# API TESTING WITH POSTMAN

## INTRODUCTION

API testing with Postman is all about validating how different applications communicate behind the scenes, making sure every request and response flows smoothly through the system. It gives developers and testers a central hub to send API calls, inspect data, and confirm that the backend behaves exactly as expected. With its simple interface and powerful features, Postman turns complex backend testing into something efficient, structured, and surprisingly intuitive.

In modern development pipelines, Postman acts like a performance catalyst, helping teams streamline validation, catch issues early, and keep the entire digital ecosystem aligned with business goals. You can test endpoints, automate workflows, manage collections, and ensure your APIs stay reliable even as systems scale. It's the kind of tool that elevates productivity, supports rapid innovation, and keeps every project moving toward a seamless, future-ready user experience.

## API OVERVIEW AND ARCHITECTURE

### 1. INTRODUCTION TO API

An API is basically the **middle-man of the digital world** — the connector that lets two software systems talk to each other without showing their internal secrets. More formally, an **API (Application Programming Interface)** is a set of rules, protocols, and tools that define how different applications should communicate, exchange data, and perform actions. It standardizes interaction so everything stays modular, scalable, and easy to maintain.

### 2. BASE URL

The Amazon API is accessible using the following base URL:

https://mockapi.io/projects/69258a3e82b59600d7241053

## 3. Main API Endpoint

The core resource managed in this project is Products.

Endpoint: **/Products**

| Field | Type | Description |
|---|---|---|
| productid | string | Unique identifier for each product |
| productName | string | Name of the product |
| price | number | Price of the product |
| stock | boolean | Product stock status(true/false) |

## 4. CRUD Operations Overview

CRUD represents the four foundational actions that power every data-driven application: Create, Read, Update, and Delete. These operations define how information is added, accessed, modified, and removed within a system, forming the backbone of modern software workflows. Whether you're managing user profiles, product details, or task lists, CRUD operations keep data flowing smoothly and consistently across your entire architecture.

In API environments, CRUD becomes the engine that drives interaction between clients and servers. Each operation maps to specific HTTP methods — POST for Create, GET for Read, PUT/PATCH for Update, and DELETE for Delete — enabling structured and predictable communication. This framework empowers developers to build scalable, future-ready services that maintain data integrity while supporting fast iterations and constant innovation.

### A) GET — The Data Fetcher

- GET is all about retrieval. It requests data from the server without changing anything.
- Think of it like scrolling through someone's public profile — you're just viewing, not modifying.

- In system terms, it’s clean, safe, and read-only, ensuring zero impact on the database.

**B) POST — The Data Creator**

- POST is the engine of creation.
- It sends new data to the server, asking the backend to generate a fresh record — a new user, a new task, a new product.
- This method fuels innovation by letting systems grow and evolve with every new entry.

## C) PUT — The Full Update

- PUT replaces an entire resource with new data.
- It’s like refreshing a whole profile page after rewriting everything.
- In corporate terms: full-scale update, zero ambiguity — you send the complete data set, and it overwrites the old one.

## D) PATCH — The Partial Update

- PATCH is the lightweight, surgical version of PUT.
- Instead of replacing everything, it updates only the fields you specify.
- It’s efficient, modern, and perfect for agile systems where small tweaks deliver big impact.

## E) DELETE — The Data Remover

- DELETE does exactly what it says — it removes a resource permanently from the server.
- It clears clutter, maintains data hygiene, and keeps the system optimized for performance.
- One command, and the record disappears from the ecosystem.

# 5. API Architecture

## A) RESTful Architecture

API architecture is the structural blueprint that defines how different systems communicate, exchange data, and stay aligned in a scalable ecosystem. It organizes the flow of requests and responses, manages authentication, controls data formats, and ensures every interaction remains secure and predictable. Think of it as the digital highway system — a streamlined network where each route is designed with clarity, speed, and purpose. A well-designed API architecture helps teams innovate faster, integrate services effortlessly, and maintain long-term stability as applications grow.

### B) JSON Data Model

JSON (JavaScript Object Notation) is the heartbeat of most modern APIs — simple, lightweight, and incredibly readable. It structures data as key–value pairs, making it easy for both humans and machines to understand. JSON travels quickly through networks, integrates seamlessly with frontend and backend technologies, and keeps the entire system efficient. It's the universal language for exchanging data in REST APIs, powering everything from user profiles to product info to real-time analytics. Clean structure, minimal overhead, maximum compatibility — that's the JSON advantage.

Example response:


```
{
  "productName": "Ball",
  "price": "820.75",
  "stock": false,
  "productid": "1"
},
```

### C) MockAPI Backend Simulation

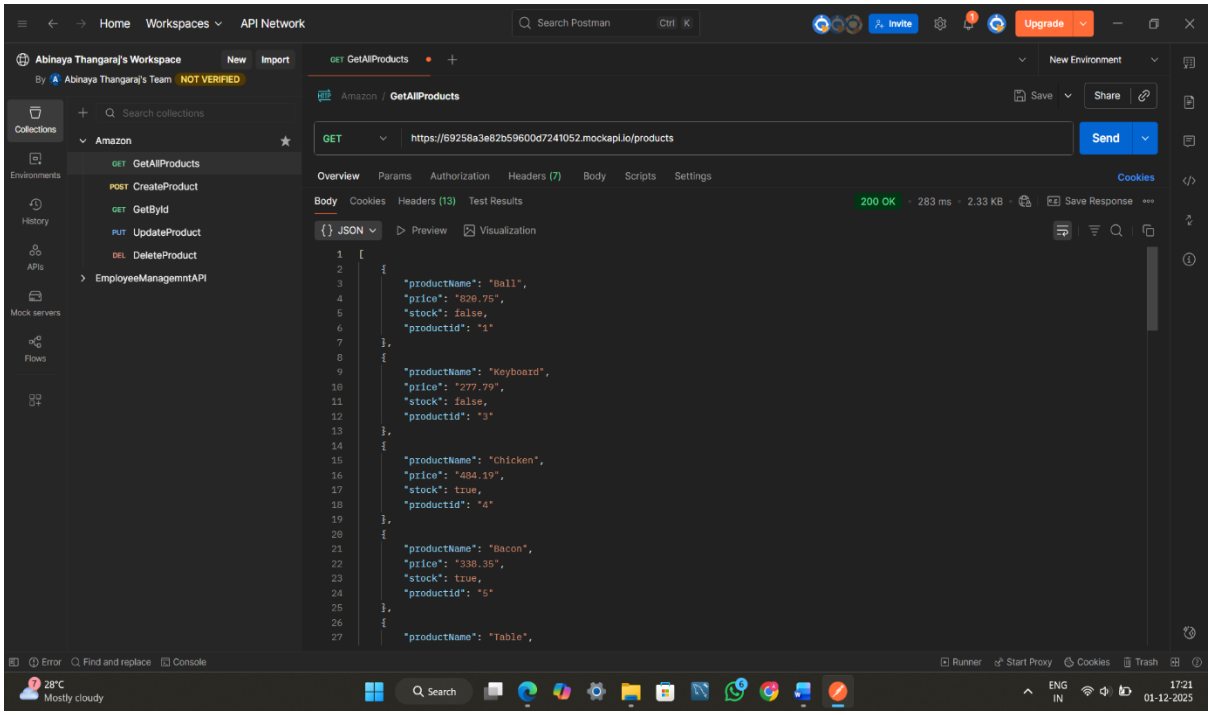
MockAPI acts like a virtual backstage for developers and testers, allowing you to simulate a backend without writing real server code. You can create endpoints, define data models, and perform full CRUD operations as if a live API already exists. This sandbox environment accelerates development, enables frontend work before the backend is ready, and gives teams a risk-free playground to validate logic, test workflows, and experiment with new features. MockAPI keeps your process agile, collaborative, and innovation-driven — a digital prototype lab built for rapid iteration.

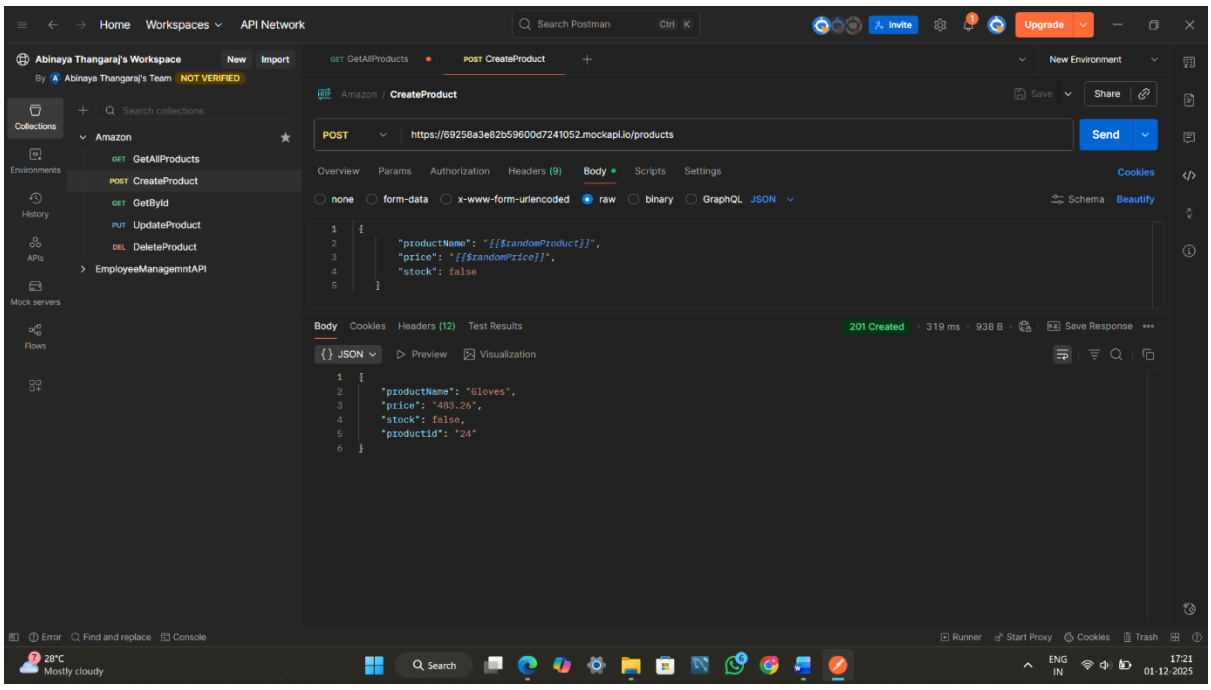
## 6. Newman and HTML Extra Reporter

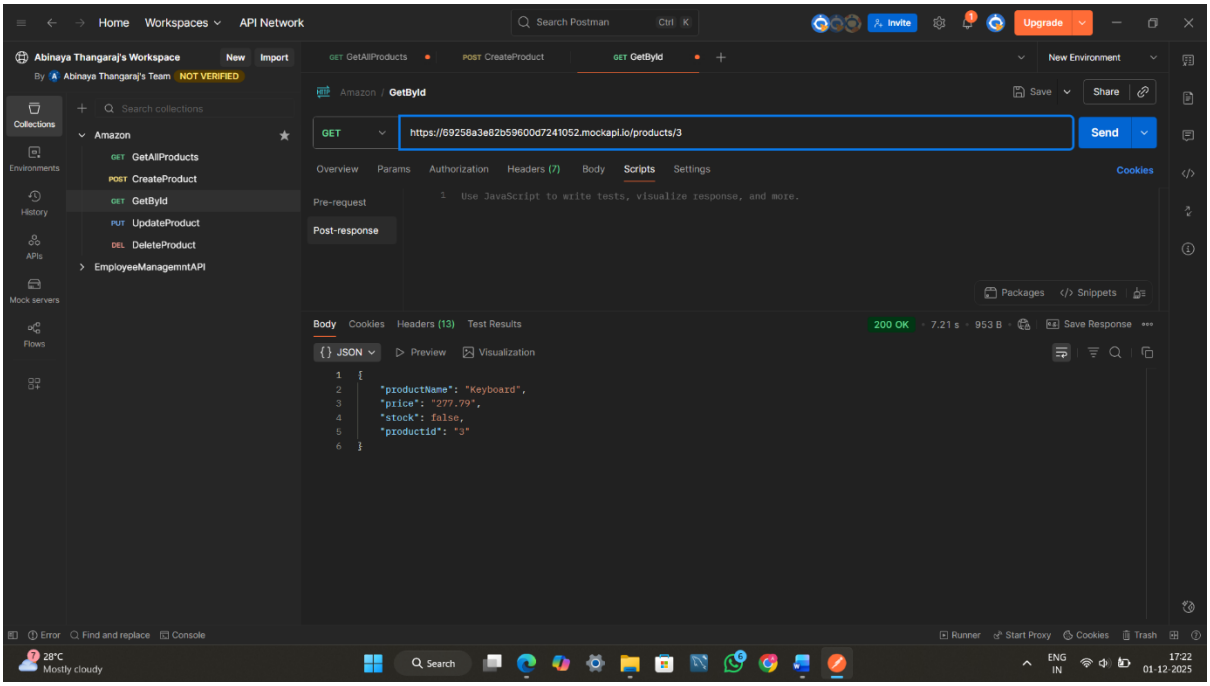
Newman is the command-line powerhouse of the Postman ecosystem — a tool built to take your API collections beyond the UI and into automated, scalable execution. It lets you run Postman tests directly from the terminal, making it perfect for CI/CD pipelines, batch runs, and enterprise workflows where consistency and speed are non-negotiable. With Newman, your collections transform into automated test suites that can validate system behaviour anytime, anywhere, keeping your delivery cycles sharp and future-ready.

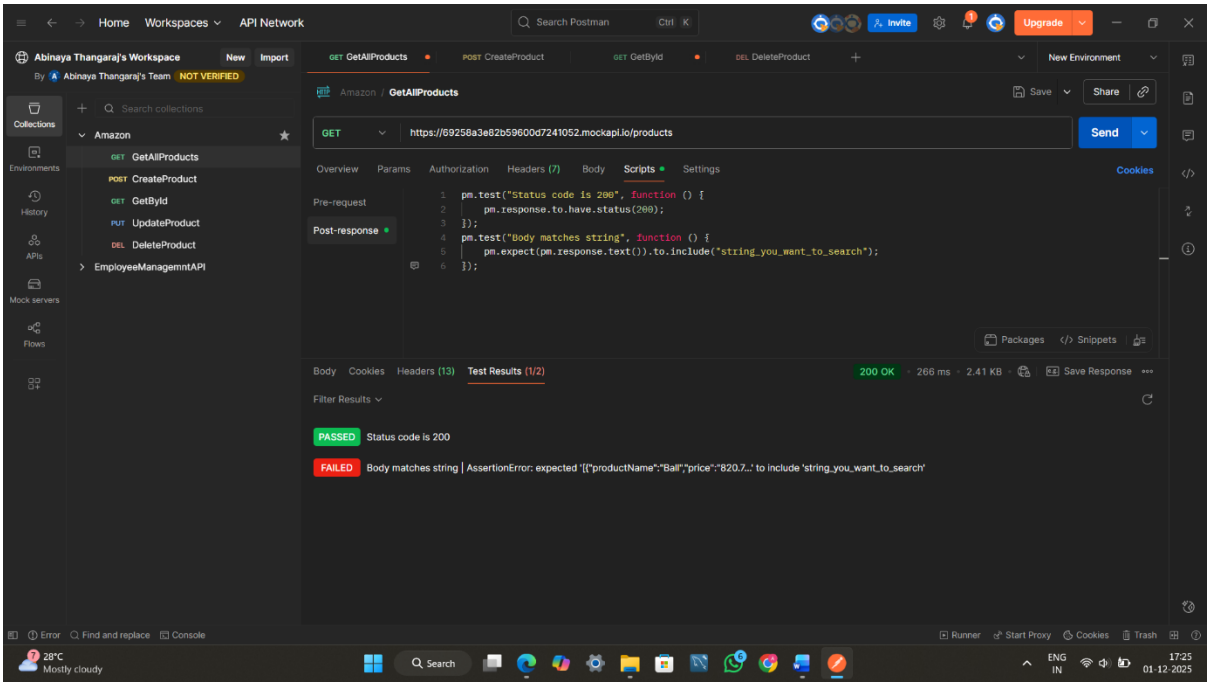
HTML Extra Reporter elevates this workflow by turning raw Newman results into beautifully structured, interactive HTML reports. It highlights passed and failed tests, response bodies, logs, and execution timelines in a clean, visually rich dashboard. This visibility helps teams troubleshoot faster, showcase test coverage, and maintain transparency across stakeholders. When paired together, Newman and HTML Extra Reporter create a synergy — automated execution powered by insightful reporting — enabling developers to deliver with confidence, clarity, and next-level efficiency.

## 7. Screenshots




Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Abinaya Thangaraj's Workspace
New
Import
Abinaya Thangaraj's Team
NOT VERIFIED
GET GetAllProducts
POST CreateProduct
GET GetById
New Environment
Amazon / GetById
Save
Share
Collections
Environments
History
APIs
Mock servers
Flows
Search collections
Amazon
GetAllProducts
CreateProduct
GetById
UpdateProduct
DeleteProduct
EmployeeManagemntAPI
GET
https://69258a3e82b59600d7241052.mockapi.io/products/3
Send
Overview
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Pre-request
Post-response
1 Use JavaScript to write tests, visualize response, and more.
Packages
Snippets
Body
Cookies
Headers (13)
Test Results
200 OK
7.21 s
953 B
Save Response
{} JSON
Preview
Visualization
{
"productName": "Keyboard",
"price": "277.79",
"stock": false,
"productid": "3"
}
Error
Find and replace
Console
Runner
Start Proxy
Cookies
Trash
28°C
Mostly cloudy
Search
ENG
IN
17:22
01-12-2025


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Abinaya Thangaraj's Workspace
New
Import
Abinaya Thangaraj's Team
NOT VERIFIED
GET GetAllProducts
POST CreateProduct
GET GetById
DEL DeleteProduct
New Environment
Amazon / GetAllProducts
Save
Share
Collections
Environments
History
APIs
Mock servers
Flows
Search collections
Amazon
GetAllProducts
CreateProduct
GetById
UpdateProduct
DeleteProduct
EmployeeManagemntAPI
GET
https://69258a3e82b59600d7241052.mockapi.io/products
Send
Overview
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Pre-request
Post-response
1 pm.test("Status code is 200", function () {
2 pm.response.to.have.status(200);
3 });
4 pm.test("Body matches string", function () {
5 pm.expect(pm.response.text()).to.include("string_you_want_to_search");
6 });
Packages
Snippets
Body
Cookies
Headers (13)
Test Results (1/2)
200 OK
266 ms
2.41 KB
Save Response
Filter Results
PASSED
Status code is 200
FAILED
Body matches string | AssertionError: expected '[{"productName":"Ball","price":"820.7...' to include 'string_you_want_to_search'
Error
Find and replace
Console
Runner
Start Proxy
Cookies
Trash
28°C
Mostly cloudy
Search
ENG
IN
17:25
01-12-2025

## 8. Conclusion

Altogether, the journey through API fundamentals, RESTful design, JSON data models, CRUD operations, MockAPI simulations, and automation tools like Newman and HTML Extra Reporter paints a clear picture of how modern digital ecosystems thrive. Every layer — from simple GET requests to full-scale automated test runs — works in harmony to deliver seamless, scalable, and resilient application experiences. These tools and architectures don't just make development easier; they create a smarter, more connected workflow where innovation moves fast and every interaction stays reliable.

In a world driven by rapid iteration and cloud-native thinking, mastering API testing equips you with the ability to build solutions that are clean, efficient, and future-ready. It empowers you to validate ideas quickly, maintain high-quality standards, and keep your systems aligned with evolving user and business needs. This is the backbone of modern software engineering — a place where precision meets creativity, and where you're always one step ahead of the curve.