1. Introduction

API Testing is a type of software testing that verifies APIs to ensure they meet functional, performance, security, and reliability requirements. Unlike UI testing, API testing focuses on the business logic, data validation, and communication between software components directly at the interface level.

---

2. Key Definitions

2.1 API (Application Programming Interface)

An API is a set of rules and protocols that allows one application to communicate with another. APIs expose endpoints that accept requests and return responses.

2.2 Web Service

A web service is a type of API that works over a network (usually HTTP) to provide a service to other applications. Web services can be SOAP-based or RESTful.

Difference:

- **Web Service**: Must operate over a network and use standardized messaging (SOAP/REST).
- **API**: Broader term; can include libraries, OS-level APIs, or hardware APIs.

Remember:
“Every Web Service is an API, but not all APIs are Web Services.”

Slogan Explanation:

- All web services are APIs because they expose endpoints and allow interaction.

- Not all APIs are web services, e.g., a local Java API or operating system API.

2.3 REST (Representational State Transfer)

A widely used architectural style for web APIs. REST APIs are stateless and typically use HTTP methods like GET, POST, PUT, DELETE to perform actions on resources.

2.4 SOAP (Simple Object Access Protocol)

A protocol-based API standard that uses XML messaging over HTTP or other protocols. SOAP APIs are strict, standardized, and include built-in error handling.

2.5 Endpoint

The URL or URI where an API or web service can be accessed. Example: https://api.example.com/users.

2.6 Request & Response

- **Request**: Call made by a client to an API endpoint, including HTTP method, headers, parameters, and sometimes a body.
- **Response**: Data returned by the API after processing a request. Includes status code, headers, and response body (JSON, XML, etc.).

2.7 HTTP Methods

- **GET**: Retrieve data from the server.
- **POST**: Send data to the server to create a resource.
- **PUT**: Update an existing resource.
- **PATCH**: Partially update a resource.
- **DELETE**: Remove a resource.

2.8 Status Codes

- **2xx**: Success (e.g., 200 OK, 201 Created)
- **4xx**: Client Errors (e.g., 400 Bad Request, 401 Unauthorized)
- **5xx**: Server Errors (e.g., 500 Internal Server Error)

2.9 JSON & XML

- **JSON (JavaScript Object Notation)**: Lightweight, human-readable data format commonly used in REST APIs.
- **XML (eXtensible Markup Language)**: Structured markup language commonly used in SOAP APIs.

---

3. API Parameters

3.1 Path Parameters

Variables in the URL path to identify specific resources.
Example:

GET https://api.example.com/users/{userId}

{userId} is a path parameter.

3.2 Query Parameters

Optional parameters appended to the URL to filter or modify the response.
Example:

GET https://api.example.com/users?role=admin&active=true

3.3 Headers

Metadata sent with requests/responses, e.g., Content-Type, Authorization.

3.4 Cookies

Small pieces of data stored by the server and sent back with requests, often for session management.

---

4. Real-Time API Examples

Twitter API (REST API)

GitHub API (REST API)

Google Maps API

Local Java API Example

4.2 Query & Path Parameter Example

Endpoint:
https://api.openweathermap.org/data/2.5/weather?q=London&appid=YOUR_API_KEY

- q → Query parameter for city
- appid → API key for authentication

4.3 Header Example

GET /user HTTP/1.1

Host: api.example.com

Authorization: Bearer <token>

Content-Type: application/json

4.4 Cookie Example

GET /profile HTTP/1.1

Host: api.example.com

Cookie: session_id=abc123xyz

---

5. API Testing Types

1. **Functional Testing** – Ensures API works as intended.
2. **Load/Performance Testing** – Evaluates API performance under heavy load.
3. **Security Testing** – Verifies authentication, authorization, and data protection.
4. **Validation Testing** – Checks if API returns correct and complete data.
5. **Error Handling Testing** – Confirms proper handling of invalid requests or server errors.

---

6. Key Components of API Testing

1. **Request URL/Endpoint**
2. **Request Method**
3. **Request Headers**
4. **Path Parameters**
5. **Query Parameters**
6. **Request Body**
7. **Response Body**
8. **Response Headers**
9. **Cookies**
10. **Status Code**

---

7. Tools for API Testing

- **Postman** – GUI for sending requests and validating responses.
- **SoapUI** – SOAP and REST API testing.

- **Rest Assured** – Java library for REST API automation.
- **JMeter** – Performance and load testing.

---

8. API Testing Process

1. Understand API requirements.
2. Identify endpoints, HTTP methods, parameters, headers, and cookies.
3. Define test scenarios (positive, negative, edge cases).
4. Send requests via tools or scripts.
5. Validate responses: status codes, headers, body, cookies.
6. Automate tests using frameworks like Postman Collections or Rest Assured.
7. Log defects with detailed evidence.

---

9. Best Practices

- Refer to API documentation before testing.
- Test with valid and invalid inputs.
- Validate status codes, headers, cookies, and response body.
- Include authentication and authorization checks.
- Automate repetitive tests for regression.
- Maintain independent and reusable tests.