1. Introduction to Postman

Postman is an API platform for building and using APIs. It simplifies each step of the API lifecycle and streamlines collaboration. It is widely used for testing APIs, automating repetitive tasks, and monitoring API performance. The interface is user-friendly and includes panels for composing requests, viewing responses, and writing test scripts. Installing Postman is straightforward for all platforms including Windows, macOS, and Linux.

2. Understanding APIs

APIs (Application Programming Interfaces) allow different software systems to communicate. REST and SOAP are common API architectures. REST APIs use HTTP methods such as GET, POST, PUT, PATCH, and DELETE to perform CRUD operations. Understanding HTTP status codes helps determine the result of an API request: 2xx indicates success, 4xx client errors, and 5xx server errors.

3. Sending Requests in Postman

Users can create a new request by selecting the HTTP method and entering the API endpoint. Postman displays the status code, time taken, size of the response, and a formatted view of the response data. It supports all HTTP methods and lets users edit requests in real time.

4. Request Components

Each request can include headers (metadata), parameters (key-value pairs in the URL), authorization settings (e.g., Bearer tokens), and request bodies (for POST/PUT). Postman supports raw input like JSON/XML and form-data for file uploads. GraphQL support is also available.

5. Collections

Collections group related API requests. They help organize and reuse requests efficiently. Users can save requests, add documentation, and structure them in folders. The Collection Runner allows batch execution of all requests in a collection.

6. Environment & Variables

Postman supports variables to simplify environment switching (dev, prod). Variable types include global, environment, collection, local, and data variables. These can be used dynamically within requests using {{variable_name}} syntax.

7. Tests & Assertions

Postman lets users write tests using JavaScript. Tests can verify status codes, response structure, or content. These tests are useful for automation and continuous integration. Example: pm.test("Status code is 200", () => { pm.response.to.have.status(200); });

8. Pre-request Scripts

Pre-request scripts are JS code snippets that execute before a request is sent. They can generate timestamps, set tokens, or prepare dynamic variables. This is helpful in creating dynamic and context-aware requests.

9. Collection Runner

The Collection Runner executes all requests in a collection. It can be used with CSV/JSON files for data-driven testing, supporting iteration and dynamic variable substitution. Results of each iteration are displayed in the runner UI.

10. Monitoring APIs

Monitoring allows users to run collections at scheduled intervals and receive alerts on failures. Useful for uptime monitoring and SLA validation. Postman provides a dashboard to visualize performance over time.

11. Mock Servers

Mock Servers return predefined responses to API requests, useful for frontend development or when the backend is unavailable. Postman lets you define mock responses and test without needing a live server.

12. Authorization Techniques

Postman supports several auth mechanisms including No Auth, API Keys, Basic Auth, Bearer Tokens, and OAuth 1.0/2.0. These can be set per request or inherited from collection/environment settings.

13. Advanced Features

Advanced Postman features include Flows (visual workflows), Newman (command-line runner), auto-generated code snippets for multiple languages, and integration with Git for version control.

14. Newman – CLI Runner

Newman is a CLI tool that runs Postman collections. Ideal for CI/CD pipelines. Syntax: newman run collection.json --environment env.json --iteration-data data.csv --reporters cli,json,html.

15. Tips & Best Practices

Organize collections with folders, avoid hardcoded values by using variables, write reusable scripts, document each request, and use monitors to ensure uptime. Always keep sensitive data secure using environment variables.