# API TESTING WITH POSTMAN

## INTRODUCTION

The Inventory Management API is a RESTful service designed to efficiently handle product inventory operations such as adding, retrieving, updating, and deleting product information. This project includes a comprehensive Postman collection that not only tests API functionality but also validates performance, structure, and reliability through advanced automated test scripts.

The system interacts with a mock API endpoint to simulate real-world inventory operations. Each API request in the collection is equipped with detailed validations, including status code checks, header verification, response schema validation, and field integrity checks. Additionally, the collection uses Postman Visualizer to provide interactive visual output such as product tables, stock distribution charts, and formatted product detail cards.

## API Overview & Architecture 1. Introduction to the API

The Inventory Management API is a RESTful service designed to handle basic inventory operations such as product creation, retrieval, updating, and deletion. It enables users to manage product details efficiently through standardized HTTP methods. The API used in this project is hosted on MockAPI, a platform that simulates real-world API behavior and provides predictable, structured responses ideal for testing.

The API follows REST principles, using resource-based URLs, structured JSON responses, and stateless communication. This makes it highly scalable, testable, and compatible with various clients including web applications, command-line tools, and automated testing frameworks such as Postman and Newman.

## 2. Base URL

The Inventory Management API is accessible using the following base URL:

https://692c60fec829d464006f6827.mockapi.io/cardata

**3. Main API Endpoint** The core resource managed in this project is Products. Endpoint: **/Products**

| **Field** | **Type** | **Description** |
|---|---|---|
| **carid** | **String** | **Unique identifier for each product** |
| **name** | **String** | **Name of the product** |
| **Stocks** | **Boolean** | **Product stock status (true/false)** |

## 4.CRUD Operations Overview

### a. GET /car – Retrieve All Car Details

This endpoint fetches a complete list of products from the inventory. Used for:

- Validating product lists

- Visualizing stock distribution
- Verifying data structure

**b. POST /car – Add a New Car Data**

Allows adding a new product to the database.

```
        {
                        "Name": "benz",
                "price": 1000,
"Stocks": true
        }
```

### c. GET /car/{carid} – Retrieve car by ID

Fetches detailed information about a single product. Used to:

- Verify specific product data
- Validate schema
- Display product card in visualizer

### d. PUT /car/{carid} – Update car

Updates a product’s name, supplier, or stock status.

### e. DELETE /car/{carid} – Delete car

Removes a product from the inventory.
Also includes a follow-up request to verify deletion.

## 5. Environment Variables Used

To make the collection flexible and reusable, the following environment variables are implemented:

| **Variable** | **Description** |
|---|---|
| **baseURL** | Main API URL |
| **endpoint** | API resource name ("Products") |
| **ID** | Dynamic ID used for GET/PUT/DELETE |
| **ProductName** | Separate ID for delete testing |
| **Stocks** | Boolean stock value |

## 6.API Architecture

### a. RESTful Architecture

The API follows a REST-based architecture, which includes:

- Client–Server Model:
  The client (Postman/Newman) sends requests, and the server (MockAPI) returns JSON responses.
- Stateless Communication:
  Each API request is independent and contains all required information.
- Uniform Resource Identifiers (URIs):
  Resources are accessed using structured endpoints such as /Products/{id}.
- Standard HTTP Methods:
  - GET → Read ○
    POST → Create ○
    PUT → Update ○
    DELETE → Remove

### b. JSON Data Model

Data flows through the API in JSON format, enabling easy validation and parsing.

Example response:

```
{
  "Name": "Audi",
  "Price": 20000,
```

  "Stocks": true,

  "carid": "20"

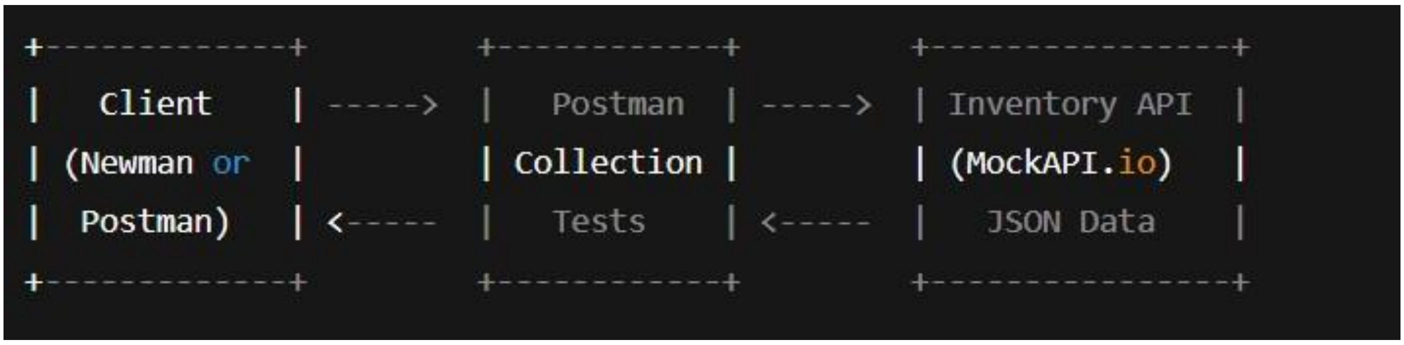
}

**c. MockAPI Backend Simulation** MockAPI

provides:

- Auto-generated IDs
- Cloud-hosted database
- Realistic request/response patterns
- Consistent structure for automated testing
  This creates an environment closely resembling a real-world API backend.

## 7. System Diagram



```
+------------+        +------------+        +---------------+
|   Client   | -----> |  Postman   | -----> | Inventory API |
| (Newman or |        | Collection |        | (MockAPI.io)  |
|  Postman)  | <----- |   Tests    | <----- |   JSON Data   |
+------------+        +------------+        +---------------+
```

## 8. Newman and HTML Extra Reporter

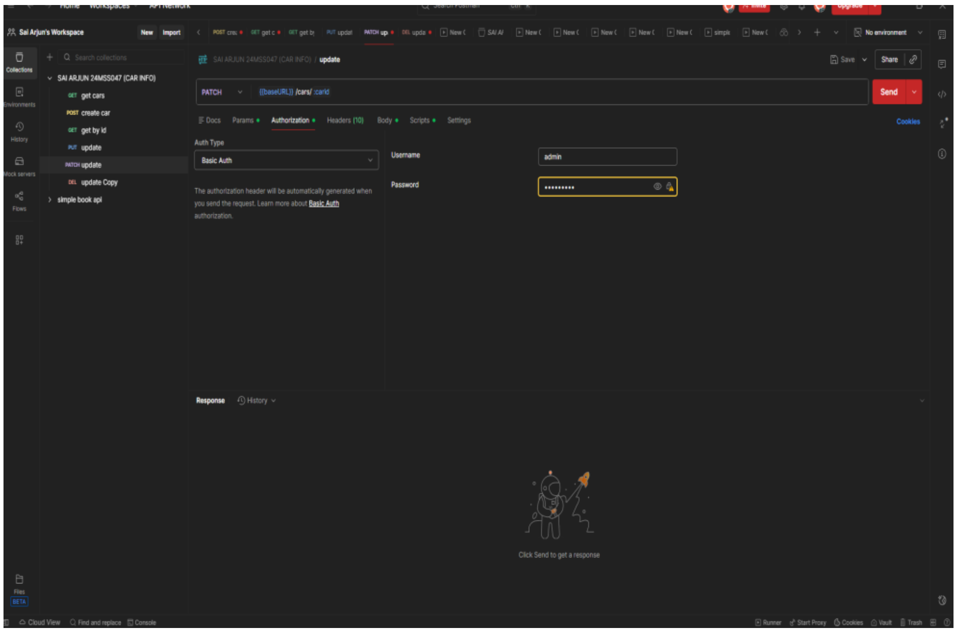
Newman is the command-line tool for running Postman collections without using the Postman application. It enables automated execution of API tests directly through the terminal, making it ideal for CI/CD pipelines and repeated testing workflows. Developers can run collections, apply environments, and export results with simple commands like:

**newman run Inventory_management.postman_collection.json newman**

**run Inventory_management.postman_collection.json -d data.csv newman run**

**Inventory_management.postman_collection.json -d data.csv -r htmlextra**

## 9.ScreenShots



SAI ARJUN 24MSS047 (CAR INFO) / update

PATCH {{baseURL}} /cars/ :carId

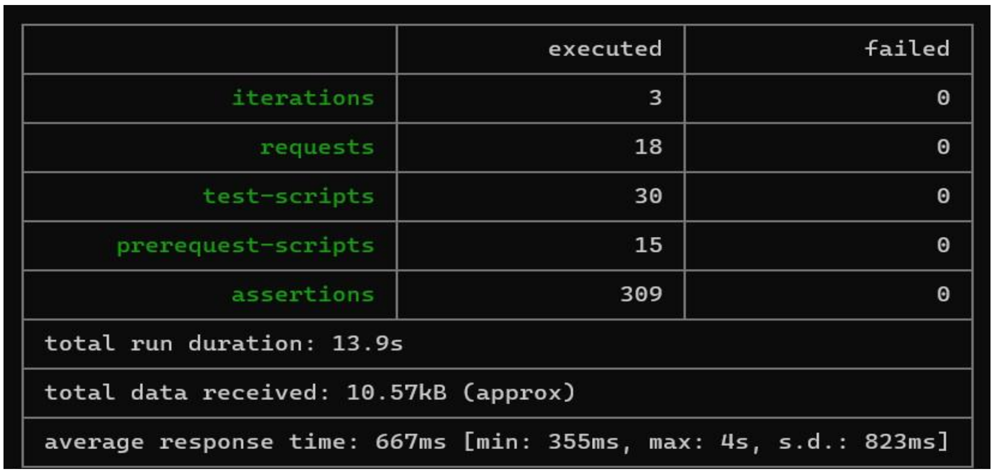
Auth Type: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about Basic Auth authorization.

Username: admin

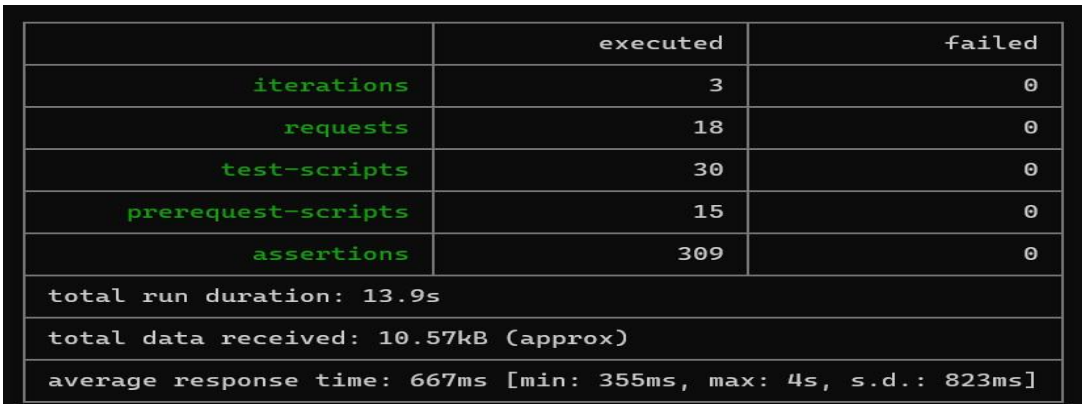
Password: ••••••••••

BASIC AUTH



```
┌─────────────────────────┬──────────────────┬──────────────────┐
│                         │         executed │           failed │
├─────────────────────────┼──────────────────┼──────────────────┤
│              iterations │                3 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│                requests │               18 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│            test-scripts │               30 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│      prerequest-scripts │               15 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│              assertions │              309 │                0 │
├─────────────────────────┴──────────────────┴──────────────────┤
│ total run duration: 13.9s                                     │
├───────────────────────────────────────────────────────────────┤
│ total data received: 10.57kB (approx)                         │
├───────────────────────────────────────────────────────────────┤
│ average response time: 667ms [min: 355ms, max: 4s, s.d.: 823ms] │
└───────────────────────────────────────────────────────────────┘
```

NEWMAN RUN USING CSV



```
┌─────────────────────────┬──────────────────┬──────────────────┐
│                         │         executed │           failed │
├─────────────────────────┼──────────────────┼──────────────────┤
│              iterations │                3 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│                requests │               18 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│            test-scripts │               30 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│      prerequest-scripts │               15 │                0 │
├─────────────────────────┼──────────────────┼──────────────────┤
│              assertions │              309 │                0 │
├─────────────────────────┴──────────────────┴──────────────────┤
│ total run duration: 13.9s                                     │
├───────────────────────────────────────────────────────────────┤
│ total data received: 10.57kB (approx)                         │
├───────────────────────────────────────────────────────────────┤
│ average response time: 667ms [min: 355ms, max: 4s, s.d.: 823ms] │
└───────────────────────────────────────────────────────────────┘
```

NEWMAN RUN USING JSON

## 10.CONCLUSION

The Inventory Management API project successfully demonstrates how RESTful services can be tested, validated, and automated using Postman and Newman. Through the use of a structured Postman collection, detailed test scripts, environment variables, and visualizers, the project provides a complete end-to-end approach to API testing. The integration of Newman and the HTML Extra reporter adds automation capabilities and generates professional-quality reports, making the solution suitable for real-world development and CI/CD pipelines. Overall, this project enhances the reliability of API workflows, improves test coverage, and showcases effective tools for modern API testing and documentation.