Software Test Design

tankerkoenig

Mahmood Odeh

- Introduction
  - Document overview

    The Tankerkoenig API Software Test Description (STD) document outlines the test preparations, test cases, and test procedures for qualification testing of the Tankerkoenig API. This document is prepared in accordance with the guidelines outlined in the Software Requirements Specification (SRS) and Software Design Description (SDD) documents for the Tankerkoenig API.

  - Project overview

    The objective of this project is to automate the testing of the Tankerkoenig API, which provides information about gas stations in Germany. The API offers data such as fuel prices, station details, and search functionalities. The automation aims to ensure the correctness and reliability of the API's responses across different scenarios.

  - project references

#	Documents	Title
R1	STP	Software Test Plan
R2	Business Specification	Business Specification
R3	Requirements document	Requirements

- Tests basic preparations
  - Hardware basic preparation
    - **Browser Devices:**
      1- Variety of devices: Test on different browsers like chrome ,edge and firefox.

      2- Operating systems: Include Devices with various operating systems such as windows unix.

    - **Computers:**

1- System Specifications: Include computers with different specifications (RAM, CPU, GPU) to test the airbnb website interface under various system loads.

- **Network Hardware:**
  1- Routers: For testing network stability, latency, and performance impacts on the app's functionality.

○ Software basic preparation

- **Operating systems:** Include Devices with various operating systems such as windows unix.
- **Testing tools:** pycharm ,selenium and selenium grid.
- **The tested software:** tankerkoenig API website.

○ Data preparation

- TestResponseKeys:
  - Data Preparation: Ensure the API response contains all the expected keys.
- TestStationStructure:
  - Data Preparation: Prepare sample station data with the correct structure for testing.
- TestSearchCenterStructure:
  - Data Preparation: Prepare sample search center data with the correct structure for testing.
- TestApiVersionFormat:
  - Data Preparation: Prepare a sample API version string in the correct format for testing.
- TestLicenseFormat:
  - Data Preparation: Prepare a sample license string in the correct format for testing.
- TestStationCoordsStructure:
  - Data Preparation: Prepare sample station coordinates data with the correct structure for testing.
- TestStationOpeningTimesStructure:
  - Data Preparation: Prepare sample station opening times data with the correct structure for testing.
- TestComplaintSuccessful:
  - Data Preparation: Prepare valid parameters for a successful complaint submission.
- TestComplaintFailure:
  - Data Preparation: Prepare invalid parameters for a complaint submission to simulate a failure.
- TestStationData:

        - Data Preparation: Prepare sample station data with the correct structure for testing.
    - TestStationDataFields:
        - Data Preparation: Prepare sample station data with all required fields for testing.
    - TestStationFuelPrices:
        - Data Preparation: Prepare sample station fuel price data with the correct structure for testing.
    - TestSearchCenter:
        - Data Preparation: Prepare sample search center data with the correct structure for testing.
    - TestLicense:
        - Data Preparation: Prepare a sample license string in the correct format for testing.
    - TestJsonStructure:
        - Data Preparation: Prepare a sample JSON response with the correct structure for testing.
    - TestCordsStructure:
        - Data Preparation: Prepare sample station coordinates data with the correct structure for testing.
    - TestFuelsStructure:
        - Data Preparation: Prepare sample fuel data with the correct structure for testing.
    - TestCountAccuracy:
        - Data Preparation: Prepare sample station count data for testing.
    - TestLicenseWebsite:
        - Data Preparation: Prepare a sample license string in the correct format for testing.
    - TestUniqueTimestamps:
        - Data Preparation: Prepare sample timestamps data with unique values for testing.

Test scenarios

- Verify Response Keys
    - Open the Tankerkoenig API URL.
    - Ensure that the response contains keys: "stations", "searchCenter", "apiVersion", "timestamp", and "license".
- Verify Station Structure
    - Open the Tankerkoenig API URL.
    - For each station in the "stations" array:

            - Verify that the station object contains the expected fields and data types: "country" (str), "id" (str), "name" (str), "brand" (str), "street" (str), "postalCode" (str), "place" (str), "coords" (dict), "isOpen" (bool), "openingTimes" (list), "dist" (float), "fuels" (list), and "volatility" (int).
- Verify Search Center Structure
    - Open the Tankerkoenig API URL.
    - Verify that the "searchCenter" object exists and contains "lat" and "lng" keys.
- Verify API Version Format
    - Open the Tankerkoenig API URL.
    - Verify that the "apiVersion" string matches the format "\d+.\d+.\d+".
- Verify License Format
    - Open the Tankerkoenig API URL.
    - Verify that the "license" string starts with "CC BY".
- Verify Station Coordinates Structure
    - Open the Tankerkoenig API URL.
    - For each station in the "stations" array:
        - Verify that the "coords" object contains "lat" and "lng" keys.
- Verify Station Opening Times Structure
    - Open the Tankerkoenig API URL.
    - For each station in the "stations" array:
        - For each opening time in the "openingTimes" array:
            - Verify that the opening time object contains "days" (list) and "times" (list) keys, where each "times" object contains "open" (str) and "close" (str) keys.
- Successful Complaint Submission
    - Open the Tankerkoenig API URL.
    - Submit a complaint with correct parameters: "apikey", "id", "type", and "correction".
    - Verify that the API returns a status code of 200.
    - Verify that the API response contains {"ok": True}.
- Failed Complaint Submission
    - Open the Tankerkoenig API URL.
    - Submit a complaint with incorrect parameters, such as an incorrect "apikey".
    - Verify that the API returns a status code of 400.
    - Verify that the API response contains {"ok": False}.
- Incomplete Complaint Submission
    - Open the Tankerkoenig API URL.
    - Submit a complaint with missing parameters, such as omitting the "id" or "type".

    - Verify that the API returns a status code of 400.
    - Verify that the API response contains an error message indicating the missing parameters.
- Duplicate Complaint Submission
    - Open the Tankerkoenig API URL.
    - Submit a complaint with the same parameters as a previously submitted complaint.
    - Verify that the API returns a status code of 400.
    - Verify that the API response contains an error message indicating that a duplicate complaint is not allowed.
- Verify Station Data Structure:
    - Scenario: When retrieving station data from the API
        - Given the API response contains station data
        - Then the station data should have the correct structure with expected keys such as "id", "name", "brand", "coords", "isOpen", "dist", "fuels", and "volatility"
- Verify Station Data Fields:
    - Scenario: When retrieving station data from the API
        - Given the API response contains station data
        - Then each station should have all the required fields such as "id", "name", "brand", "coords", "isOpen", "dist", "fuels", and "volatility" with non-null values
- Verify Station Fuel Prices:
    - Scenario: When retrieving station data from the API
        - Given the API response contains station data
        - Then each station's fuel prices should have the correct structure with expected keys such as "category", "name", "price", and "lastChange", and "lastChange" should have a timestamp in the correct format
- Verify Search Center:
    - Scenario: When retrieving search center data from the API
        - Given the API response contains search center data
        - Then the search center data should have the correct structure with keys "lat" and "lng"
- Verify License Information:
    - Scenario: When retrieving license information from the API
        - Given the API response contains license information
        - Then the license information should start with "CC BY"
- Verify JSON Structure:
    - Scenario: When retrieving data from the Tankerkoenig API
        - Given the API response is received
        - Then the response should contain a "stations" array
        - And each station object in the array should have the following keys: "country", "id", "name", "brand", "street",

"postalCode", "place", "coords", "isOpen", "openingTimes", "dist", "fuels", and "volatility"

- Verify Coordinates Structure:
  - Scenario: When retrieving data from the Tankerkoenig API
    - Given the API response is received
    - Then each station object in the "stations" array should have a "coords" object
    - And the "coords" object should contain "lat" and "lng" keys
- Verify Fuels Structure:
  - Scenario: When retrieving data from the Tankerkoenig API
    - Given the API response is received
    - Then each station object in the "stations" array should have a "fuels" array
    - And each fuel object in the "fuels" array should have the keys: "category", "name", "price", and "lastChange"
    - And the "lastChange" object within each fuel object should have "amount" and "timestamp" keys
- Verify Station Count:
  - Scenario: When retrieving data from the Tankerkoenig API
    - Given the API response is received
    - Then the response should contain at least one station in the "stations" array
- Verify Count Accuracy:
  - Scenario: When retrieving data from the Tankerkoenig API
    - Given the API response is received
    - When calculating the total count of fuel types ("E5", "E10", "Diesel")
    - Then the total count should match the number of stations returned in the response
- Verify License Information:
  - Scenario: When retrieving data from the Tankerkoenig API
    - Given the API response is received
    - Then the license information should start with "CC BY"
    - And the license information should not contain the string "creativecommons.org"
- Verify API Version Format:
  - Scenario: When retrieving data from the Tankerkoenig API
    - Given the API response is received
    - Then the API version should be in the format "X.X.X" (e.g., "1.0.0")
    - And each part of the version number should be a non-negative integer
- Verify Unique Timestamps:

- Scenario: When retrieving data from the Tankerkoenig API
  - Given the API response is received
  - When collecting timestamps from the response
  - Then all timestamps should be the same, indicating that they are unique across all stations

Requirements Traceability

Requirement ID	Requirement Description	Test Case ID	Test Case Description
REQ001	**API response should contain specific keys**	**TestResponseKeys**	**Verify that the API response contains expected keys**
REQ002	**Station data in the API response should have a specific structure**	**TestStationStructure**	**Verify the structure of each station in the API response**
REQ003	**Search center data in the API response should have a specific structure**	**TestSearchCenterStructure**	**Verify the structure of the search center in the API response**
REQ004	**API version in the API response should have a specific format**	**TestApiVersionFormat**	**Verify the format of the API version in the API response**
REQ005	**License information in the API response should have a specific format**	**TestLicenseFormat**	**Verify the format of the license in the API response**

REQ006	**Coordinates for each station in the API response should have a specific structure**	**TestStationCoordsStructure**	**Verify the structure of the coordinates for each station in the API response**
REQ007	**Opening times for each station in the API response should have a specific structure**	**TestStationOpeningTimesStructure**	**Verify the structure of the opening times for each station in the API response**
REQ008	**API should allow successful submission of complaints**	**TestComplaintSuccessful**	**Verify that a complaint can be successfully submitted**
REQ009	**API should handle failed complaint submissions correctly**	**TestComplaintFailure**	**Verify that a complaint submission fails with incorrect parameters**
REQ010	**Station data in the API response should have a specific structure**	**TestStationData**	**Verify the structure of station data in the API response**
REQ011	**Station data should have all required fields**	**TestStationDataFields**	**Verify the presence and validity of fields in station data**
REQ012	**Fuel prices for each station should have a specific structure and validity**	**TestStationFuelPrices**	**Verify the structure and validity of fuel prices for each station**

REQ013	**API response should contain search center information**	**TestSearchCenter**	**Verify the structure and presence of the search center in the API response**
REQ014	**License information in the API response should have a specific format and content**	**TestLicense**	**Verify the format and content of the license information in the API response**
REQ015	**API response should have a specific JSON structure**	**TestJsonStructure**	**Verify the JSON structure of the API response**
REQ016	**Station coordinates in the API response should have a specific structure**	**TestCordsStructure**	**Verify the structure of the coordinates for each station in the API response**
REQ017	**Fuel data for each station in the API response should have a specific structure**	**TestFuelsStructure**	**Verify the structure of the fuel data for each station in the API response**
REQ018	**Count values in the API response should be accurate**	**TestCountAccuracy**	**Verify the accuracy of the count values in the API response**
REQ019	**License information in the API response should not contain certain URLs**	**TestLicenseWebsite**	**Verify the format and content of the license information in the API response**

REQ020	**Timestamps in the API response should be unique**	**TestUniqueTimestamps**	**Verify that timestamps in the API response are unique**