Postman Training Manual — Beginner → Intermediate → Advanced

> This is a structured, hands-on training manual. You can read, follow steps, and practice directly in Postman while going through each section.

==========

BEGINNER LEVEL

==========

1. What is Postman? (Conceptual Foundation)

Postman is an **API development and testing platform** used to: - Send API requests - Inspect responses - Debug backend services - Automate testing - Share APIs with teams - Document APIs automatically

Why Backend Developers Use Postman

You use Postman instead of writing full programs just to test APIs because it is: - Faster than coding - Visual and interactive - Easy to debug - Reusable - Team-friendly

---

2. Understanding APIs (Before Using Postman)

An API usually works like this:

Client (Postman/Browser) → Request → Server → Response → Client

Example:

```
GET https://api.example.com/users
```

Response:

```
{
  "status": 200,
  "users": ["Alice", "Bob"]
}
```

Key things you must always notice in Postman: - URL - HTTP Method (GET, POST, PUT, DELETE) - Headers - Body (for POST/PUT) - Status Code (200, 400, 500, etc.)

---

3. Installing and Opening Postman

Screenshot Description (What you will see)

When you open Postman: - Left panel: Collections - Middle panel: Request builder - Right panel: Response viewer - Top right: Environment selector (dropdown)

---

4. Your First API Request (Step-by-Step)

Step 1 — Create a Request

1. Open Postman
2. Click **New → HTTP Request**
3. Change method to **GET**
4. Enter this URL:

```
https://jsonplaceholder.typicode.com/posts
```

5. Click **Send**

What you should see:

- Status: **200 OK**
- Body: List of JSON posts
- Time taken
- Response size

This proves: Your Postman is working correctly ✅

==========

INTERMEDIATE LEVEL

==========

5. Creating a Collection (Step-by-Step)

A **Collection = Group of related APIs**.

Steps (Imagine the screen):

1. Click **Collections** (left side)
2. Click **+ New Collection**
3. Name it:

```
User Management APIs
```

4. Click **Create**

Add your first request to collection:

1. Click **Add Request**
2. Name it: `Get All Users`
3. Enter URL:

```
https://jsonplaceholder.typicode.com/users
```

4. Click **Save to User Management APIs**

---

6. Environment Variables (Very Important Concept)

Why we need environments

Instead of:

```
https://dev.api.com
https://qa.api.com
https://prod.api.com
```

We define one variable:

```
base_url
```

Step-by-step: Create Environment

1. Click **gear icon ⚙️ (top right)**
2. Click **Add Environment**
3. Name it:

```
Dev Environment
```

4. Add variables:

Key	Value
base_url	https://jsonplaceholder.typicode.com

1. Click **Save**

Using environment in request

Change your URL to:

```
{{base_url}}/users
```

Click **Send** — it should still work.

---

7. POST Request (Creating Data)

Create new request in same collection

Name: `Create User` Method: **POST** URL:

```
{{base_url}}/users
```

Go to **Body → raw → JSON** and paste:

```
{
  "name": "John Doe",
  "email": "john@example.com"
}
```

Click **Send**

Expected Status: **201 Created**

---

8. Exporting a Collection

Steps: 1. Click three dots (...) next to your collection 2. Click **Export** 3. Choose **Collection v2.1** 4. Save the JSON file

This file can be shared with teammates.

---

9. Importing a Collection

Steps: 1. Click **Import** (top left) 2. Upload the JSON file 3. Click **Import** 4. Your collection appears again

---

==========

ADVANCED LEVEL

==========

10. Running All APIs at Once (Collection Runner)

This is like running a test suite.

Steps: 1. Click **Runner** (top left) 2. Select **User Management APIs** 3. Choose **Dev Environment** 4. Click **Run**

Postman will execute: - Get Users - Create User - Any other APIs in order

You will see: - Pass/Fail status - Time taken - Errors (if any)

---

11. Writing Automated Tests in Postman

Open any request → go to **Tests** tab → paste:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

Click **Send**. You should see: **Test Passed** 🔗

---

12. Save Token Automatically (Real-World Use Case)

Imagine login API returns a token.

In Tests tab:

```
var json = pm.response.json();
pm.environment.set("token", json.token);
```

Then in other APIs, add header:

```
Authorization: Bearer {{token}}
```

Now all APIs are connected automatically.

---

13. Newman + CI/CD (For Professionals)

You can run Postman tests from command line:

```
newman run collection.json -e dev.json
```

This is used in: - Jenkins - GitHub Actions - GitLab CI

Hands-on Practice Lab (Do This Yourself)

Lab 1 — Beginner

1. Create collection: **My First APIs**
2. Add GET request to:

   ```
   https://jsonplaceholder.typicode.com/posts
   ```

Lab 2 — Intermediate

1. Create Dev environment
2. Use {{base_url}}
3. Add POST request to create data

Lab 3 — Advanced

1. Add test script
2. Run in Collection Runner
3. Export and re-import collection

If you want, I can next convert this into: - PowerPoint slides - PDF training manual - YouTube-style course outline - Interview Q&A guide

---

COMPLETE PRACTICE PROJECT — END-TO-END (REALISTIC BACKEND WORKFLOW)

Below is a guided, hands-on project you can actually build in Postman. Treat this like a mini real-world assignment.

==========

PROJECT NAME: Book Management System APIs

==========

🕗 Objective

After finishing this project, you will be able to: - Design a collection like a real backend team - Use environments correctly - Chain APIs using tokens - Write automated tests - Run everything in Collection Runner - Export and share your work

---

STEP 1 — SET UP YOUR WORKSPACE

1. Open Postman
2. Click **New → Workspace**
3. Name it:

```
Postman Practice Project
```

4. Choose: **Personal Workspace**
5. Click **Create Workspace**

STEP 2 — CREATE ENVIRONMENT

Go to 🛑 (Manage Environments) → Click **Add**

Create environment named:

```
Practice Dev Environment
```

Add these variables:

Key	Value	Notes
base_url	https://jsonplaceholder.typicode.com	Public test API

Key	Value	Notes
token	(leave blank)	Will be filled automatically

Click **Save** and select this environment from the top-right dropdown.

STEP 3 — CREATE COLLECTION

Click **Collections → New Collection** and name it:

```
Book Management APIs
```

Click **Create**.

Inside this collection, create these folders: - Authentication - Books - Users

---

STEP 4 — CREATE API REQUESTS (REAL PRACTICE)

📁 Folder: Authentication

Request 1 — Fake Login (Simulated Auth)

Name: **Login User** Method: POST URL:

```
{{base_url}}/posts
```

Body → raw → JSON:

```
{
  "email": "student@test.com",
  "password": "password123"
}
```

Go to **Tests** tab and add:

```
pm.environment.set("token", "FAKE_JWT_TOKEN_12345");
pm.test("Login simulated", function(){
    pm.expect(pm.response.code).to.be.oneOf([200,201]);
});
```

Click **Send**.

Now your environment variable **token** is set automatically.

📁 Folder: Books

Request 2 — Get All Books

Name: **Get Books** Method: GET URL:

```
{{base_url}}/posts
```

Headers:

```
Authorization: Bearer {{token}}
```

Tests tab:

```
pm.test("Status 200", () => pm.response.to.have.status(200));
```

Request 3 — Create New Book

Name: **Create Book** Method: POST URL:

```
{{base_url}}/posts
```

Body → raw → JSON:

```
{
  "title": "Postman Mastery",
  "author": "You",
  "year": 2025
}
```

Headers:

```
Authorization: Bearer {{token}}
```

Tests tab:

```
pm.test("Book created", () => pm.response.to.have.status(201));
```

Request 4 — Update Book

Name: **Update Book** Method: PUT URL:

```
{{base_url}}/posts/1
```

Body → raw → JSON:

```
{
  "title": "Postman Advanced",
  "author": "You",
  "year": 2026
}
```

Request 5 — Delete Book

Name: **Delete Book** Method: DELETE URL:

```
{{base_url}}/posts/1
```

Tests tab:

```
pm.test("Deleted successfully", () => pm.response.code === 200 ||
pm.response.code === 204);
```

STEP 5 — RUN EVERYTHING TOGETHER

1. Click **Runner** (top left)
2. Select **Book Management APIs** collection
3. Choose **Practice Dev Environment**
4. Click **Run**

You should see all APIs execute in this order: 1. Login User 2. Get Books 3. Create Book 4. Update Book 5. Delete Book

Green = Pass, Red = Fail.

STEP 6 — EXPORT YOUR PROJECT

1. Click three dots (...) on collection
2. Click **Export** → choose **v2.1**
3. Save file as:

```
Book_Management_Postman.json
```

STEP 7 — IMPORT AND REUSE

Pretend you are a teammate: 1. Click **Import** in Postman 2. Upload your exported file 3. Select environment again 4. Run Collection Runner — everything should work.

REAL SKILLS YOU GAIN FROM THIS PROJECT

You now practiced: - Workspace setup - Environment variables - Collection structure - Auth token handling - CRUD operations (GET, POST, PUT, DELETE) - Automated tests - Collection Runner - Export/ Import workflow

---

Want a Next-Level Version?

I can upgrade this to use: - Real Spring Boot APIs, or - Node.js Express sample backend, or - Mock Server in Postman, or - Newman + GitHub Actions pipeline.