**🚀 API Testing Interview Notes (Expanded Version)**

**1️ What is an API?**

* **API (Application Programming Interface)**: A bridge that allows two software systems to communicate with each other.
* Example: A weather app uses an API to fetch weather data from a server.

**2️ What is API Testing?**

* **API Testing**: Validates the functionality, reliability, performance, and security of APIs.
* Focuses on sending requests to API endpoints and verifying the responses.
* Tests if the API returns the correct data, handles errors properly, and performs under load.

**3️ Key HTTP Methods in API Testing**

| **Method** | **Purpose** | **Example Use Case** |
| --- | --- | --- |
| **GET** | Retrieve data | Get a list of users/books |
| **POST** | Send new data | Create a new order/account |
| **PUT** | Update entire data | Replace a user's profile |
| **PATCH** | Update partial data | Update only the user's email |
| **DELETE** | Remove data | Delete a user/order |

**🧐 PUT vs PATCH**

* **PUT**: Replaces the entire resource with new data.
* **PATCH**: Updates only specific parts of the resource.

**5️ Authentication vs. Authorization**

| **Concept** | **What It Means** | **Purpose** |
| --- | --- | --- |
| **Authentication** | Verifies "Who you are" | Example: Logging into a website. |
| **Authorization** | Verifies "What you can do" | Example: Allowing a user to access specific files. |

**💡 Why APIs use Authorization instead of Authentication?**

* **Authentication** is about verifying identity, while **Authorization** is about permissions.
* APIs often deal with sensitive resources, so they must verify **what a user is allowed to access**, which is why they focus on **authorization**.

**6️ Headers in API Requests**

* **Headers**: Metadata sent with an API request to provide information about the request.
* Common Headers:
  + **Authorization**: Bearer <token> for authenticated requests.
  + **Content-Type**: Specifies the format of the request body (e.g., JSON, XML).
  + **Accept**: Tells the server what response format to return.

**7️ Request Parameters (How to Send Data in a GET Request)**

| **Type** | **Where it Appears** | **Example** |
| --- | --- | --- |
| **Query Parameters** | In URL after ? and separated by & | /books?type=fiction&limit=10 |
| **Path Parameters** | Part of the URL path | /books/:bookId (e.g., /books/123) |
| **Headers** | Metadata sent with the request | Authorization: Bearer <token> |
| **Cookies** | Stored in the browser, sent automatically | User session tokens |

**❓ Can we send data in the body of a GET request using Axios or Fetch?**

* **No**, GET requests do not have a body. Data is sent using **query parameters**, **headers**, or **cookies**.

**8️ API Authentication (How It Works)**

1. **Register API Client**: Send POST request with clientName and clientEmail.
2. **Receive Access Token**: The server returns a token (like a temporary password).
3. **Include Token in Headers**: Add the token in the request header:

Authorization: Bearer <your-token>

**🎯 Essential Interview Questions**

1. **How to handle authentication and authorization in API testing?**

*Authentication is done by passing API keys, OAuth tokens, or JWT in the headers (e.g., "Authorization: Bearer <token>"). Authorization is tested by trying to access restricted endpoints with different user roles, ensuring access control is properly enforced.*

1. **How to validate API responses?**

*API responses are validated by checking status codes like 200 for success, 404 for not found, and 500 for server errors. The response body is validated for correct fields, data types, and values. Headers like Content-Type and Authorization are also checked to ensure correctness.*

1. **How to perform error handling in API testing?**

*Error handling is tested by sending invalid inputs, such as missing or incorrect data, and ensuring the API returns appropriate error messages and status codes (e.g., 400 or 500). Edge cases like empty or large payloads are also tested for proper handling.*

1. **Parts of a Website URL**

Example: https://blog.example.com/articles/web-development?topic=api

* *Protocol* (https://): Specifies the communication method, in this case, secure HTTP.
* *Subdomain* (blog): A subdivision of the main domain for specific sections (e.g., blog, shop).
* *Domain* (example.com): The unique address of the website.
* *Path* (/articles/web-development): Identifies the specific resource or page.
* *Query String* (?topic=api): Passes additional data to the server, often for filtering or searching.

1. **Encryption in HTTP & HTTPS**

*"HTTP is not secure as it sends plain text data. HTTPS encrypts data using* ***SSL/TLS****, making it secure. The encryption process starts with a handshake, where the client and server share keys, and then data is encrypted using symmetric encryption."*

1. **What is SSL/TLS in HTTPS?**

*"SSL (Secure Sockets Layer) and TLS (Transport Layer Security) secure HTTPS connections. They use* ***public-key cryptography*** *for the handshake to exchange keys, and then* ***symmetric encryption*** *is used for secure, fast data transfer."*

**Types of APIs**

**REST API**

*"REST API uses HTTP methods like GET, POST, PUT, and DELETE to communicate between client and server. It is stateless, lightweight, and widely used in web and mobile applications. Data is usually exchanged in JSON or XML format."*

**SOAP API** Simple Object Access Protocol

*"SOAP API is a protocol that uses XML to send and receive messages. It is more secure, follows strict standards, and is used in financial services and payment gateways like PayPal."*

**GraphQL API**

*"GraphQL allows clients to request only the specific data they need from a single endpoint. It avoids over-fetching or under-fetching of data and is used by companies like GitHub and Facebook."*

**WebSocket API**

*"WebSocket API provides real-time, two-way communication over a persistent connection. It is used in live chat apps, stock price updates, and online gaming where instant updates are required."*