https://chatgpt.com/share/6847fa22-b88c-800d-b5dd-8484479d038a

**API Testing (Postman)-Introduction**

Great! Let's go through **each topic** from your image **with simple explanations, examples, and answers** — especially tailored for **beginners in API Testing using Postman**.

### ✅ **1. What is an API and API Testing?**

* **API (Application Programming Interface)** allows software to talk to other software.
* **API Testing** checks if the API works correctly, returns expected responses, handles errors, etc.

🔸 Example: A weather app requests data from a weather API → API Testing checks if the response includes correct temperature, location, etc.

**TYPES OF API**

1. SOAP
2. JSON RPC
3. XML RPC
4. REST

### ✅ **2. Difference between API Testing and Unit Testing**

| **Unit Testing** | **API Testing** |
| --- | --- |
| Tests individual functions/methods | Tests interaction between systems |
| Done by developers | Done by QA testers |
| Does not involve HTTP/requests | Uses HTTP requests/responses |

### ✅ **3. What is Web Service?**

* A **Web Service** is an API that works over a network (usually HTTP).
* **REST** and **SOAP** are two types.

### ✅ **4. Types of Defects in API Testing**

* Wrong status codes (404 instead of 200)
* Invalid response data
* Missing fields (e.g., missing “email” in response)
* Unauthorized access
* Incorrect headers

### ✅ **5. Learn What is XML and JSON**

* **XML** and **JSON** are data formats used in APIs.

🔸 Example JSON:

{

"name": "Alice",

"age": 25

}

🔸 Example XML:

<user>

<name>Alice</name>

<age>25</age>

</user>

### ✅ **6. URI and URL**

* **URL (Uniform Resource Locator):** Exact address of a web page or API.
* **URI (Uniform Resource Identifier):** A broader term that includes URLs.

🔸 Example:  
https://api.example.com/users/1 – URL of user with ID 1.

### ✅ **7. Tools for API Testing**

* **Postman** – main tool for sending requests and checking responses.
* **Swagger UI** – view and test API documentation.
* **Newman** – run Postman collections from the command line.

### ✅ **8. SOAP Introduction**

* **SOAP (Simple Object Access Protocol)** is a strict XML-based protocol.
* Used in older enterprise systems.
* Heavier than REST.

### ✅ **9. Why Learning REST is Important?**

* REST is **simpler, faster**, and **widely used** in modern APIs.
* Uses **standard HTTP methods** (GET, POST, etc.)

### ✅ **10. Difference Between SOAP and REST**

| **Feature** | **SOAP** | **REST** |
| --- | --- | --- |
| Format | XML only | JSON, XML |
| Speed | Slower | Faster |
| Protocol | Strict | Flexible |

### ✅ **11. REST Constraints (Rules)**

* **Stateless**: Server doesn’t remember previous requests.
* **Cacheable**: Responses can be stored.
* **Uniform Interface**: Same way to access resources.
* **Client-Server**: Separation of UI and backend.

### ✅ **12. POSTMAN Basics**

* Install Postman
* Create requests (GET, POST, etc.)
* Create **collections** (group of requests)

### ✅ **13. Environments in Postman**

* Use **variables** (like {{baseUrl}}) to switch between staging and production.

🔸 Example:

{{baseUrl}}/users

### ✅ **14. Authorization in Postman**

* Set tokens, API keys, basic auth in the **Authorization tab**.

### ✅ **15. Assertions/Test Scripts in Postman**

* Written in JavaScript under the **Tests tab**.
* Example:

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

### ✅ **16. Newman – Command Line Tool**

* Run Postman collections from terminal.

🔸 Example Command:

newman run my\_collection.json

### ✅ **17. Run Postman Collections on Docker, Jenkins**

* Used in automation pipelines (CI/CD).
* Jenkins + Newman = automated API tests after every code push.

### ✅ **18. JSON Schema Validation**

* Confirms that API response matches expected structure.

🔸 Example:

const schema = {

type: "object",

required: ["name", "email"]

};

pm.test("Validate schema", function () {

pm.response.to.have.jsonSchema(schema);

});

### ✅ **19. Data-Driven Testing with CSV**

* Upload CSV file with multiple inputs → Postman tests each input.
* Done using **Collection Runner**.

### ✅ **20. Exploring More Postman Features**

* **Mock Server:** Fake API for frontend testing.
* **Stubbing:** Simulate real APIs.

### ✅ **21. Testing GraphQL, WebSockets**

* Postman also supports these advanced protocols.

### ✅ **22. Writing API Test Cases & Test Plan**

* Document what you want to test:
  + URL
  + Method (GET/POST)
  + Expected Status
  + Expected Response

### ✅ **23. Advanced JSON Schema**

* Use nested schemas, arrays, and types for deep validation.

### ✅ **24. LIVE API Testing Example**

Try testing this public API:  
🔸 https://jsonplaceholder.typicode.com/users

* Method: **GET**
* Expected Status: **200 OK**
* Check name, email in response.

## 🧪 What Do We Test in API Testing? (Key Features)

| **🔍 Feature** | **✅ What to Check** |
| --- | --- |
| **Status Code** | Should be 200 for success, 400/404 for errors |
| **Response Body/Data** | Verify fields, data types, values, structure |
| **Response Time** | Should be fast (e.g., under 2 seconds) |
| **Error Messages** | Should be meaningful and accurate (e.g., "User not found") |
| **Authentication** | Valid token/auth key required for protected APIs |
| **Headers** | Check for correct headers (Content-Type: application/json) |
| **Data Accuracy** | Compare database data and API response |
| **Data Formats** | JSON/XML should be correct |
| **Schema Validation** | Structure of response should match expected format |
| **Boundary Tests** | Test with empty values, large inputs, etc. |

## 🧭 How to Test an API (Step-by-Step in Postman)

### 🔹 Step 1: Open Postman and Create a Request

* Choose **method**: GET, POST, PUT, DELETE
* Enter **API URL**: e.g., https://api.example.com/users

### 🔹 Step 2: Add Headers (if needed)

* Example:
  + Content-Type: application/json
  + Authorization: Bearer <token>

### 🔹 Step 3: Add Request Body (for POST/PUT)

json

CopyEdit

{

"name": "Alice",

"email": "alice@example.com"

}

### 🔹 Step 4: Click **Send** and View Response

* Check:
  + **Status Code**: 200 OK?
  + **Response Body**: Expected data?
  + **Headers**: Correct content type?

### 🔹 Step 5: Add Tests (in **Tests tab**)

Example:

js

CopyEdit

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

## 🔧 Postman Components You Use During API Testing

| **Component** | **Description** |
| --- | --- |
| **Request Method** | GET, POST, PUT, DELETE etc. |
| **Request URL** | API endpoint like https://api.example.com/login |
| **Headers Tab** | Add auth tokens, content-type, etc. |
| **Body Tab** | Add JSON body for POST/PUT |
| **Params Tab** | Add query parameters (?userId=1) |
| **Authorization Tab** | Add token, API key, etc. |
| **Tests Tab** | Write test scripts in JavaScript |
| **Pre-request Script Tab** | Set up variables or tokens before sending request |
| **Collections** | Group multiple API requests for a project |
| **Environment** | Manage different values for dev, test, prod |
| **Runner** | Run a collection with multiple test cases |
| **Newman** | CLI tool to run Postman tests in terminal |
| **Monitor** | Schedule API tests (Postman cloud feature) |

## 📝 Example Test Case for API

| **Field** | **Example** |
| --- | --- |
| **API Name** | Get User Details API |
| **Method** | GET |
| **URL** | https://api.example.com/users/1 |
| **Expected Status** | 200 OK |
| **Expected Fields** | name, email, age |
| **Test** | Assert that name = "Alice" |

Excellent — let’s go step-by-step so you fully understand **Environment Variables** and **Authorization in Postman** (very important topics for API Testing).

# ✅ 1️⃣ **Environment Variables in Postman**

### 🔶 What are Environment Variables?

* Variables you define inside Postman to store values.
* They help you avoid hard-coding values like base URLs, tokens, API keys, etc.
* You can switch environments easily (Ex: dev, staging, production).

### 🔶 Why do we use them?

* Reusability: You don’t need to change your requests again and again.
* Easy Maintenance: Update variables in one place.
* Multiple Environments: Same requests work in different environments.

### 🔶 Example:

Suppose you're testing on different servers:

| **Environment** | **Base URL** |
| --- | --- |
| Dev | https://dev.api.com |
| Staging | https://staging.api.com |
| Production | https://api.com |

Instead of hard-coding URLs in each request, you define a variable:

{{base\_url}}

And set:

* base\_url = https://dev.api.com (in Dev environment)

Your request becomes:

{{base\_url}}/users

Postman will replace {{base\_url}} automatically based on the environment.

### 🔶 How to set Environment Variables:

1. In Postman, go to "Environments" (top right corner).
2. Create New Environment.
3. Add variables like base\_url, token, etc.
4. Save.
5. Select that environment before sending requests.

### 🔶 Example of variables:

| **Variable** | **Initial Value** | **Current Value** |
| --- | --- | --- |
| base\_url | https://api.com | https://api.com |
| token | abcdef123456 | abcdef123456 |

# ✅ 2️⃣ **Authorization in Postman**

### 🔶 Why do we need Authorization?

* To prove to the API that you’re allowed to access it.
* Prevents unauthorized access.

### 🔶 Types of Authorization in Postman

| **Type** | **Usage** | **Example** |
| --- | --- | --- |
| No Auth | No authentication needed | Public APIs |
| API Key | Use a key as query parameter or header | ?api\_key=xxx |
| Bearer Token | Send token in header | Authorization: Bearer <token> |
| Basic Auth | Username & Password (Base64 encoded) | Authorization: Basic <base64> |
| OAuth 2.0 | Modern token-based auth | Google, Facebook APIs |

### 🔶 Common Example (Bearer Token):

**Request Headers:**

Authorization: Bearer {{token}}

Here, token can be stored in environment variables!

### 🔶 How to set Authorization in Postman:

1. Open your request.
2. Go to the **Authorization** tab.
3. Choose type (e.g. Bearer Token).
4. Enter your token (or use {{token}} if stored in environment variables).
5. Done — Postman will add the header automatically.

# ✅ ✅ **Quick Summary Table**

| **Concept** | **Why?** | **How?** |
| --- | --- | --- |
| Environment Variables | Store values like URLs, tokens | Use {{variable\_name}} |
| Authorization | Prove identity | Use Authorization tab or Headers |