Assignment-3

Introduction to Postman

1️⃣ What is Postman, and why is it used in API testing?

Postman is a popular API testing tool that allows developers and testers to send requests to APIs, inspect responses, and automate testing. It provides a user-friendly interface to interact with APIs without writing code, making it easier to debug, document, and automate API workflows. Postman is widely used for API testing because it supports various request methods, authentication mechanisms, scripting, and automation.

2️⃣ How do you send a GET request in Postman?

To send a GET request in Postman:

1. Open Postman and select **GET** from the request method dropdown.
2. Enter the API URL in the request field (e.g., https://api.example.com/users).
3. Click **Send**, and Postman will retrieve the response from the server.
4. The response, including status code, headers, and body, will appear in the response section.

3️⃣ What is the purpose of the Authorization tab in Postman?

The **Authorization** tab in Postman is used to include authentication credentials in API requests. Many APIs require authentication to ensure secure access. Postman supports multiple authentication types, such as:

- **API Key**
- **Basic Auth** (username and password)
- **Bearer Token** (e.g., JWT)
- **OAuth 2.0**
- **Digest Auth**

4️⃣ How can you pass query parameters in Postman?

Query parameters are used to filter or modify API requests (e.g., searching, sorting). To pass them in Postman:

1. Click the **Params** tab below the request URL field.
2. Enter the **key-value pairs** for the parameters (e.g., name=John&age=30).
3. Postman automatically appends them to the request URL (https://api.example.com/users?name=John&age=30).
4. Click **Send** to execute the request.

5️⃣ What are Postman Collections, and how do they help in API testing?

Postman Collections are groups of saved API requests organized in folders. They help in API testing by:

- Storing and reusing API requests.
- Organizing endpoints based on functionality.
- Allowing automated execution using test scripts.
- Enabling team collaboration by sharing API workflows.

6️⃣ What is the use of environment variables in Postman?

Environment variables in Postman allow users to store and reuse dynamic values across requests, reducing redundancy and improving flexibility. They are used for:

- Storing API base URLs ({{base_url}}).
- Managing authentication tokens ({{auth_token}}).
- Creating different environments (e.g., **dev**, **staging**, **production**).
- Simplifying request updates by changing variables instead of hardcoded values.

7️⃣ How do you validate API responses in Postman using assertions?

Postman allows writing **test scripts** using JavaScript to validate API responses. These assertions ensure that the API behaves as expected.
To add assertions:

1. Open the **Tests** tab in Postman.
2. Write JavaScript tests, such as:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response contains user ID", function () {
    pm.expect(pm.response.json()).to.have.property("id");
});
```

3. Click **Send**, and the results will appear under the **Test Results** section.

8️⃣ What is the difference between Params, Headers, and Body in Postman?

 Params: Used for query parameters (e.g., ?name=John) to filter or modify requests.

 Headers: Contain metadata (e.g., Content-Type, Authorization) for the request.

 Body: Holds the actual data for **POST, PUT, PATCH** requests (e.g., JSON payload).

9️⃣ How can you automate API requests in Postman?

API requests in Postman can be automated using:

- **Tests and Assertions**: Automatically validate responses.
- **Pre-request Scripts**: Execute logic before sending requests (e.g., generate tokens).
- **Postman Collection Runner**: Run multiple requests in sequence.
- **Newman (CLI Tool)**: Automate API tests in CI/CD pipelines.

10 What is Newman in Postman, and how is it useful?

Newman is Postman's **command-line tool** used to run collections outside the Postman app. It is useful for:

- **CI/CD Integration**: Running API tests in Jenkins, GitHub Actions, etc.
- **Automated Testing**: Executing test collections programmatically.
- **Scheduled Testing**: Running tests periodically for API health checks.