POSTMAN

What is postman?

- Postman is a powerful, standalone API (Application Programming Interface) development and testing platform used to build, test, design, modify, monitor, and document APIs.
- It provides a simple and intuitive Graphical User Interface (GUI) that allows developers and testers to send HTTP requests and view responses without writing any HTTP client code.
- Instead of manually coding network calls, Postman enables users to create test suites called **Collections** and interact directly with APIs, making API development faster and more efficient.

Postman is an API development and testing tool used to:

- Send HTTP requests (GET, POST, PUT, PATCH, DELETE)
- Test backend APIs without frontend
- Debug responses
- Automate API tests
- Manage environments and variables

👉 Used by Frontend, Backend, and QA engineers

Key Reasons to Use Postman:

- **API Testing & Debugging:** Easily send requests, check status codes, response times, and data, debugging issues with the built-in console.
- **Automation:** Automate testing with Collection Runner for end-to-end workflows and integrate tests into CI/CD pipelines Newman.
- **Collaboration:** Share collections, environments, and mock servers with teams, streamlining teamwork.
- **API Design & Mocking:** Design APIs and mock responses without a backend, helping in early development.
- **Documentation:** Generate machine-readable API documentation directly from requests.

- **Workflow Management**: Organize requests into collections, use variables for dynamic data, and manage different environments (dev, staging. prod).
- **Code Generation**: Transform requests into code snippets for various languages (JavaScript, Python).
- **Support for Various APIs**: Works with REST, SOAP, GraphQL, and other API types.

How To Create Postman Account

- Step 1 - Goto Postman website https://www.postman.com/
- Step 2 - Create Account | Check Email and verify account
- Step 3 - Can download Postman OR use Postman on web browser
- Step 4 - Login to Postman
- Step 5 - Explore GUI and features

How to add GET API Request

- Step 1 - Select the HTTP Method
- Step 2 - Add the API endpoint
- Step 3 - Add Headers, Authorizations etc as needed
- Step 4 - For POST, PUT requests add Body
- Step 5 - Save & Run, Check response.

Sample APIs For Testing

1. https://reqres.in/
2. https://httpbin.org/
3. https://fakerestapi.azurewebsites.net/
4. https://dummy.restapiexample.com/

Common HTTP Request Methods (Verbs)

HTTP defines a set of methods to indicate the desired action for a given resource:

- GET - Retrieves data from the server.
- POST - Submits data to the server to create a new resource or perform an action that changes the server's state.
- PUT - Replaces all current representations of the target resource with the data provided in the request body. It is an idempotent method.
- PATCH - Applies partial modifications to a resource.

- DELETE: Removes the specified resource from the server.
- HEAD: Asks for a response identical to a GET request, but without the response body. Useful for checking a resource's availability or metadata.
- OPTIONS: Describes the communication options available for the target resource, such as the supported HTTP methods.

How to add POST API Request

Step 1 - Create a new HTTP Request

Step 2 - Select the HTTP Method POST

Step 3 - Add the API endpoint

Step 4 - Add Headers, Authorizations etc as needed

Step 5 - For POST requests add Body

Step 6 - Save & Run, Check response.

How to add PUT and PATCH API Request

Step 1 - Create a new HTTP Request

Step 2 - Select the HTTP Method PUT for replacing the complete resource PATCH for partially updating the resource.

Step 3 - Add the API endpoint

Step 4 - Add Headers, Authorizations etc as needed

Step 5 - Add Body as needed

Step 6 - Save & Run, Check response.

How to add DELETE API Request

Step 1 - Create a new HTTP Request

Step 2 - Select the HTTP Method DELETE

Step 3 - Add the API endpoint

Step 4 - Add Headers, Authorizations etc as needed

Step 5 - Save & Run, Check response.

Collection

What is a Collection?

A Postman Collection is a grouping of related API requests, It acts as an organized container that allows you to save, reuse, request, along with all the necessary details like authorization, parameters, headers, scripts, variables, and documentation.

Collections are fundamental for:

- Organization: Grouping requests by project, feature, or workflow to keep workspace tidy.
- Collaboration: Sharing API requests and related information with team members.
- Automation & Testing: Running a sequence of requests using the Collection Runner to automate functional or performance testing.
- Documentation: Generating and publishing API documentation automatically from organized requests.

2. How to create a Collection

You can create a new collection in a few simple steps:

1. In the Postman app, go to the **Collections** tab in the left sidebar.
2. Click the + icon next to "Collections" or the **New** button in the top left and select **Collection**.
3. A new, blank collection will be created. You can name it immediately (e.g., "My API Project').
4. You can now add requests to it or configure collection-level details like variables, authorization, and pre-request scripts.

3. How to create folders inside a Collection

Folders (also known as item groups) help further categorizelection requests within a collection, especially useful for complex APIs.

- Hover over your collection's name in the left sidebar, click the three-dots (...) menu that appears next to it, and select **Add folder**.
- Give your folder a name.
- To add requests to the folder, you can either create a new request inside the folder (by right-clicking the folder and selecting **Add request**) or drag and drop existing requests into the new folder.

4. How to arrange requests inside a Collection

Postman allows you to manually reorder requests and folders to match your desired workflow or logical sequence.

- In the **Collections** sidebar, simply **drag and drop** individual requests or entire folders to change their order within the collection or move in between different folders.
- You can also sort elements alphabetically hovering over the collection/folder, clicking the (...) menu, and selecting Sort > Folders first, A to Z (or other sorting options.

5. How to bring your Collection to the top (Pinning/Favoriting)

While you cant manually sort collections in a custom order in the sidebar, you can use the favorite feature to bring important collections to the top of the list.:

- In the **Collections sidebar**, hover over the collection you want to prioritize.
- Click the **star icon** (Favorite) that appears next to the collection name.
- Favorited collections will always appear at the very top of the list in the Collections tab, above non-favorited ones. You can a favorite multiple collections,

Variables

In Postman, **variables** are reusable placehoiders that store values, allowing you to avoid hard coding data and create dynamic, efficient API testing workflows.

- In the **Collections** sidebar, hover over the collection you want to prioritize.
- Click the **star icon** (Favorite) that appears next to the collection name.
- Favorited collections-will always appear at the very top of the list in the Collections tab, above non-favorited ones. You can favorite multiple collections.

Common use cases:

- **Switching Environments**: Easily change between development, staging, and production servers by updating a `baseURL` variable in different environments.
- **Chaining Requests**: Extract data (e.g., a user ID or authentication token) from the response of one request and use it as input for a subsequent request in a sequence.

- **Switching Environments**: Easily change between development, staging, and production servers by updating a `baseURL` variable in different environments.
- **Chaining Requests**: Extract data (e.g., a user ID or authentication token) from the response of one request and use it as input for a subsequent request in a sequence.

- **Data-Driven Testing:** Use the Collection Runner to run the same request multiple times with different data sets from a CSV or JSON file.
- **Managing Sensitive Data:** Store sensitive information like API keys or passwords securely as "secret" type variables or in your Postman Vault.

How to create variables

You can create variables with different scopes, which determines their accessibility and lifespan: Global, Collection, Environment, Data, and **Local**.

Via the Postman UI

- **Environments/Globals:** Click **Environments** in the sidebar, or the environment selector dropdown in the top right, and select **Globals** or an existing environment name. Add the variable name and value in the table.
- **Collections:** In the Collections tab in the sidebar, select a collection, then ggo to the **Variables** tab. Add your variables there.
- **From a request:** In the request builder (URL,parameters, headers, or body), highlight a value, right-click, select **Set as variable**, and choose a scope.

Via Scripts (Pre-request or Test tabs)

You can set variables programmatically using the pm object in your scripts.

- Set a global variable: `pm.globals.set("variable_key", "variable_value");`
- Set an environment variable: `pm.environment.set("variable_key", "variable_value");`
- Set a collection variable: `pm.collectionVariables.set("variable_key", "variable_value");`
- Set a local variable (temporary): `pm.variables.set("variable_key", "variable_value");`

How to refer to variables in Postman

You can use variables in two primary ways. These: in the Postman UI and within scripts.

- **In the Postman UI (URLs, headers, body, etc.)**
- Enclose the variable name in double curly braces. `{{variable_name}}`. Postman will replace this placeholder with the current value when the request is sent.
- Example URL: `https://{{base_url}}/users/{{user_id}}`
- **In Scripts**
- Use specific pm object methods to get the variable's value in pre-request or test scripts.

- Get a global variable: pm.globals.get("variable_key");
- Get an environment variable: pm.environment.get("variable_key");
- Get a collection variable: pm.collectionVariables.get("variable_key");
- Get the variable from the narrowest scope: pm.variables.get("variable_key");
- Log a variable's value to the console for debugging: console.log(pm.variables.get("variable_key"))

Environment

In Postman, an environment is a set of key-value pairs (variables) that allow you to manage and switch between different configurations for your API requests, such as development, testing, and production servers.

What is an Environment

An environment in Postman is essentially a named collection of variables that you can use as placeholders in your API requests and scripts. These variables hold dynamic values like base URLs, API keys, or authentication tokens.

Why We Need Environments

Environments enhance effciency, reusability, and collaboration in API development and testing:

- Dynamic Values: You can use the same request structure (e.g, {{base_url}}/users) while the actual URL changes based on the selected environment (e.g, dev.api.com vs prod.api.com).
- Reusability: Variables allow you to reuse values across multiple requests and collections so you only need to update the value in one place if it changes.
- Collaboration: Environments can be shared with team members, ensuring everyone is using the correct configuration and managing sensitive data (like passwords or tokens).
- Scoped Access: Only one environment can be active at a time, preventing accidental use of a production API key while testing in a development environment.

How to Create an Environment

1. In the Postman application, select Environments in the left sidebar.

Using Environments in Postman

2. Click the + icon to open a new environment tab. You can also use environment selector dropdown in the top-right corner of the workbbench and select New Environment.
3. Enter a descriptive name for your new environment (e.g, "Development", "Staging", or "Production").
4. (Optional) Add initial variables as key-value pairs in the table below. You can specify a `Variable` name, Type (default or secret), and an `Initial Value`.
5. Postman automatically saves your changes.

How to Use an Environment

1. Select the **Active Environment**: Before sending a request, select the desired environment from the **environment selector** dropdown list in the top-right corneof the workbench. The selected environment becomes the active environment.
2. **Reference Variables**: In your request (URL, headers, body, or scripts), refer to the variables using double curly braces syntax: `{{variable_name}}`.
3. **Send Request**: When you send the request, Postman automatically repaices the variable name with its current value from the active environment.

How to Delete an Environment

1. Select Environments in the left sidebar to view all available environments.
2. Hover over the environment you want to delete.
3. Select the more actions icon (usually three dots) next to the environment name.
4. Select **Delete**.

Deleting an environment also deletes all the variables within it, so proceed with caution.

Environment is a set of key-value pairs

Step 1 - Create a api request

Step 2 - Create environments and add key-value pairs (variables)

Step 3 - Refer the variables in request

Step 4 - Select the environment from dropdown and run request

Step 5 - Create more environments and execute request

Request and Response in postman

In Postman, the request is the message a client sends to a server asking for an action or data, and the response is the message the server sends back with the result. Postman provides an interface to build, send, analyze, debug these interactions for API development and testing.

The Request

The request is how you, as the client (or the application you're simulating), communicate your intentions to the API server. Key components you configure in Postman include:

- **Endpoint URL:** The address of the resource you want to access (e.g., `https://api.example.com/users/`).
- **HTTP Method:** The type of action you want to perform (e.g., `GET` to retrieve data, `POST` to create new data, `PUT` or `PATCH` to update existing data, `DELETE` to remove data).
- **Headers:** Metadata providing additional information about the request, such as `Content-Type` to specify the data format or `Authorization` for access credentials.
- **Body:** The actual data payload you are sending to the server, typically used with `POST`, `PUT`, and `PATCH` requests (e.g., a JSON object with new user details).
- **Query Parameters:** Optional key-value pairs appended to the URL to filter, sort, or customize the request (e.g., `?status-active`).

The Response

The response is what the server returns after processing your request. Postman displays the response in a structured manner, providing essential details to help you verify the API's behavior. Key components of the response include:

- **Status Code:** A three-digit number indicating the outcome of the request (e.g, `200 OK` for success, `201 Created` for successful creation, `404 Not Found` if the resource doesn't exist, or `500 Internal Server Error` for a server issue).
- **Response Body:** The main content returned by the server, which can be in formats like JSON, XML, or HTML, and contains the requested data or a confirmation message.
- **Headers:** Additional metadata from the server, providing information about the response itself, such as caching instructions or the content type of the body data.
- **Response Time:** The duration it took for the server to process the request and send the response, useful for performance assessment.
- **Size:** The size of the data received in the response.

  In Postman, you can also write JavaScript test scripts that run automatically after receiving a response to validate the data, status codes, and other details.