API Test Plan: JSONPlaceholder

API Under Test: JSONPlaceholder (https://jsonplaceholder.typicode.com/)

Objective:

The primary objective of this test plan is to ensure the functionality, reliability, and performance of the key endpoints of the JSONPlaceholder API. This plan will cover all aspects of testing including functional, performance, and security testing.

Assumptions

- No authentication is required for this API.
- The API returns simulated responses (data isn't actually persisted).
- HTTP status codes and response formats follow REST conventions.

Scope & Endpoints to Test

The test plan focuses on the following key endpoints for the /posts resource:

- GET /posts: Retrieve a list of all posts.
- GET /posts/{id}: Retrieve a single post by its ID.
- POST /posts: Create a new post.
- PUT /posts/{id}: Update an existing post.
- DELETE /posts/{id}: Delete a post.

Types of Testing

This plan will include the following types of API testing:

- **Functional Testing:**
  - **Positive Scenarios:** Verify that each endpoint performs its intended function correctly with valid inputs.
  - **Negative Scenarios:** Validate how the API handles invalid inputs, incorrect data formats, and missing parameters.
  - **Edge Cases:** Test boundary conditions, such as the first and last items in a list, or very large data payloads.
- **Performance Testing:**
  - **Load Testing:** Simulate a large number of concurrent users to see how the API behaves under expected load.
  - **Stress Testing:** Push the API beyond its normal operating limits to determine its breaking point and how it recovers.
- **Security Testing:**
  - **Authentication & Authorization:** Given the assumption of no authentication, this would focus on ensuring the API rejects requests that attempt to use credentials.
  - **Input Validation & Injection:** Attempt to pass malicious data, such as SQL injection strings or scripts, to ensure the API handles them securely.

- **Regression Testing:**
  - Re-running existing functional test cases after changes or updates to the API to ensure no new bugs were introduced.
- **Concurrency Testing:**
  - Test how the API handles multiple requests from the same user or different users simultaneously to prevent race conditions or data corruption.
- **Rate Limit Testing:**
  - Verify that the API correctly enforces rate limits (if any are configured) by sending a high volume of requests within a short period and checking the response.
- **Error Handling Testing:**
  - Validate that the API returns appropriate and clear error messages and status codes for various failure scenarios, such as bad requests, unauthorized access, or server-side issues.
- **Data Validation Testing:**
  - Ensure that the API correctly validates incoming data payloads against the expected schema, rejecting invalid data and providing meaningful error messages.
- **Third-Party Integration Testing:**
  - While JSONPlaceholder is a standalone API, in real-world scenarios, this would involve testing the interaction and data exchange with any external services or APIs.

Tools Used

- **Postman & Rest-Assured:** Used for manual and automated functional testing.
- **Gatling:** Utilized for load and performance testing.
- **Burp Suite:** Employed for basic security vulnerability testing.
- **Allure:** Used for generating comprehensive and interactive test reports.

Key Validations to Perform

Each test case will include validations for the following:

- **HTTP Status Codes:**
  - 200 OK: For successful GET and PUT requests.
  - 201 Created: For successful POST requests.
  - 204 No Content: For successful DELETE requests.
  - 400 Bad Request: For invalid request formats or data.
  - 404 Not Found: When a resource does not exist.
  - 500 Internal Server Error: For unexpected server issues.
- **Response Body:**
  - Verify the content of the response body, including data types, structure, and values.
  - Ensure the JSON format is correct.
  - Validate that the response contains all expected fields and that no unexpected fields are present.
- **Response Headers:**
  - Validate important headers such as Content-Type.
  - Verify that caching headers (Cache-Control, ETag) are correctly implemented for GET requests.

- **Response Time:**
  - Check that the API responds within acceptable performance limits.
  - Validate that the average response time under load remains stable and within the defined threshold.
- **Data Consistency:**
  - Since the data is not persisted, this validation will ensure that a POST or PUT request returns a response body that reflects the sent data, including a new simulated ID for POST requests.
- **Schema Validation:**
  - Verify that the response schema conforms to the API's contract, including data types, nullability, and field constraints.