Q. What is Postman, and what are its primary features?

Postman is a collaboration platform for API development and testing.

It provides a user-friendly interface to build, test, document, and share APIs. Key features are:

- **Collections**: Organize requests into groups.
- **Environment Variables**: Manage different setups for development, staging, and production environments.
- **Pre-request Scripts and Tests**: Automate tests and pre-processing of requests.
- **Monitors**: Schedule and automate API testing.
- **Postman Console**: Debug and view request logs.
- **API Documentation**: Auto-generate documentation from collections.

Q. Explain the core components of a Postman request.

A Postman request typically consists of some components:

- Method: The HTTP method used for the request (e.g., GET, POST, PUT, DELETE)
- URL: The complete URL of the API endpoint
- Headers: Key-value pairs of information sent along with the request (e.g., Authorization, Content-Type)
- Body: The data sent in the request, usually in JSON, XML, or form-data format

Q. What is API testing?

Testing the interface between two applications or programs is called as API testing.

Q. Why API testing is necessary?

In order to make sure the connection or the interface is working as expected with respect to the functionality, performance, security etc…

When API testing has to be performed?

i. In the absence of user interface.
ii. API testing is done before manual testing.
iii. API testing has to be done in order to test all the web service.

Q. How do you create a new request in Postman?

- Open Postman and click "New" > "Request".
- Choose the HTTP method (GET, POST, PUT, DELETE, etc.).

- Enter the API endpoint (URL).
- Optionally, we can add headers, authorization, request body, and parameters.
- Click "Send" to execute the request and see the response.

Q. What are collections in Postman?

Collections in Postman are used to organize and group API requests. They allow users to:

- Structure related requests in folders.
- We can save and reuse requests across different environments.
- We can share and export the collection with others for collaboration.
- We can automate requests using the Collection Runner.

Q. How do you set up a collection in Postman?

- A collection is a group of related requests.
- To create a collection, click the "New Collection" button in the left sidebar.
- We can then add individual requests to the collection and organize them using folders.

Q. Explain the use of environments in Postman.

- Environments in Postman store different sets of variables, such as URLs, API keys, and other parameters, to be used across multiple requests.
- This allows us to switch between environments (like Development, Testing, and Production) without manually editing request data.
- We can create and access environment variables using {{variableName}} notation in the requests.

Q. What is a Postman Environment Variable, and how do you use it?

Environment variables in Postman store values (like base URLs or authentication tokens) that can be reused across multiple requests.

We can set variables by:

- Creating an environment under Manage Environments.
- Defining key-value pairs.
- Using the variable in our request by wrapping it with double curly braces: {{variableName}}.
- We can also set dynamic variables in pre-request scripts or manually update them before execution.

Q. What is a Postman runner, and when would you use it?

- The Postman runner is a tool for automating the execution of collections.
- We can use it to run collections repeatedly, generate reports, and integrate with CI/CD pipelines.

Q. How can you handle authentication in Postman?

Postman supports various authentication mechanisms:

- **Basic Authentication:** Use username and password in the Authorization tab.
- **Bearer Token:** Use a static or dynamic token in the Authorization tab.
- **OAuth 1.0/2.0:** Postman can handle the OAuth process by getting and refreshing tokens automatically.
- **API Key:** Add API keys directly to the request as a query parameter or header.

Authentication can also be dynamically set in pre-request scripts using:

```
pm.request.headers.add({key: "Authorization", value: "Bearer " + token});
```

Q. Explain how to use a Bearer Token in Postman?

- First, I make sure I have the Bearer Token, either from an API login or an authentication endpoint. [Query parameter and path parameter]
- In Postman, I create a new request, set the request method (like GET or POST), and enter the API URL.
- I then go to the Authorization tab, select Bearer Token from the dropdown, and paste the token in the field provided.
- After that, when I hit Send, Postman automatically adds the token in the Authorization header as Bearer <token>.

Go to Postman

- We need to create a new collection & add a new request
- Select POST method & paste the API URL
- click on Authorization & select BEARER Token
- Paste the token copied from Git
- Go to Body, select raw and enter a body (name is a must) in JSON format
- Click on Send button

HTTP method-:

1. **POST -:** It is use to create a resource.
2. **GET -:** It is use to read or retrieve the resource.
3. **PUT -:** It is use for complete updating of the resource.

4. **PATCH -:** It is use for partial updating of the resource.
5. **DELETE-:** It is use to delete the resource.

Payload -: Pass the input details to API/body

For POST & Patch/put – we must need payload/details

For GET & Delete – we not required we can do that with query or path parameter.

Status code-:

100-: Continue
200-: Success
201-: Successfully created
204 -: Successfully deleted.
300-: Redirected
400 -: Client-side error
401 unauthorised-: bad requests.
403 Forbidden -: No permission to delete the repo.
404 not found -: The protocol and the end point are missing.
405 -: http method is wrong.
415 -: Body is missing.
422 -: Cannot create repository with the same name.
429-: Too Many Requests
500-: Server-side error

Q. How can you test for API response codes (e.g., 200, 404, 500) in Postman?

- We can use assertions to check the response status code.
- For example, we can assert that the response code is 200 to indicate a successful request.

Q. Describe how to write assertions to validate API responses.

- Assertions are used to verify that the API response meets our expectations.
- We can write assertions to check the response body, headers, status code, and more.

Q. How can you automate API tests with Postman?

- We can automate API tests in Postman by writing test scripts in JavaScript. These scripts are executed after the request is sent and the response is received.
- We can also use the Postman Collection Runner or Newman to run collections of tests automatically

Q. How do you automate API testing using Postman collections and Newman?

- To automate API testing, create a collection of related requests.
- Then, use Newman, a command-line tool, to run the collection.
- Newman can be integrated with CI/CD pipelines for continuous testing.

Q. What are Newman and the Postman Collection Runner?

- Newman is a command-line tool used to run Postman collections outside the Postman app, allowing for CI/CD integration.
- The Postman Collection Runner is a built-in feature that allows we to run collections within the Postman app, enabling batch execution of requests.

Q. How do you handle response validation in Postman?

- Response validation is handled by writing test scripts in JavaScript within the Tests tab.
- These scripts can check status codes, response times, and response bodies to ensure they meet expected criteria.

Q. What are global variables in Postman and how do you use them?

- Global variables are variables that can be accessed across all collections, environments, and requests within Postman.
- They are useful for storing values that are reused across different contexts, such as base URLs or authentication tokens.

Q. What is the difference between pre-request scripts and tests in Postman?

Pre-request scripts:

- JavaScript code that runs before the API request is sent.
- Useful for setting dynamic variables (like timestamps, tokens) or modifying request headers.

Tests:

- JavaScript code that runs after the API request is received, primarily used for validating the response (e.g., checking status codes, response times, and content).

Q. Whether an API is public or private:

1. To determine if an API is public or private, I need to check the documentation for access requirements and authentication methods.
2. If it's openly accessible and widely documented, it's public.
3. If it's restricted and requires special access permissions, it's private.

Public API

- Public APIs are usually well-documented and accessible to everyone.
- Public APIs might use simple authentication methods like API keys.
- Public APIs are often listed on the provider's website or public directories.
- Public APIs will have openly published terms of service and usage limits.

Private API

- Private APIs often require specific credentials or permissions to access their documentation.
- Private APIs generally implement stronger authentication, such as OAuth tokens, and restrict access to authorized users only.
- Private APIs aren't advertised and are typically used internally or with specific partners.
- Private APIs are usually governed by internal policies or agreements with specific partners.

Q. What are Postman Collections and Monitors, and how do they work together?

- ➢ **Collections** are groups of API requests that can be run together or individually.
- **Monitors** are used to schedule automated runs of these collections at specific intervals (e.g., hourly, daily). They allow you to:
- Continuously check the health of your APIs.
- Receive alerts when a request fails.
- Monitor can be set up from the Postman collection page with integrated test scripts for automated validation.

Q. How can you automate tests in Postman?

Postman allows us to write JavaScript code in the Tests tab to automatically validate responses.

We can add assertions to check status codes, response times, and specific data in the response. For example:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});
pm.test("Response time is less than 200ms", function () {
   pm.expect(pm.response.responseTime).to.be.below(200);
});
```

We can automate running a collection of tests using the Collection Runner or Newman CLI.

Q. What is chaining requests in Postman, and how is it useful?

- Chaining requests in Postman allows us to link multiple requests together within a collection.
- This is useful for simulating workflows or testing APIs that depend on previous responses.
- We can use variables and scripts to pass data between chained requests.

Q. Explain how to use Postman scripts (JavaScript) to customize API testing.

- Postman scripts allow us to write custom JavaScript code to extend Postman's functionality.
- We can use scripts to manipulate requests, responses, and environment variables.

Q. How do you handle dynamic parameters or variables in API requests?

- We can use environment variables, collection variables, or data-driven testing to handle dynamic parameters.
- Postman also supports dynamic variables in request URLs, headers, and bodies.

Q. Discuss the concept of API mocking and how it's implemented in Postman.

- API mocking involves creating simulated responses for APIs that are not yet available or are difficult to test in a real environment.
- Postman provides tools for creating mock servers and defining mock responses.

Q. How have you used Postman to troubleshoot API issues in a previous project?

- When I encountered a 500 error while testing a payment API, I used Postman to isolate the issue.
- By carefully examining the request and response logs, I discovered that a missing header parameter was causing the error.

Q. Describe a scenario where you used Postman to test a RESTful API.

- I used Postman to test a RESTful API for a social media platform. I created a collection of requests to fetch user profiles, create posts, and query for feed data.
- Postman's ability to handle HTTP methods and JSON payloads made it well-suited for testing RESTful APIs.

Q. How would you use Postman to test a GraphQL API?

- I used Postman to test a GraphQL API for a social media platform.
- I created a collection of requests to fetch user profiles, create posts, and query for feed data.
- Postman's ability to handle GraphQL queries with variables and fragments made it a valuable tool for testing the API's functionality.

Q. Can you explain how you would use Postman for API performance testing?

- While Postman is not primarily a performance testing tool, we can use it to gather performance metrics like response times and latency.
- We can also use plugins like Postman Interceptor to integrate with performance testing tools like JMeter.

Q. Have you ever used Postman for API documentation or sharing?

- Yes, Postman can be used to generate API documentation.
- We can export collections as OpenAPI specifications or HTML files.

- Postman also allows us to share collections with team members for collaboration.

Q. How do you test an API endpoint that requires a JSON payload?

Set the request method to POST (or PUT, PATCH).

In the Body tab, select raw and choose JSON format.

Write the JSON payload in the editor.

Example:

```
{
  "name": "John",
  "age": 30
}
```

Click Send to execute the request, and validate the response in the Tests tab using assertions.

Q. Describe how you would debug an API request that is returning an error.

- **Check the status code:** Ensure the correct endpoint and HTTP method are used.
- **Inspect headers:** Look at Authorization, Content-Type, and other required headers.
- **Examine the request payload:** Ensure correct JSON or form data is sent.
- **Use Postman Console:** View the request logs, including headers and full payload.
- **Run in Collection Runner:** Repeat the request and verify logs/output.

Q. How do you import and export collections in Postman?

- **To import a collection**: Go to the main workspace and click on Import, then upload the collection file (.json).
- **To export a collection:** Click on the three dots next to the collection and choose Export. Postman will download the collection as a .json file.

Q. Imagine you have an API that returns a list of users. How would you validate that the response contains the expected data?

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response is an array", function () {
    pm.expect(pm.response.json()).to.be.an('array');
});
pm.test("First user is John", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData[0].name).to.eql("John");
});
```

Q. How would you test rate limiting for an API?

- Use Postman Collection Runner or Newman to send multiple requests in rapid succession.
- Validate the response for rate-limiting errors (e.g., status code 429, "Too Many Requests").
- Add a delay between requests to test if the API stops responding after reaching the threshold.

To set up API testing, I follow these steps:

- Choose a tool based on the project requirements, like Postman, Rest Assured for Java.
- **Install the tool**—for example, download Postman or add Rest Assured dependencies in Maven for Java projects.
- **Define the API endpoints**, including the base URL, specific endpoints, HTTP methods (like GET, POST), and any necessary parameters such as headers or query parameters.
- **Set up authentication** if needed—using methods like OAuth, Bearer Tokens, or Basic Authentication.
- **Configure the API request** by specifying the HTTP method, adding the URL, headers (like Content-Type), and the request body for methods like POST or PUT.
- **Send the request and verify the response** by checking the status codes (e.g., 200 OK), response body, headers, and response time.

Optionally, automate API tests:

- In Postman, I create collections and write test assertions.
- In Rest Assured, I write Java test scripts with assertions, often integrated with TestNG or JUnit.

Q. What are some best practices for organizing and managing Postman collections?

- Using descriptive names for collections and folders.
- Organizing requests into logical groups.
- Using environment variables to manage test data.
- Documenting your collections and requests.
- Real-World Use Cases