

# API Testing using Postman (Screenshots and summary)

**Project Name:** Practice Website

**Prepared By:** Reeti Thapa

**Tools used:** Postman

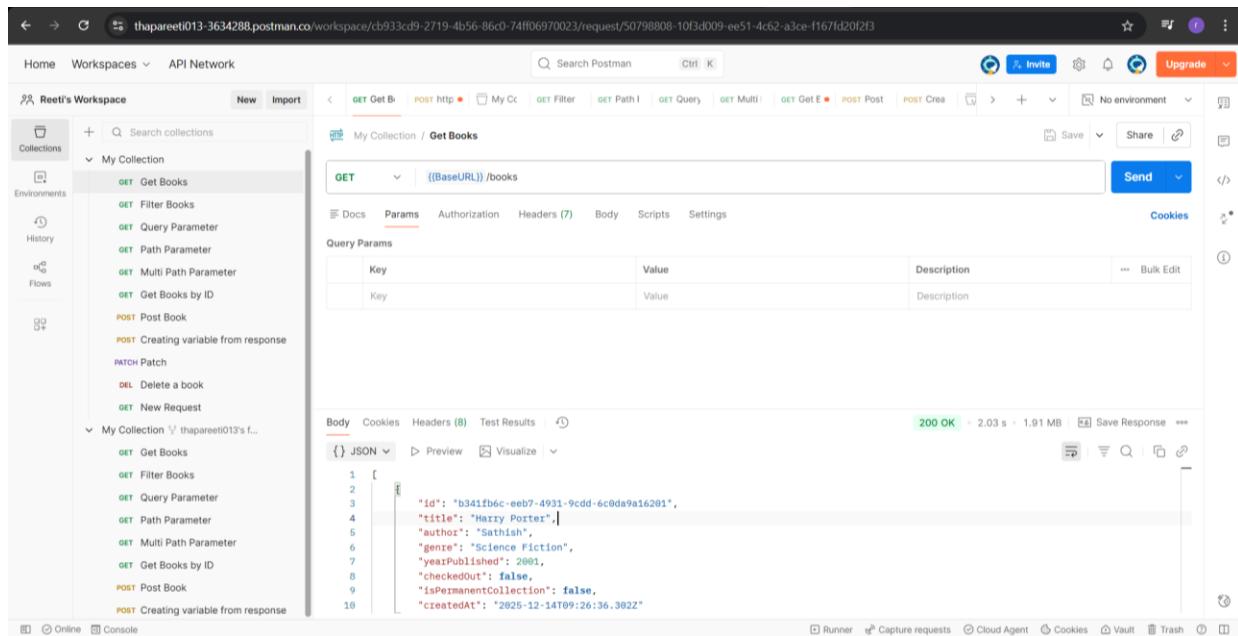
## Description:

This document contains screenshots and explanations of API testing performed using Postman.

The APIs were tested using GET, POST, PATCH, and DELETE methods, including positive and negative scenarios.

**Note:** This document contains selected API Testing using Postman screenshots as visual proof of my testing work. The full test case list with other details is available in **06\_API\_Testing-Postman.xlsx**

## 1. Test Scenario: Get all books



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Reeti's Workspace' containing collections like 'My Collection' and 'My Collection (2)'. The main area shows a 'Get Books' request under 'My Collection / Get Books'. The request method is 'GET' and the URL is '({{BaseUrl}})/books'. The 'Params' tab shows a single parameter 'Key' with 'Value'. The 'Body' tab is collapsed. The 'Test Results' section shows a 200 OK response with a JSON payload. The JSON response is as follows:

```
1 [ { 2 "id": "b341fb6c-eeb7-4931-9cdd-6c0da9a16201", 3 "title": "Harry Potter", } 4 "author": "Sathish", 5 "genre": "Science Fiction", 6 "yearPublished": 2001, 7 "checkedOut": false, 8 "isPermanentCollection": false, 9 "createdAt": "2025-12-14T09:26:36.302Z" 10 ]
```

**Execution Note:** GET Request returned 200 OK with list of books

## 2. Test Scenario: Get book by valid ID

The screenshot shows the Postman interface with a collection named "My Collection". A GET request titled "Get Books by ID" is selected. The request URL is `https://[BaseURL] /books/:id`. The "Params" tab is active, showing a parameter "id" with the value "b341fb6c-eeb7-4931-9cdd-6c0da9a16201". The "Body" tab shows a JSON response with the following content:

```
1 {
2   "id": "b341fb6c-eeb7-4931-9cdd-6c0da9a16201",
3   "title": "Harry Potter",
4   "author": "Sathish",
5   "genre": "Science Fiction",
6   "yearPublished": 2001,
7   "checkedOut": false,
8   "isPermanentCollection": false,
9   "createdAt": "2025-12-14T09:26:36.302Z"
10 }
```

The status bar at the bottom indicates a 200 OK response with 855 ms latency and 570 B size.

**Execution Note:** GET request returned 200 OK with correct book details for the given ID

## 3. Test scenario: Create book with valid data

The screenshot shows the Postman interface with a collection named "My Collection". A POST request titled "Post Book" is selected. The request URL is `https://library-api.postmanapi.com/books`. The "Body" tab is active, showing a JSON payload with the following content:

```
1 {
2   "id": "24fb01ca-4e04-4d67-93d1-1e7df1234",
3   "title": "Quality Assurance",
4   "author": "New Person",
5   "genre": "Information Technology",
6   "yearPublished": 2025
7 }
```

The status bar at the bottom indicates a 201 Created response with 3.24 s latency and 590 B size.

**Execution Note:** POST request executed successfully. API returned 201 Created with a generated book ID.

## 4. Test Scenario: Update book using PATCH

The screenshot shows the Postman interface with a collection named "My Collection". A PATCH request is selected with the URL `((BaseURL))/books/:id`. The request body contains the following JSON:

```
1 {
2     "checkedOut": true
3 }
4 }
```

The response status is 200 OK, indicating success.

**Execution Note:** Book details updated successfully. API returned 200 OK

## 5. Test Scenario: Delete book with valid ID

The screenshot shows the Postman interface with a collection named "My Collection". A DELETE request is selected with the URL `((BaseURL))/books/:id`. The request body contains the following JSON:

```
1 {
2     "id": "2689cb04-b430-464c-acfc-27aa8231095a"
3 }
```

The response status is 204 No Content, indicating success.

**Execution Note:** Book deleted successfully with 204 No Content response.

## 6. Test Scenario: Verify deleted book is not retrievable

The screenshot shows the Postman interface with a collection named "My Collection". A specific test scenario titled "Deleted book is not retrievable" is selected. The request URL is `((BaseURL)) /books/:id`. The "Params" tab is active, showing a key "id" with the value "2689cb04-b430-464c-acfc-27aa8231095a". The response status is 404 Not Found, with the message: "Book with id '2689cb04-b430-464c-acfc-27aa8231095a' not found".

**Execution Note:** GET request returned 404 Not Found, confirming successful deletion.

## 7. Test Scenario: Verify BaseURL variable works

The screenshot shows the Postman interface with the same collection and scenario. The request URL is now `((BaseURL)) /books`. The "Params" tab is active, showing a key "Key". The response status is 200 OK, indicating success. The JSON response body contains details about a book, including its ID, title ("Harry Potter"), author ("Sathish"), genre ("Science Fiction"), year published (2001), checked out status (false), permanent collection status (false), and creation date ("2025-12-14T09:26:36.392Z").

**Execution Note:** Request executed successfully using variable for BaseURL.

## **Conclusion:**

The API endpoints were tested successfully using Postman for positive and negative scenarios. Core CRUD operations (GET, POST, PATCH, DELETE) were validated, and responses were verified using appropriate status codes. This practice improved my understanding of API behavior and manual API testing.