@author Bhimashankar Teli

Test Plan: Schema Validation for All APIs

1. Introduction

This test plan outlines the approach, scope, objectives, and methodology for schema validation of all APIs. The primary goal is to ensure the API responses conform to the defined schema and specifications, ensuring data consistency, correctness, and adherence to contracts.

---

2. Objective

- Validate the structure, data types, required fields, and constraints of API responses against the defined schema.
- Identify deviations between the actual response and the expected schema (**Not validating any values here**).
- Automate schema validation for consistency across all API endpoints.

---

3. Scope

- **In Scope**:
    - All REST APIs in the application.
    - Validation of JSON schemas for success and error responses.
    - Testing for different HTTP methods (GET, POST, PUT, DELETE).
    - Validation across various environments (development, QA, staging, production).
- **Out of Scope**:
    - Performance and load testing.
    - Validation of XML or other formats (unless explicitly required).

---

4. Test Approach

1. **Identify API Endpoints**:
    - Document all API endpoints, methods, and expected response payloads.
    - Refer to API documentation for schema definitions (Swagger specifications).
2. **Tool Selection**:
    - Use tools like **Postman**, **Rest Assured**, or **JSON Schema Validator**.

    - For automation, integrate schema validation with testing frameworks like **JUnit/TestNG** in Java or **Pytest** in Python.

3. **Schema Validation Workflow**:
    - Retrieve schema definitions from Swagger or JSON schema files.
    - Compare the actual API response against the schema.
    - Log and report deviations or validation errors.
4. **Test Cases**:
    - Validate success response schema (e.g., 200 OK).
    - Validate error response schema (e.g., 400, 401, 404, 500).
    - Validate nested objects and arrays in the response payload.
    - Test mandatory fields, data types, and constraints (e.g., enum values).
5. **Automation**:
    - Automate schema validation for continuous testing during development.
    - Integrate with CI/CD pipelines to catch schema-related issues early.

---

5. Test Environment

- **Required Resources**:
    - API endpoints (accessible URLs).
    - JSON schema files or Swagger documentation.
    - Tools: Postman, Rest Assured, JSON Schema Validator, JMeter (if needed).
    - Test automation framework.
- **Environments**:
    - Development
    - QA/Staging
    - Production (for monitoring)

6. **Test Data**

- **Sources**:
    - API response payloads.
    - Example requests and responses from Swagger.

- **Test Data Creation**:
    - Mock data for validating various scenarios (e.g., edge cases, invalid payloads).
    - Use real data from lower environments (development or QA).

7. Test Scenarios

Positive Test Scenarios:

1. Schema validation for successful responses (200 OK).
2. Validation of nested objects and arrays.
3. Validation of data types and format (e.g., email, date).

Negative Test Scenarios:

1. Schema validation for error responses (400 Bad Request, 500 Internal Server Error).
2. Missing or additional fields in the response.
3. Invalid data types (e.g., string instead of integer).

8. Test Execution

- **Manual Testing**:
    - Use Postman or Swagger to validate schemas manually.
- **Automation**:
    - Write automation scripts using Rest Assured (Java) or similar frameworks.
    - Automate validation for all endpoints in one suite (Create Jenkins jobs. Schedule it twice run every day).