing process to ensure the Tankerkoenig API meets their requirements and expectations.
- Scenario Testing: Test real-world scenarios to validate that the API behaves as expected in practical use cases.

- Feedback Collection: Gather feedback from stakeholders to identify any usability issues or areas for improvement, ensuring a user-friendly API.

c. Test Approach

- Manual Testing: Conduct manual tests to verify API functionality, usability, and data accuracy.
- Automated Testing: Use automated testing tools and scripts to perform regression and load testing, ensuring API reliability and performance.
- Security Testing: Perform security testing to identify and address vulnerabilities in the Tankerkoenig API.
- Performance Testing: Conduct performance testing to evaluate the API's response times and scalability, ensuring optimal performance under various loads.

d. Data Approach

- Use both synthetic and real-world data for testing to simulate various fuel price scenarios.
- Generate test data to simulate different gas station and fuel price scenarios, ensuring comprehensive test coverage.
- Use production-like data volumes to test performance and scalability, ensuring the API can handle large data sets effectively.

e. Levels of Testing

Exploratory

- **Purpose:** The purpose of this test is to identify and remove critical defects before proceeding to the next levels of testing.
- **Testers:** Testing team.
- **Method:** Conducted in the application without predefined test scripts or documentation.
- **Timing:** At the start of the testing phase.

- **Functional and non Functional Tests**

  Functional Test

  - **Purpose:** To test each function of the Tankerkoenig API by providing appropriate input and verifying the output against the functional requirements.
  - **Testers:** Testing team.
  - **Method:** Test cases stored in the TestRail environment will be executed.
  - **Timing:** After the completion of exploratory testing.

  Non Functional Test

  - **Purpose:** To evaluate the non-functional aspects of the Tankerkoenig API, including performance, usability, reliability, security, and localization.
  - **Testers:** Testing team.
  - **Method:** Test cases stored in the TestRail environment will be executed.
  - **Timing:** After the completion of functional testing.

Test Tree

- TestResponseKeys
  - Description:Verify that the API response contains expected keys.
- TestStationStructure
  - Description: Verify the structure of each station in the API response.
- TestSearchCenterStructure
  - Description: Verify the structure of the search center in the API response.
- TestApiVersionFormat
  - Description: Verify the format of the API version in the API response.
- TestLicenseFormat
  - Description: Verify the format of the license in the API response.
- TestStationCoordsStructure
  - Description: Verify the structure of the coordinates for each station in the API response.
- TestStationOpeningTimesStructure
  - Description: Verify the structure of the opening times for each station in the API response.
- TestComplaintSuccessful

        - Description: Verify that a complaint can be successfully submitted.
    - TestComplaintFailure
        - Description: Verify that a complaint submission fails with incorrect parameters.
    - TestStationData
        - Description: Verify the structure of station data in the API response.
    - TestStationDataFields
        - Description: Verify the presence and validity of fields in station data.
    - TestStationFuelPrices
        - Description: Verify the structure and validity of fuel prices for each station.
    - TestSearchCenter
        - Description: Verify the structure and presence of the search center in the API response.
    - TestLicense
        - Description: Verify the format and content of the license information in the API response.
    - TestJsonStructure
        - Description: Verify the JSON structure of the API response.
    - TestCordsStructure
        - Description: Verify the structure of the coordinates for each station in the API response.
    - TestFuelsStructure
        - Description: Verify the structure of the fuel data for each station in the API response.
    - TestCountAccuracy
        - Description: Verify the accuracy of the count values in the API response.
    - TestLicenseWebsite
        - Description: Verify the format and content of the license information in the API response.
    - TestApiVersionFormat
        - Description: Verify the format of the API version in the API response.
    - TestUniqueTimestamps
        - Description: Verify that timestamps in the API response are unique.

■ User Acceptance Test

- **Purpose:** End users or clients will test the Tankerkoenig API to ensure it can handle required tasks in real-world scenarios, according to specifications.
- **Testers:** Independent outside users.
- **Method:** End users will execute the test cases, performing tasks as they would in their day-to-day work. This includes navigating through the application, inputting data, and using the features and functions of the system.
- **Timing:** After the Functional and Non-Functional tests have been completed.

3. Execution Strategy

a. Test environment setup

- Environment Configuration: Ensure the test environment is configured to match the production environment as closely as possible.
- Data Setup: Prepare test data, including fuel price scenarios, gas station information, and other relevant data for testing the Tankerkoenig API.

b. Test Execution

- Unit Testing: Developers will perform unit tests using appropriate testing frameworks for backend code.
- Integration Testing: Test integration between different modules and components to ensure they work together correctly.
- System Testing: Test the entire Tankerkoenig API to verify that all features and functionalities work as expected.
- Acceptance Testing: Conduct acceptance tests with stakeholders to validate that the Tankerkoenig API meets the requirements.

c. Test Automation

- Regression Testing: Automate regression tests to ensure that new changes do not break existing functionality.
- Smoke Testing: Automate smoke tests to quickly verify that critical functionalities are working after each build.

d. **Defect Management**

- Defect Identification: Use tools like Jira to track and manage defects.
- Defect Prioritization: Prioritize defects based on severity and impact on the system.
- Defect Resolution: Developers will resolve defects, and testers will verify the fixes.

e. Reporting and Monitoring

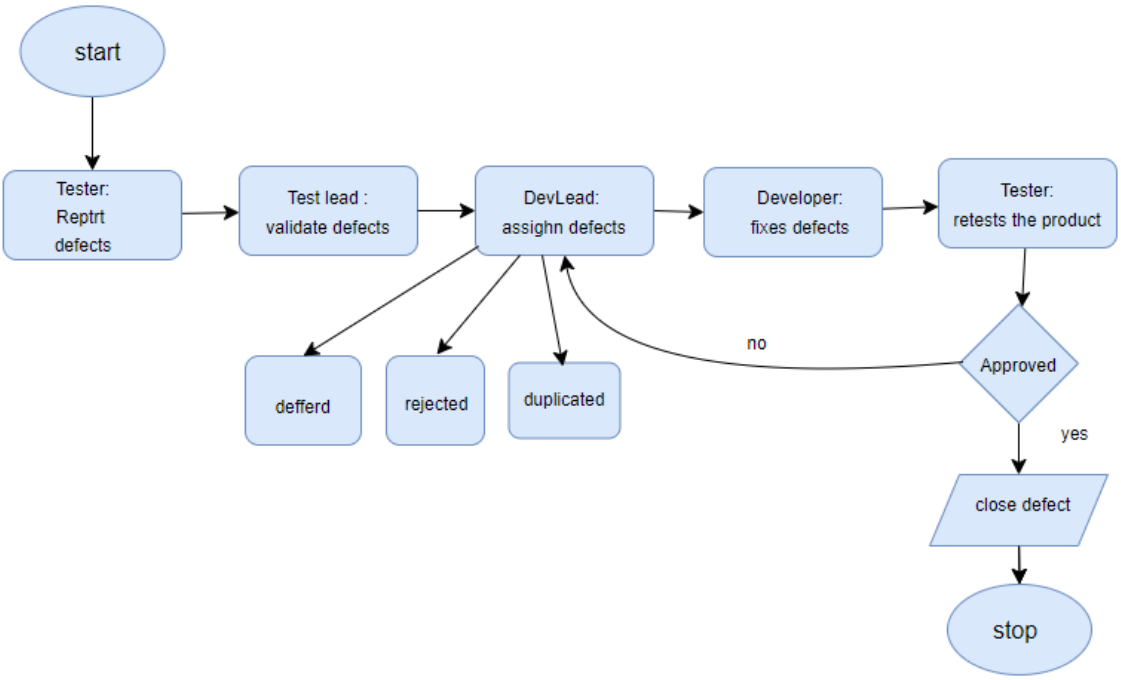
- Test Execution Reports: Generate and share test execution reports to track progress and identify issues.
- Monitoring: Monitor the test execution process to ensure that it is on track and meets the timeline.

f. Continuous Improvement

- Feedback Loop: Collect feedback from stakeholders and team members to identify areas for improvement.
- Process Optimization: Continuously optimize the testing process to improve efficiency and effectiveness.

g. **Defect tracking & Reporting**

- The presented flowchart illustrates the Defect Tracking Process:



4. TEST MANAGEMENT PROCESS

a. Test Planning

- Define Test Strategy: Define the overall testing strategy, including the use of Selenium Grid for parallel test execution.
- Identify Test Scenarios: Identify test scenarios for each layer of the application (infrastructure, logic, UI).
- Create Test Plan: Create a detailed test plan outlining the testing approach, resources, schedule, and deliverables.

b. Test Design

- Create Test Cases: Develop test cases for each test scenario, specifying the steps to be executed and the expected results.
- Implement Test Logic: Implement the test logic for each test case, including setting up the test environment and configuring Selenium Grid for parallel execution.

c. Test Execution

- Execute Tests: Run the tests using PyCharm and Selenium, utilizing Selenium Grid to run tests in parallel across different browsers and environments.
- Monitor Execution: Monitor the test execution process to ensure tests are running correctly and any issues are promptly addressed.

d. Test Reporting

- Generate Reports: Generate test reports using PyCharm or other reporting tools to track test execution results and identify any failures.
- Analyze Results: Analyze test results to identify trends, patterns, and areas for improvement.

e. Defect Management

- Identify Defects: Use PyCharm or a defect tracking tool to identify and log defects found during testing.
- Prioritize Defects: Prioritize defects based on severity and impact on the application.
- Resolve Defects: Work with the development team to resolve defects, retesting as necessary to verify fixes.

f. Test Closure

- Evaluate Testing: Evaluate the testing process to identify successes and areas for improvement.
- Document Lessons Learned: Document lessons learned from the testing process for future projects.
- Prepare Test Closure Report: Prepare a test closure report summarizing the testing activities, results, and any outstanding issues.

g. Continuous Improvement

- Review and Update: Regularly review and update the test plan, test cases, and test strategy based on feedback and lessons learned.
- Training and Development: Provide training and development opportunities for team members to improve their testing skills and knowledge.

5. Test Environment

Platforms:

- tankerkoenig API website
- Windows 11.

Devices:

- A range of desktop computers devices have the firefox,chrome and edge

Network Conditions:

- Tests were performed under various network speeds and conditions (Wi-Fi, 4G, 3G) to assess performance and reliability.

Test Tools:

- pycharm
- selenium
- selenium grid

6. Approvals

Name	Role	Signature
Mahmood Odeh	Project Management	
etsahe	Test Lead	
yair amon	Business Analyst	