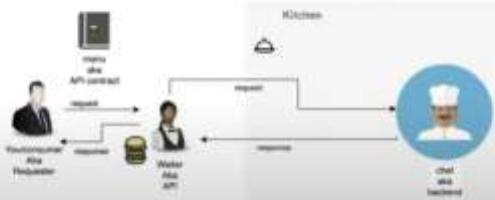


API Testing

Introduction

API - Restaurant analogy



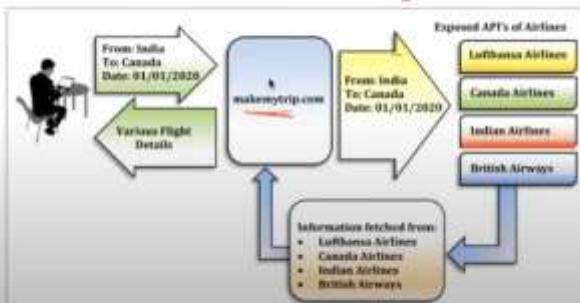
What is Client & Server?

A **client** is a computer hardware device or software that accesses a service made available by a server. The server is often (but not always) located on a separate physical computer.

A **server** is a physical computer dedicated to run services to serve the needs of other computers. Depending on the service that is running, it could be a file server, database server, home media server, print server, or web server.



Makemytrip example:



Client/Server Architecture



Types Of API

There are two types of API's,

1. Simple Object Access Protocol (SOAP)
2. REST (Representational State Transfer).

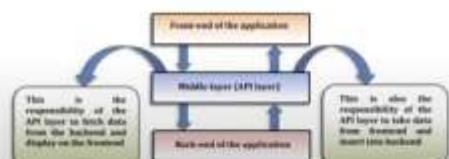
Both are the web services.

API Vs Webservice

- Web Service is an API wrapped in HTTP.
- All Web Services are API but APIs are not Web Services.
- A Web Service needs a network while an API doesn't need a network for its operation.

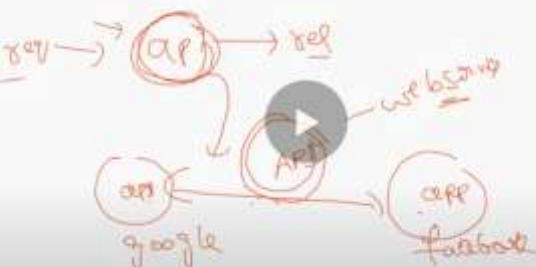
What is an API?

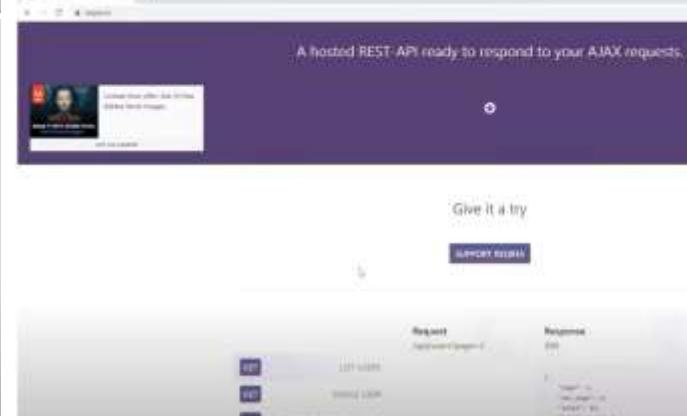
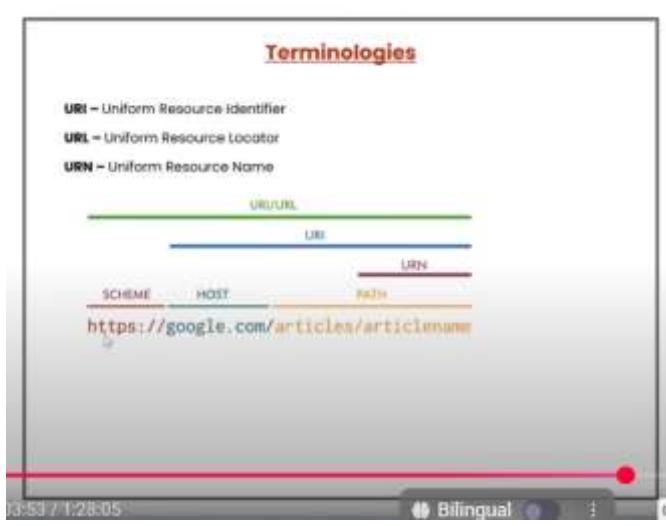
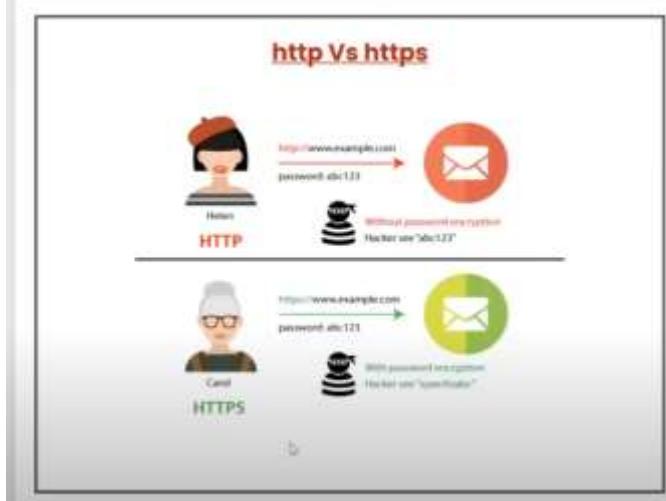
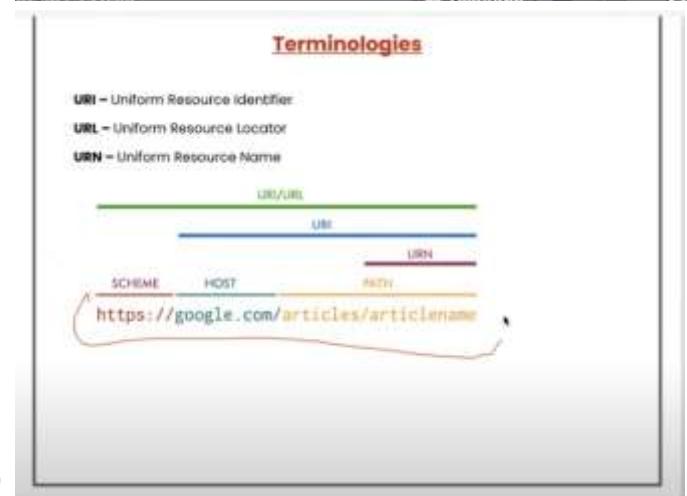
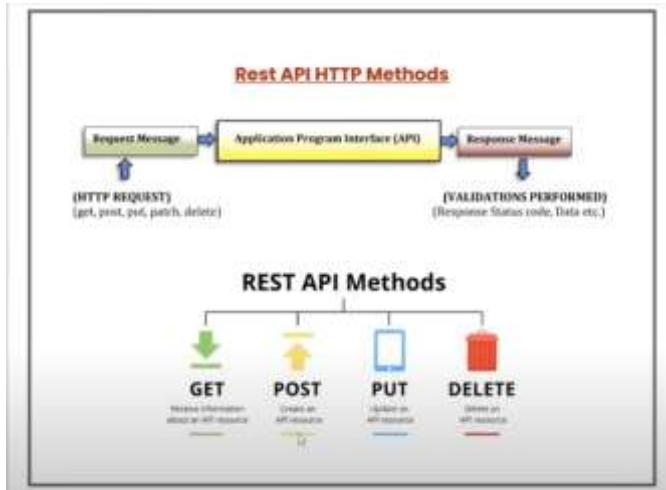
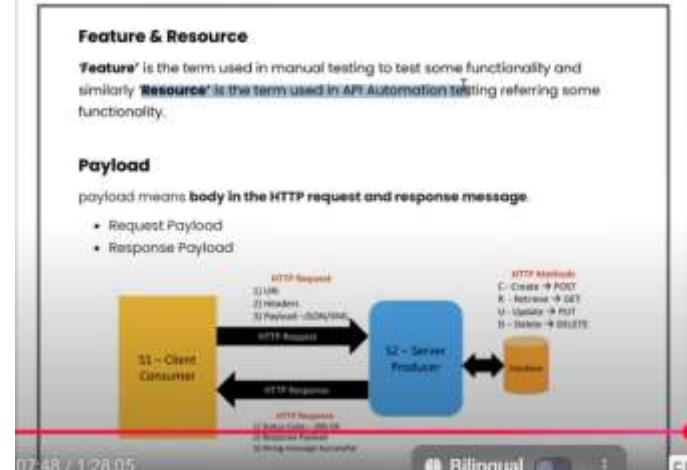
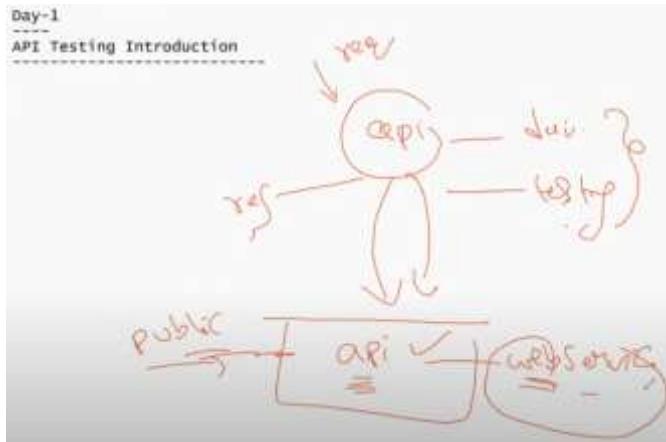
Application Program Interface (API): is the way of communication between two applications where applications may differ in their platforms or in terms of technology.



** API is a way of communication between two layers.

Day-1 API Testing Introduction





Day-1

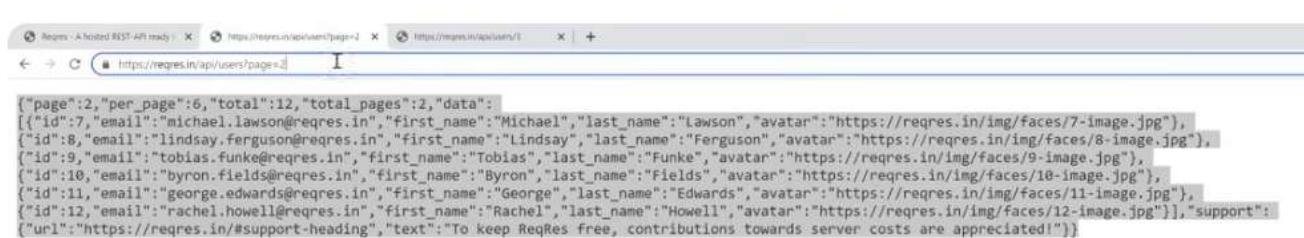
API Testing Introduction

Rest API methods / http request

get
post
put
delete

--
<http://www.google.com/>

<https://reqres.in/api/users?page=2>



The screenshot shows the ReqRes API documentation. It features two main sections: 'SUPPORT REQRES' on the left and 'REQUESTS' on the right. The 'REQUESTS' section lists several API endpoints with their descriptions:

- GET /api/users (LIST USERS)
- GET /api/users/:id (SINGLE USER)
- GET /api/users/:id (NOT FOUND)
- GET /api/users (LIST -RESOURCE)
- GET /api/users/:id (SINGLE -RESOURCE)
- GET /api/users/:id (DELETE -RESOURCE) (NO BODY)

The screenshot shows the ReqRes API documentation. It features two main sections: 'SUPPORT REQRES' on the left and 'REQUESTS' on the right. The 'REQUESTS' section lists several API endpoints with their descriptions:

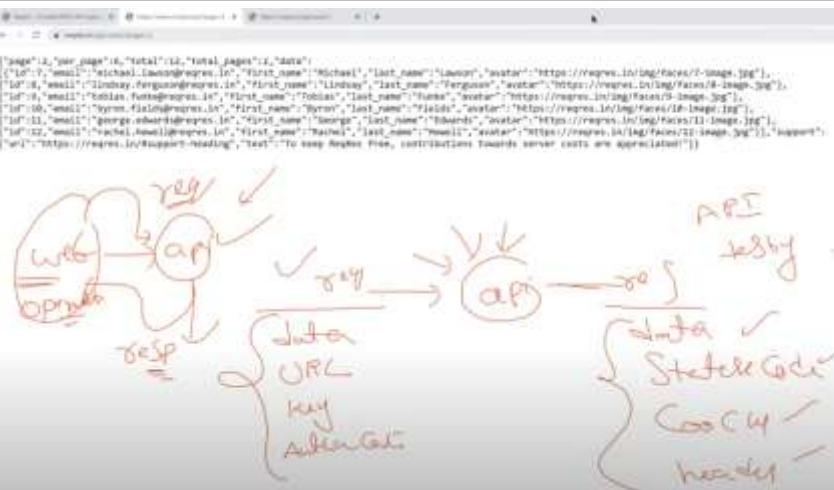
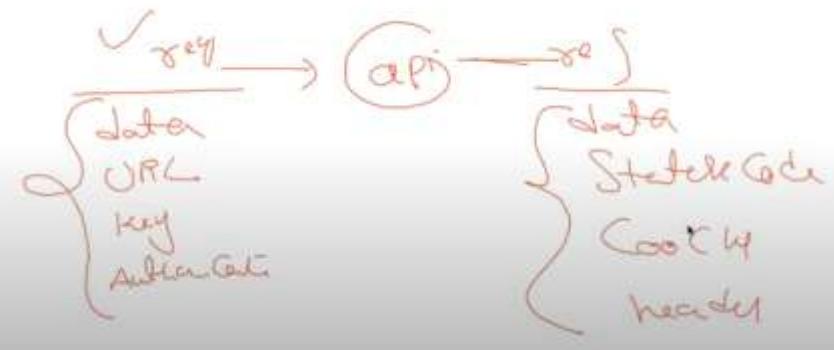
- GET /api/users (LIST USERS)
- GET /api/users/:id (SINGLE USER)
- GET /api/users/:id (NOT FOUND)
- GET /api/users (LIST -RESOURCE)

Give it a try

SUPPORT REQRES

The screenshot shows the ReqRes API documentation. It displays a single API endpoint:

Request	Response
/api/users/2	204



Feature & Resource

'Feature' is the term used in manual testing to test some functionality and similarly 'Resource' is the term used in API Automation testing referring some functionality.

Payload

payload means body in the HTTP request and response message.

- Request Payload
- Response Payload



Rest API Examples

1. Payment Gateways
2. Google Maps
3. Twitter
4. Facebook
5. Linked-In
6. GitHub

The Postman app interface. At the top, there's a navigation bar with 'Products', 'Home', 'Enterprise', 'Documentation', 'Support', and 'Logout'. Below the navigation is a section titled 'Download Postman' with a link to download the app. The main area shows a collection of API requests with their status and details.

The Postman Client Installation page. It features a red header 'Postman Client Installation'. Below it is a 'Download Postman Tool' section with a link to the download page. The page also includes a 'The Postman app' section with a brief description and a screenshot of the app's interface. There's also an orange button for 'Windows 64-bit' and a note about accepting the license agreement.

A screenshot of the Postman sign-up page. It features a logo at the top right. The main area has two sections: 'Create an account or sign in' with a 'Sign In' button, and 'A free Postman account lets you...' with several checkboxes for account features like 'Import your API documentation & test cases', 'Create private environments to isolate different code', 'Receive our weekly enhancement email', and 'Import your local API development solutions for faster development'. A video player at the bottom shows a thumbnail of a video.

A screenshot of the Postman sign-up page focusing on the 'Why sign up?' section. It lists benefits such as 'Organize all your API documentation under one message center', 'Start using Postman's built-in service delivery tools to make your day-to-day tasks easier', and 'Get help'. Below this is a 'Create Postman Account' button. A video player at the bottom shows a thumbnail of a video.

A screenshot of the Postman sign-up form. It has fields for 'Email' (placeholder 'you@example.com'), 'Password' (placeholder 'password'), and 'Confirm Password' (placeholder 'password'). Below the password fields is a 'I agree to the Postman Terms of Service' checkbox. A video player at the bottom shows a thumbnail of a video.



File Edit View Help



Create an account or sign in

Create Free Account

Sign In

Create your account or sign in later? [See what you can do](#)

A free Postman account lets you

- Organize all your API development in workspaces
- Create public workspaces to collaborate with over 10 million users
- Back up your work on Postman's cloud
- Experience the best API development platform for free!



1:12:57

Bilingual

The screenshot shows the Postman Home page. At the top, there are sections for 'Workspaces' (with a 'myworkspace' button), 'API Network' (with a 'Discover' button), and 'Explore' (with a 'Search' bar). Below these are 'Announcements' for 'Postman and GitHub Marketplace' and 'Postman and Jira Software'. The main area features a 'myworkspace' workspace card with a 'Started with Postman' section, a 'Start with something new' section, and a 'Import an existing file' section. To the right, there's a 'Explore popular APIs' section with a 'Postman API' button.

This screenshot shows the 'Create workspace' dialog box. It has fields for 'Name' (set to 'myworkspace') and 'Summary' (with placeholder text 'Add a short summary about this workspace'). Below these are 'Visibility' options: 'Personal' (selected), 'Public', 'Team', and 'Org'. At the bottom are 'Create workspace' and 'Cancel' buttons.

The screenshot shows the 'myworkspace' overview page. It includes sections for 'Overview' (with a 'myworkspace' button), 'Collections' (listing 'My first collection' and '11 other workspace collections'), 'APIs' (listing 'My first API' and '11 other workspace APIs'), 'Environments' (listing 'My first environment' and '11 other workspace environments'), 'Mock Servers' (listing 'My first mock server' and '11 other workspace mock servers'), and 'Metrics' (listing 'My first metric' and '11 other workspace metrics'). A 'Create a collection for your requests' section is also present.

This screenshot shows the 'Create workspace' dialog box again, but with a different set of options listed under 'Create workspace': 'HTTP Request' (Create a basic API request), 'Websocket Request - MQTT' (Create and test your MQTT connections), 'Collection' (Create a collection in a workspace for reuse and sharing), 'Environment' (Use custom environments in your workspace), 'Workspace' (Create a workspace to isolate your API development), 'API Documentation' (Create API documentation for your workspace), 'Mock Server' (Create a mock server for your workspace), and 'Metrics - Metrics' (Monitor workspace metrics and track performance of your API).

The screenshot shows a 'Delete Workspace?' confirmation dialog box. It states that everything in 'myworkspace' will be gone forever if deleted. An input field asks to 'Enter workspace name to confirm' with the value 'myworkspace'. At the bottom are 'Cancel' and 'Delete workspace' buttons.

Day-2

Postman - API testing tool
we can do manual testing of API's using postman.

desktop / web

workspace: area where we maintain files and saved.

workspace - creat workspace, rename , delete
creating collection

Day-2

Postman - API testing tool
we can do manual testing of API's using postman.

desktop / web

workspace: area where we maintain files and saved.

workspace - creat workspace, rename , delete

creating collection - contains number of folders and http requests.
create, rename, delete, run the collection
we can create any number of collections under workspace.

Hey there, early bird!

Recently visited workspaces

Get started with Postman

Explore popular APIs

Announcements

Postman and NYC Marathon NYC Marathon

Postman is looking up which new ingredients of the marathon NYC Marathon update ingredients

July 27, 2022 · New York City Request More

Postman New York 2022 · In Postman

Postman is looking up which new ingredients of the marathon NYC Marathon update ingredients

July 27, 2022 · New York City Request More

New Collection

Create new Collection

This collection does not use any authentication. Learn more about authentication.

Auth

Type

MyCollection

Auth

Type

This collection does not use any authentication. Learn more about authentication.

Auth

Type

MyCollection

Auth

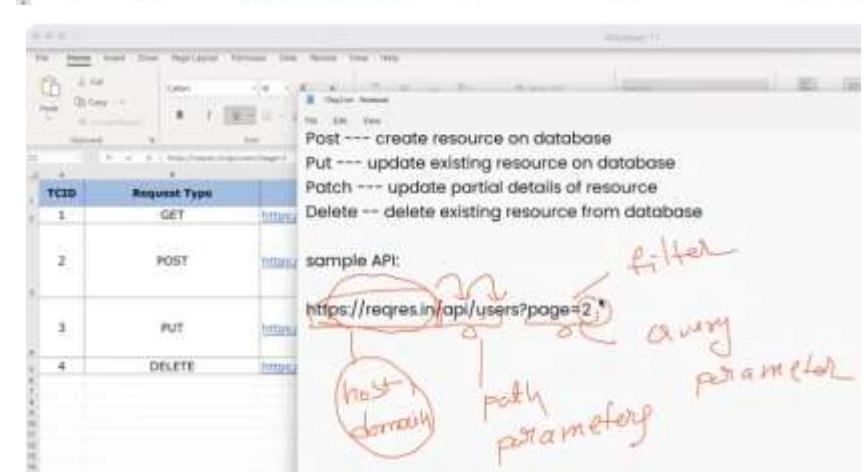
Type

This collection does not use any authentication. Learn more about authentication.

Auth

Type

TCID	Request Type	URL	Request Payload	Response Payload	Status Code
1	GET	https://reqres.in/api/users?page=2	na	{ "page": 2, "per_page": 6, "total": 12, "total_pages": 2, "data": [{ "id": 1, "name": "pavan", "job": "trainer" }, { "id": 2, "name": "pavan", "job": "engineer" }] }	200
2	POST	https://reqres.in/api/users	{ "name": "pavan", "job": "trainer" }	{ "id": 3, "name": "pavan", "job": "trainer", "createdAt": "2018-07-07T05:43:53.310Z" }	201
3	PUT	https://reqres.in/api/users/598	{ "name": "pavan", "job": "engineer" }	{ "id": 3, "name": "pavan", "job": "engineer", "updatedAt": "2022-07-16T05:09:14.041Z" }	200
4	DELETE	https://reqres.in/api/users/598	na	{}	204



	Report	Response
GET	LIST USERS	
GET	SINGLE USER	
GET	WHOLE SITE NOT FOUND	
GET	NOT FOUND	
GET	NOT FOUND	
GET	WHOLE RESOURCE NOT FOUND	
POST	CREATE	
PUT	UPDATE	
PATCH	UPDATE	
DELETE	DELETE	
DELETE	DELETE - SUCCESSFUL	

TCID	Request Type	URI	Request Payload	Response Payload	Status Code
1	GET	/users	NA	Returns list of users in a page	200
2	POST	/users	{"name": "pavan", "job": "trainer", "id": "411", "createdAt": "2022-07-16T02:00:32.846Z"}	{"name": "pavan", "job": "trainer", "id": "411", "createdAt": "2022-07-16T02:00:32.846Z"}	201
3	PUT	https://reqres.in/api/users/200	{"name": "pavan", "job": "engineer", "id": "200", "updatedAt": "2022-07-16T05:08:14.041Z"}	{"name": "pavan", "job": "engineer", "id": "200", "updatedAt": "2022-07-16T05:08:14.041Z"}	200
4	DELETE	https://reqres.in/api/users/200	na	na	204

The screenshot shows the Microsoft Graph API Explorer interface. A search bar at the top has "GET /me/messages/{id}/replies" entered. Below it, a dropdown menu shows "Me" selected. The main area displays a table with columns: Recipient, Conversation, Message ID, Body, File Resource, Status, and Message. One row is highlighted with a blue background. At the bottom, there is a large text box containing the JSON response body:

```
HTTP/1.1 200 OK
{
    "value": [
        {
            "id": "AAMkADQDAAA=...",
            "from": {
                "email": "jdoe@contoso.com"
            },
            "body": {
                "text": "Hello World"
            }
        }
    ]
}
```

The screenshot shows the NetworkMiner interface with a selected packet. The packet details pane shows a POST request to 'http://127.0.0.1:5000/api/v1/locations'. The payload pane displays the JSON data sent in the request:

```
[{"name": "paris", "lat": 48.8566, "lon": 2.3522}, {"name": "london", "lat": 51.5074, "lon": 0.1278}, {"name": "newyork", "lat": 40.7128, "lon": -74.0060}, {"name": "sydney", "lat": -33.86, "lon": 151.20}].
```

```
POST https://expressjs.com/api/users

{
  "name": "pavan",
  "job": "trainer"
}
```

```
curl -X POST https://api.sentry.io/monitors/1234567890abcdef1234567890abcdef/active?environment=staging --data-binary @- <pre>{"events": [{"id": "1234567890abcdef1234567890abcdef", "type": "error", "level": "error", "file": "app.js", "line": 123, "stack": "Error: Something went wrong\n    at handleRequest (app.js:123)\n    at handleRequest (app.js:123)\n    at handleRequest (app.js:123)", "timestamp": "2023-07-15T12:00:00Z"}]</pre>
```

The screenshot shows the SAP Fiori Launchpad interface. At the top, there's a search bar and a 'New' button. Below it, a navigation bar includes links for Home, Workstation, SAP Address, and Fiori. The main area displays a grid of application cards. One card for 'SAP ERP' is highlighted with a blue border and has a large blue 'Launch' button with a white arrow pointing right. Other visible cards include 'SAP BusinessObjects BI', 'SAP Analytics Cloud', 'SAP Data Services', 'SAP HANA Studio', 'SAP Fiori Launchpad', and 'SAP Fiori Launchpad (Mobile)'. On the left side, there's a sidebar with icons for Home, Workstation, SAP ERP, SAP Fiori Launchpad, and SAP Fiori Launchpad (Mobile). The bottom of the screen features a footer with various icons.

A screenshot of the Microsoft Teams desktop application. The main window shows a video call in progress with two participants. The participant on the left is a man with glasses and a blue shirt, while the participant on the right is a woman with blonde hair and a white top. The video feed for the woman is slightly blurred. At the bottom of the screen, there is a red button with a yellow play/pause icon. On the far left, there is a sidebar with various icons and a search bar. The overall interface is dark-themed.

A screenshot of the SAP Fiori Launchpad interface. On the left, there's a navigation bar with icons for Home, Overview, Applications, Help, and Logout. The main area shows a grid of cards representing different applications. One card for 'SAP Fiori' is visible, showing its name, a small icon, and a 'View Details' button. The background has a light grey gradient.

HTTP Status Codes

Level 200

200: OK
201: Created
202: Accepted
203: Non-Authoritative Information
204: No content

Level 400

400: Bad Request
401: Unauthorized
403: Forbidden
404: Not Found
409: Conflict

Level 500

500: Internal Server Error
501: Not Implemented
502: Bad Gateway
503: Service Unavailable
504: Gateway Timeout
509: Network Timeout

The screenshot shows the Postman application interface. On the left, there's a sidebar with a tree view of collections, currently expanded to show 'API Requests'. Below it is a table for 'TCID' (Test Case ID) with four rows. The main area displays a list of API requests. One request is highlighted in purple, showing a 'GET' method to 'https://reqres.in/api/users?page=2'. The response body is visible, showing JSON data for two users.

This screenshot is similar to the one above, showing the Postman interface with a collection named 'API Requests'. It lists several API endpoints, including 'GET /api/users?page=2' and others like 'POST /api/users' and 'PUT /api/users/1'. The interface includes a sidebar for 'TCID' and a central workspace for viewing requests and responses.

Delete -- delete existing resource from database

sample API:

<https://reqres.in/api/users?page=2>

validations

status code

time

size data

response body (json/xml)

cookies

headers

This is a summary card titled 'HTTP Status Codes' divided into three sections: 'Level 200', 'Level 400', and 'Level 500'. Each section lists specific HTTP status codes with their meanings. The 'Level 200' section includes 200, 201, 202, 203, and 204. The 'Level 400' section includes 400, 401, 403, 404, and 409. The 'Level 500' section includes 500, 501, 502, 503, 504, and 509.

This screenshot shows the Postman interface with a collection named 'API Requests'. It lists several API endpoints, including 'GET /api/users?page=2' and others like 'POST /api/users' and 'PUT /api/users/1'. The interface includes a sidebar for 'TCID' and a central workspace for viewing requests and responses.

This screenshot shows the Postman interface with a collection named 'API Requests'. It lists several API endpoints, including 'GET /api/users?page=2' and others like 'POST /api/users' and 'PUT /api/users/1'. A modal dialog is open in the bottom right corner, titled 'EXPORT COLLECTION', with instructions to skip exporting or share. It also shows options to export as a JSON file and to export the collection itself. Buttons for 'Cancel' and 'Export' are at the bottom right.

This screenshot shows the Postman interface with a collection named 'API Requests'. It lists several API endpoints, including 'GET /api/users?page=2' and others like 'POST /api/users' and 'PUT /api/users/1'. The interface includes a sidebar for 'TCID' and a central workspace for viewing requests and responses.

This screenshot shows the Postman interface with a collection named 'API Requests'. It lists several API endpoints, including 'GET /api/users?page=2' and others like 'POST /api/users' and 'PUT /api/users/1'. The interface includes a sidebar for 'TCID' and a central workspace for viewing requests and responses.

A file save dialog box is shown, prompting the user to select a file type. The 'Save as type:' dropdown menu is open, showing options like 'PDF (*.pdf)', 'Word (*.docx)', 'Excel (*.xlsx)', 'Microsoft Word (*.doc)', 'Rich Text Format (*.rtf)', and 'Microsoft Excel (*.xls)'. The 'PDF (*.pdf)' option is currently selected. The 'File name:' field contains 'API Requests Collection.pdf'. The 'Save' and 'Cancel' buttons are at the bottom right.

Postman interface showing a collection named 'mycollection'. The collection contains several API requests:

- GET** GETRequest https://replies.in/api/users?page=2 / GETRequest
- POST** POSTRequest https://replies.in/api/users / POSTRequest
- PUT** PUTRequest https://replies.in/api/users/411 / PUTRequest
- DELETE** DeleteRequest https://replies.in/api/users/411 / DeleteRequest

Import dialog box in Postman. It shows the file path: 'Select files to import' (1 file selected). The file is named 'mycollection' and is located at 'Postman-Collection.v3.1'. The 'Import All' button is highlighted.

Day-3

Step1) NodeJS

npm - node package manager

Step2) json-server

How to create dummy API's

1) Install NodeJS

Download link: <https://nodejs.org/en/download/>

Node.js download page. It shows the LTS version (Recommended for most users) and the Current version. Downloads available include:

- Windows Installer (.msi)
- Windows Binary (.zip)
- macOS Installer (.dmg)
- macOS Binary (.tar.gz)
- Linux Binaries (.deb)
- Linux Binaries (.rpm)
- Source Code

Google search results for 'download nodejs'. The top result is a link to the Node.js download page: <https://nodejs.org/download>. The snippet below the link reads: 'Download - Node.js'.

The snippet continues: 'Latest LTS Version: 10.16.0 (includes npm 6.11.0). Download the Node.js source code or a pre-built installer for your platform, and start developing today.'

Below the snippet, there are links for 'Linux on Power LE Systems: 64-bit', 'Installing Node.js via package...', 'Current Latest Features', and 'Previous Releases'.

Further down the page, there is another link: <https://nodejs.dev/download>.

<https://nodejs.org/en/download/>

The Node.js website provides several download options:

- LTS**: Recommended for most users.
- Current**: Latest version.

Downloads available:

Platform	Windows Installer (.msi)	MacOS Installer (.dmg)	Source Code
Windows	64-bit	64-bit	GitHub
Windows	32-bit	32-bit	GitHub
macOS	64-bit (.dmg)	64-bit (.dmg)	GitHub
macOS	64-bit (.tar.gz)	64-bit (.tar.gz)	GitHub
Linux	64-bit (.deb)	64-bit (.deb)	GitHub
Linux	64-bit (.rpm)	64-bit (.rpm)	GitHub
Linux	Source (.tar.gz)	Source (.tar.gz)	GitHub

Node.js Setup Wizard steps:

- Welcome to the Node.js Setup Wizard
- The Setup Wizard will install Node.js on your computer.
- End-User License Agreement: Please read the following license agreement carefully. The "Next" button is highlighted.
- End-User License Agreement: Node.js is licensed for use as follows. The "Accept the terms in the License Agreement" checkbox is checked. The "Next" button is highlighted.
- Ready to install Node.js: Click Install to begin the installation. The "Install" button is highlighted.
- Installing Node.js: Please wait while the Setup Wizard installs Node.js. Status: Copying new files.
- Completed the Node.js Setup Wizard: Click the Finish button to exit the Setup Wizard. Node.js has been successfully installed.

Completed the Node.js Setup Wizard



Click the Finish button to exit the Setup Wizard.

Node.js has been successfully installed.

2) Check node & npm versions | npm

C:\Users\pavan>node --version
v16.15.1

C:\Users\pavan>npm --version
npm WARN config global '--global',

3) install json-server

npm install -g json-server

Set Environment variable:



2) Check node & npm versions (npm comes along with node.js)

Command Prompt

C:\Users\pavan>node --version
v16.15.1

C:\Users\pavan>npm --version
npm WARN config global '--global',
8.11.0

3) install json-server

npm install -g json-server

Day-3

step1) NodeJS

npm - node package manager

node --version

npm --version

step2) json-server

Day-3

step1) NodeJS

npm - node package manager

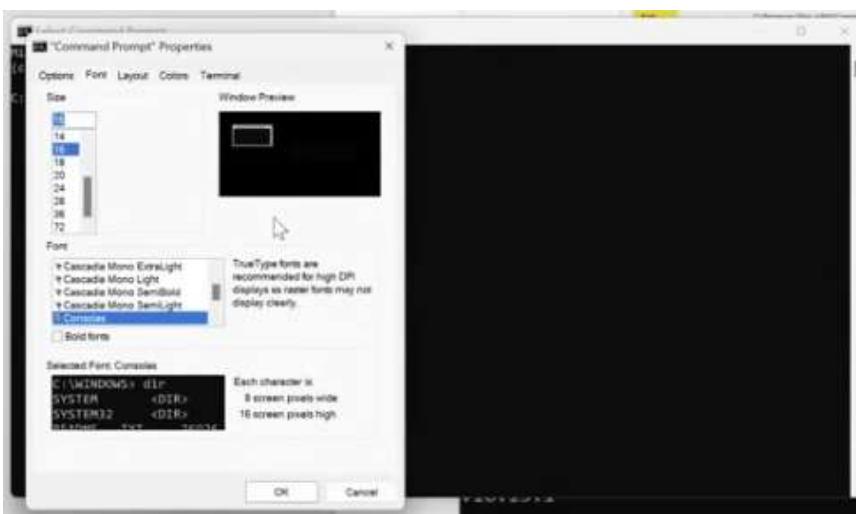
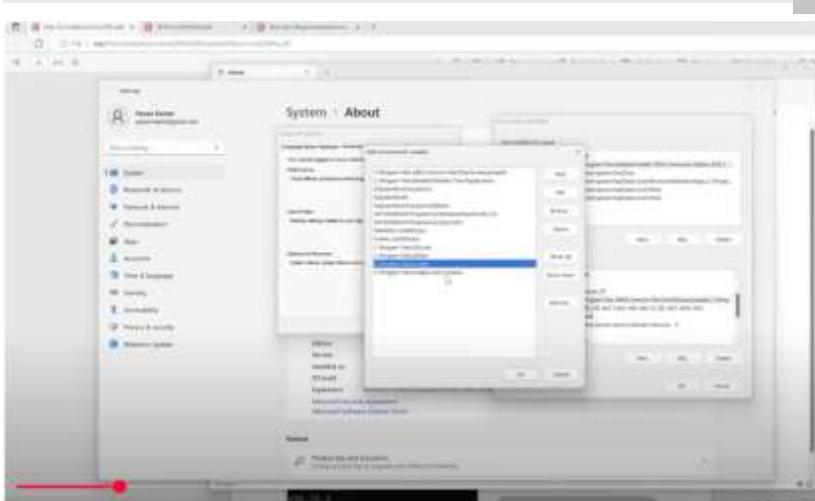
node --version

npm --version

step2) json-server

runthe below command in the cmd/terminal

npm install -g json-server



Command Prompt

```
C:\Users\pavan>node --version  
v16.15.1  
  
C:\Users\pavan>npm --version  
npm WARN config global `--global`  
8.11.0
```

3) install json-server

```
npm install -g json-server
```

Command Prompt

```
C:\Users\pavan>npm install -g json-server
```

Day-3

Creating our own API's

step1) NodeJS

```
npm - node package manager
```

```
node --version
```

```
npm --version
```

I

step2) json-server

runthe below command in the cmd/terminal

```
npm install -g json-server
```

Command Prompt

```
C:\Users\pavan>npm install -g json-server  
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global`  
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global`  
added 182 packages, and audited 183 packages in 11s  
  
90 packages are looking for funding  
  run 'npm fund' for details  
  
1 moderate severity vulnerabilities  
  
To address all issues (including breaking changes), run:  
  npm audit fix --force  
  
Run 'npm audit' for details.  
npm notice  
npm notice New minor version of npm available! 8.11.0 -> 8.13.2  
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.13.2  
npm notice Run npm install -g npm@8.13.2 to update!  
npm notice
```

4) create students.json file with the following data

```
{  
  "students": [  
    {  
      "id": 1,  
      "name": "John",  
      "location": "india",  
      "phone": "1234567890",  
      "courses": [  
        "Java",  
        "Selenium"  
      ]  
    }  
  ]  
}
```

1:23

Bilingual

```
"students": [  
  {  
    "id": 1,  
    "name": "John",  
    "location": "india",  
    "phone": "1234567890",  
    "courses": [  
      "Java",  
      "Selenium"  
    ]  
  },  
  {  
    "id": 2,  
    "name": "Kim",  
    "location": "US",  
    "phone": "98876543213",  
    "courses": [  
      "Python",  
      "Angular"  
    ]  
  }  
]
```

1:23

Bilingual

Ignore this

```
"Java",
"Selenium"
],
{
  "id": 2,
  "name": "Kim",
  "location": "US",
  "phone": "98876543213",
  "courses": [
    "Python",
    "Appium"
  ],
  "id": 3,
  "name": "Smith",
  "location": "Canada",
  "phone": "165498765",
  "courses": [
    "C#",
    "RestAPI"
  ]
},
```

```
students.json - Notepad
File Edit View
"phone": "98876543213",
"courses": [
  "Python",
  "Appium"
]
},
{
  "id": 3,
  "name": "Smith",
  "location": "Canada",
  "phone": "165498765",
  "courses": [
    "C#",
    "RestAPI"
  ]
}
```

```
1:23
students.json - Notepad
File Edit View
{
  "students": [
    {
      "id": 1,
      "name": "John",
      "location": "india",
      "phone": "1234567890",
      "courses": [
        "Java",
        "Selenium"
      ]
    },
    {
      "id": 2,
      "name": "Kim",
      "location": "US",
      "phone": "98876543213",
      "courses": [
        "Python",
        "Appium"
      ],
      "id": 3,
      "name": "Smith",
      "location": "Canada",
      "phone": "165498765",
      "courses": [
        "C#",
        "RestAPI"
      ]
    }
  ]
}
```

```
C:\Windows\system32\cmd.exe - node "C:\Users\pavan\AppData\Roaming\npm\node_modules\json-server\lib\cli\bin.js" students.json
Microsoft Windows [Version 10.0.25158.1000]
(c) Microsoft Corporation. All rights reserved.

C:\AutomationPractice\jsonfiles>json-server students.json
\{^_^\}/ hi!
Loading students.json
Done

Resources
http://localhost:3000/students

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
```

```
Microsoft Windows [Version 10.0.25158.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pavan>node --version
v16.15.1

C:\Users\pavan>npm --version
npm [yellow]config global '--global', '--local' are deprecated. Use '--location=global' instead.
8.11.0

C:\Users\pavan>
```

```
C:\Windows\system32\cmd.exe - node "C:\Users\pavan\AppData\Roaming\npm\node_modules\json-server\lib\cli\bin.js" students.json
Microsoft Windows [Version 10.0.25151.1010]
(c) Microsoft Corporation. All rights reserved.

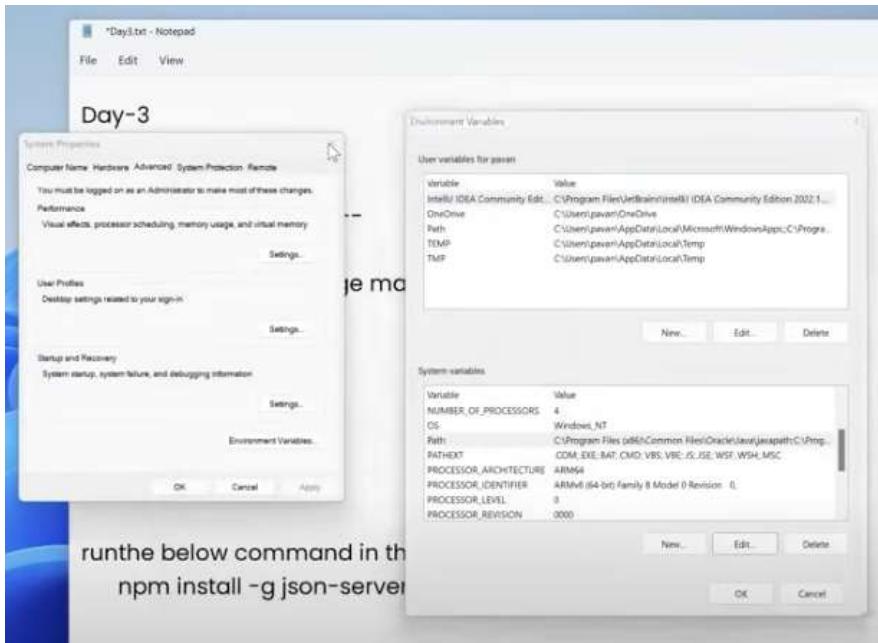
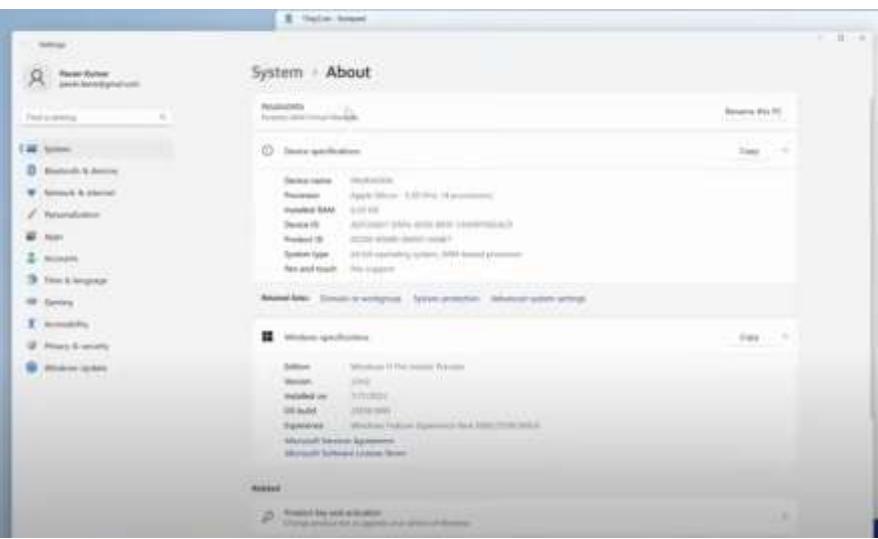
C:\AutomationPractice\jsonfiles>json-server students.json
\{^_^\}/ hi!
Loading students.json
Done

Resources
http://localhost:3000/students

Home
http://localhost:3000
```

```
{
  "id": 1,
  "name": "John",
  "location": "India",
  "phone": "1234567890",
  "courses": [
    "Java",
    "Selenium"
  ],
  {
    "id": 2,
    "name": "Kim",
    "location": "US",
    "phone": "9887654321",
    "courses": [
      "Python",
      "Appium"
    ]
  },
  {
    "id": 3,
    "name": "Smith",
    "location": "Canada",
    "phone": "165498765",
    "courses": [
      "C",
      "RestAPI"
    ]
  }
}
```

TCID	Title / Description	URL	HTTP Request	Authentication	Request Payload	Response Payload	Status Code	Validations
1	Get Single Student Data	http://localhost:3000/students/1	Get			{"id": 1, "name": "John", "location": "India", "phone": "1234567890", "courses": ["Java", "Selenium"]}	200	
2	Get all students data	http://localhost:3000/students	Get			[{"id": 1, "name": "John", "location": "India", "phone": "1234567890", "courses": ["Java", "Selenium"]}, {"id": 2, "name": "Kim", "location": "US", "phone": "9887654321", "courses": ["Python", "Appium"]}, {"id": 3, "name": "Smith", "location": "Canada", "phone": "165498765", "courses": ["C", "RestAPI"]}]	200	
3	Create new student	http://localhost:3000/students	Post		{"name": "Scott", "location": "France", "phone": "123456", "courses": ["C", "C++"]}	{"id": 1, "name": "John", "location": "India", "phone": "1234567890", "courses": ["Java", "Selenium"]}, {"id": 2, "name": "Kim", "location": "US", "phone": "9887654321", "courses": ["Python", "Appium"]}, {"id": 3, "name": "Smith", "location": "Canada", "phone": "165498765", "courses": ["C", "RestAPI"]}, {"id": 4, "name": "Scott", "location": "France", "phone": "123456", "courses": ["C", "C++"]}	201	



runthe below command in the terminal
npm install -g json-server

A	B	C	D	E	F	G	H	I
3	Create new student	http://localhost:3000/students	Post	NA	{"name": "Scott", "location": "France", "phone": "123456", "courses": ["C", "C++"]}	{"name": "Scott", "location": "France", "phone": "123456", "courses": ["C", "C++"]}	201	
4	Update Student	http://localhost:3000/students/4	Put	NA	{"name": "Scott", "location": "Germany", "phone": "654321", "courses": ["C#", "C++"]}	{"name": "Scott", "location": "Germany", "phone": "654321", "courses": ["C#", "C++"]}	200	
5	Delete Student	http://localhost:3000/students/4	Delete	NA	{}()	{}()	200	

The screenshot shows the Microsoft Azure API Management interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', and 'Explore'. A search bar is at the top right. Below the navigation, there's a sidebar with 'Workspaces' selected, showing a tree view of 'My Requests' and 'My Services'. Under 'My Requests', 'StudentAPI' is expanded, showing 'Get Single Student', 'Get All Students Data', 'Create New Student', 'Update Student', and 'Delete Student'. The 'Update Student' item is highlighted with a blue selection bar.

The main area is titled 'StudentAPI - Update Student'. It shows a 'POST' method pointing to 'https://studentapi.000000000000.azurewebsites.net/api/students/{id}'. The 'Body' tab is selected, showing a JSON payload:

```
1. { "name": "Scott", "2. "location": "Seattle", "3. "phone": "5554212", "4. "languages": [ "C#", "5. C", "6. Java", "7. Python", "8. Go" ] }
```

Below the body, there are tabs for 'Headers', 'Authentication', 'Headers (V2)', and 'Preview'. At the bottom, there are buttons for 'Run', 'Stop', and 'Delete'.

The screenshot shows the Postman application interface. The left sidebar displays a collection named 'mycollection' with several requests: 'Get Single Student', 'Get All Students', 'Create New Student', 'Update Student', and 'Delete Student'. The main workspace shows a 'Get Single Student' request with the following details:

- Method:** GET
- URL:** `http://localhost:5000/api/students/:id`
- Headers:** Content-Type: application/json
- Query Parameters:** None
- Body:** Raw (JSON) - A JSON object representing a student:

```
[{"id": 1, "name": "John", "age": 20, "city": "India", "phone": "1234567890", "courses": ["Java", "Selenium"]}]
```

The screenshot shows the Postman application interface. The left sidebar has 'APIs' selected under 'Workspaces'. The main area shows a collection named 'API Examples'. A specific API endpoint, 'Get API Students Data', is selected. The 'Method' dropdown shows 'GET'. The 'Request' tab is active, displaying the following JSON body:

```
[{"id": 1, "name": "India", "phone": "1234567890", "language": ["Java", "C", "C++"]}, {"id": 2, "name": "Raja", "phone": "9876543210", "language": ["Python", "Apex"]}]
```

The screenshot shows a Microsoft Project interface with the following details:

- Project Name:** 1st Single Review
- Task List:** The task list includes:
 - Start Date: 2024-01-01
 - End Date: 2024-01-02
 - Duration: 1 day
 - Progress: 0%
 - Remaining Work: 0 hours
 - Cost: \$0.00
- Task Details:** A table lists tasks with their status, duration, and start/end dates.

Task	Status	Duration	Start Date	End Date
1st Single Review	In Progress	1 day	2024-01-01	2024-01-02
Review All Meeting Notes	Not Started	1 day	2024-01-01	2024-01-02
Update New Ideas	Not Started	1 day	2024-01-01	2024-01-02
Finalize Slides	Not Started	1 day	2024-01-01	2024-01-02
Review Previous Feedback	Not Started	1 day	2024-01-01	2024-01-02
- Filter Options:** On the right, there are several filter buttons:
 - Normal
 - Summary
 - Project Status
 - Not Started
 - In Progress
 - Completed
 - Remaining Work
 - Cost
 - Due Date
 - Start Date
 - Assigned To

step 1 json-server

run the below command in the cmd/namerd

```
npm install -g json-server
```

tests/validations

JSON - A JS Script Object Notation

Data Types

- **Array**
 - Values in JSON can be arrays.
 - Example:

```
[{"employees": ["John", "Anna", "Peter"]}]
```
- **Boolean**
 - Values in JSON can be true/false.
 - Example:

```
{"sale":true}
```
- **Null**
 - Values in JSON can be null.
 - Example:

```
{"middleName":null}
```

JSON - Syntax

- Data should be in name/value pairs
- Data should be separated by commas
- Curly braces should hold objects
- Square brackets hold arrays

```
{ "student": [ { "id": "01", "name": "Tom", "lastname": "Price" }, { "id": "02", "name": "Mike", "lastname": "Thompson" } ] }
```

JSON vs XML

JSON	XML
JSON is simple to read and write.	XML is less simpler as compared to JSON.
It also supports array.	It doesn't support array.
JSON files are more human-readable than XML.	XML files are less human readable.
It supports only text and number data type.	It supports many data types such as text, number, Images, charts, graphs, etc.

Examples

JSON Example

```
[{"Employee": [ {"name": "John", "email": "jewwa1987@gmail.com"}, {"name": "Rahul", "email": "rahul2@gmail.com"}, {"name": "Sai", "email": "sai2@gmail.com"} ] }
```

XML Example

```
<Employees>
  <Employee>
    <name>John</name>
    <email>jewwa1987@gmail.com</email>
  </Employee>
  <Employee>
    <name>Rahul</name>
    <email>rahul2@gmail.com</email>
  </Employee>
  <Employee>
    <name>Sai</name>
    <email>sai2@gmail.com</email>
  </Employee>
</Employees>
```

JSON Object

- JSON object holds key/value pair. Each key is represented as a string in JSON and value can be of any type.
- The keys and values are separated by colon. Each key/value pair is separated by comma.
- The curly brace { represents JSON object.
- Example:

```
{ "Employee": { "name": "John", "salary": 50000, "married": true }}
```



```
{ "students": [100,200,300] }
```

```
{ "students": ["A","B","C"] }
```

What is JSON?

- **JSON - Java Script Object Notation**
- JSON is a syntax for storing and exchanging data.
- Basically it was designed for human-readable data interchange.
- JSON is text, written with Java Script Object Notation.
- It has been extended from the JavaScript scripting language.
- The filename extension is **.json**.
- JSON Internet Media type is **application/json**.

www.pythondoctraining.com

JSON Data Types

Number

java> json -www

numbers command in the cmd terminal
numbers@192.168.1.11:~

java> json

JSON - Java Script Object Notation

any@192.168.1.11:

java> json

JSON Object with String

```
{"name": "Scott", "email": "scott@solita1987@gmail.com"}
```

JSON Object with Numbers

```
{"integer": 34, "fraction": 2145, "exponent": 6.61788e+0}
```

JSON Object with Boolean

```
{"first": true, "second": false}
```

JSON Nested Object

```
{"firstname": "John", "lastname": "Doe", "age": 27, "address": { "street": "123 Main Street", "city": "Anytown", "state": "CA", "zip": "90210" }}
```

JSON Data Types

String

Strings in JSON must be written in double quotes.

Example:

```
{ "name": "John" }
```

Numbers

Numbers in JSON must be an integer or a floating point.

Example:

```
{ "age": 30 }
```

Object

Values in JSON can be objects.

Example:

```
{ "Employee": { "name": "John", "age": 30, "city": "New York" } }
```

JSON Array

- JSON array represents ordered list of values.
- JSON array can store multiple values. It can store string, number, boolean or object in JSON array.
- In JSON array, values must be separated by comma.
- The [(square bracket) represents JSON array.

JSON Array of Strings

[“Sunday”, “Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday”]

JSON Array of Numbers

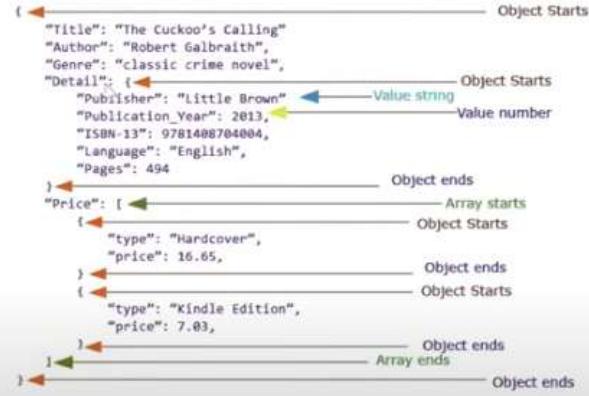
[12, 34, 56, 43, 95]

JSON Array of Booleans

[true, true, false, false, true]

JSON Array of Objects

```
{'employees': [
    {'name': 'Ram', 'email': 'ram@gmail.com', 'age': 23},
    {'name': 'Shyam', 'email': 'shyam23@gmail.com', 'age': 28},
    {'name': 'John', 'email': 'john@gmail.com', 'age': 33},
    {'name': 'Bob', 'email': 'bob32@gmail.com', 'age': 41}
]}
```



Capture & Validate JSON Path

<https://jsonpathfinder.com/>

<https://jsonpath.com/>

```
students[0]
  {
    'sid': 103,
    'sname': 'Sagar',
    'grad': 'C'
  }
  {
    'sid': 101,
    'sname': 'John',
    'grad': 'A'
  }
  {
    'sid': 102,
    'sname': 'Kim',
    'grad': 'B'
  }
}
```

```
students[0]
  {
    'sid': 101,
    'sname': 'John',
    'grad': 'A'
  }
  {
    'sid': 102,
    'sname': 'Kim',
    'grad': 'B'
  }
  {
    'sid': 103,
    'sname': 'Sagar',
    'grad': 'C'
  }
}
```

students[0].name -----> John

The screenshot shows the JSON Path Finder interface with the following details:

- Path:** `/students[0].name`
- Evaluation Results:**
 - Path: `/students[0].name`
 - Value: `John`
 - Type: `String`
 - Location: `101`
 - Phone: `1234567890`
 - Address: `None`

The screenshot shows the JSON Path Finder interface with the following details:

- Path:** `/students[0].name`
- Evaluation Results:**
 - Path: `/students[0].name`
 - Value: `John`
 - Type: `String`
 - Location: `101`
 - Phone: `1234567890`
 - Address: `None`

npm --version

step2) json-server

runthe below command in the cmd/terminal
npm install -g json-server

tests/validations

JSON - Java Script Object Notation

key values pairs

key: value

key is always included in "" quotation

```
{
  "firstname": "John",
  "secondname": null,
  "age": 30,
  "phone": [1234567890],
  "status": true
}
```

students data

```
students[0]
  {
    'sid': 103,
    'sname': 'Sagar',
    'grad': 'C'
  }
  {
    'sid': 101,
    'sname': 'John',
    'grad': 'A'
  }
  {
    'sid': 102,
    'sname': 'Kim',
    'grad': 'B'
  }
}
```

The screenshot shows the JSON Path Finder interface with the following details:

- Path:** `/students[0].name`
- Evaluation Results:**
 - Path: `/students[0].name`
 - Value: `John`
 - Type: `String`
 - Location: `101`
 - Phone: `1234567890`
 - Address: `None`

Day-4	
Response validations	
Status code	Status code
Headers	Headers
Cookies	Cookies
Response time	Response time
Response body	Response body
Assertion - validation	
pm - library functions	pm - library functions
javascript	javascript

Response Validations

(Adding Tests)

- Status code
- Headers
- Cookies
- Response time
- Response body

```
pm.test("Test Name", function()
{
  // assertion
})
```

```
pm.test("Test Name", () =>
{
  // assertion
})
```

Testing status codes

Test for the response status code:

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

If you want to test for the status code being one of a set, include them all in an array and use `oneOf`:

```
pm.test("Successful POST request", () => {
  pm.expect(pm.response.code).to.be.oneOf([201,202]);
});
```

Check the status code text:

```
pm.test("Status code name has string", () => {
  pm.response.to.have.status("Created");
});
```

functions
javascript

Normal function
Arrow function

```
pm.test("Test Name", () =>
  {
    // assertion;
  }
);
```

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

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

pm.expect(pm.response.code).to.be.oneOf([201,202]);

pm.expect(pm.response.text).to.contain('{"id":1,"name":"John","last_name":"Doe","phone":"1234567890","success":true,"validation":"Valid"}');
```

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

Testing headers

Check that a response header is present:

```
pm.test("Content-Type header is present", () => {
  pm.response.to.have.header("Content-Type");
});
```

Test for a response header having a particular value:

```
pm.test("Content-Type header is application/json", () => {
  pm.expect(pm.response.headers.get('Content-Type')).to.eql('application/json; charset=utf-8');
});
```

```
Microsoft Windows [Version 10.0.25150.1000]
c) Microsoft Corporation. All rights reserved.

:AutomationPractice\jsonfiles>json-server --watch students.json

\(_*)/nL

Loading students.json
Done

Resources
http://localhost:3000/students

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
```

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

pm.expect(pm.response.code).to.be.oneOf([201,202]);

pm.expect(pm.response.text).to.contain('{"id":1,"name":"John","last_name":"Doe","phone":"1234567890","success":true,"validation":"Valid"}');
```

```

GET - https://reqres.in/api/users/2
pm.test("Status code is 200", () => {
  pm.response.to.have.status(200);
});

```

Testing status codes

Test for the response status code:

```

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

```

If you want to test for the status code being one of a set, include them all in an array and use `oneOf`

```

pm.test("Successful POST request", () => {
  pm.expect(pm.response.code).to.be.oneOf([201, 202]);
});

```

Check the status code text:

```

pm.test("Status code name has string", () => {
  pm.response.to.have.status("Created");
});

```

```

pm.test("Cookie 'language' is present", () => {
  pm.expect(pm.cookies.has('language')).to.be.true;
});

```

Testing cookies

Test if a cookie is present in the response:

```

pm.test("Cookie 'language' is present", () => {
  pm.expect(pm.cookies.has('language')).to.be.true;
});

```

Test for a particular cookie value:

```

pm.test("Cookie language has value 1", () => {
  pm.expect(pm.cookies.get('language')).to.eql('en-gb');
});

```

Testing response times

Test for the response time to be within a specified range:

```

pm.test("Response time is less than 200ms", () => {
  pm.expect(pm.response.responseTime).to.be.below(200);
});

```

Response Validations

(Adding Tests)

- Status code
- Headers
- Cookies
- Response time
- Response body

Chai Assertion Library

```

pm.test("Test Name", function() {
  // assertion
});

pm.test("Test Name", () =>
  // assertion
);

```

```

pm.expect(pm.response.json()).to.be.an("object");
pm.expect(pm.response.json().name).to.be.a("string");
pm.expect(pm.response.json().id).to.be.a("number");
pm.expect(pm.response.json().courses).to.be.an("array");

```

Asserting a value type

Test the type of any part of the response:

```

{
  "id": 1,
  "name": "John",
  "location": "india",
  "phone": "1234567890",
  "courses": [
    "Java",
    "Selenium"
  ]
}

```

```

const jsonData = pm.response.json();
pm.test("Test data type of the response", () => {
  pm.expect(jsonData).to.be.an("object");
  pm.expect(jsonData.name).to.be.a("string");
  pm.expect(jsonData.id).to.be.a("number");
  pm.expect(jsonData.courses).to.be.an("array");
});

```

```

11: 111
12: expect(pm.response.json().name).to.be.a("string");
13: expect(pm.response.json().name).to.be.equal("John");
14: expect(pm.response.json().id).to.be.a("number");
15: expect(pm.response.json().id).to.be.equal(1);
16: expect(pm.response.json().location).to.be.a("string");
17: expect(pm.response.json().location).to.be.equal("india");
18: expect(pm.response.json().phone).to.be.a("string");
19: expect(pm.response.json().phone).to.be.equal("1234567890");
20: expect(pm.response.json().courses).to.be.an("array");
21: expect(pm.response.json().courses).to.be.length(2);
22: expect(pm.response.json().courses[0]).to.be.a("string");
23: expect(pm.response.json().courses[0]).to.be.equal("Java");
24: expect(pm.response.json().courses[1]).to.be.a("string");
25: expect(pm.response.json().courses[1]).to.be.equal("Selenium");

```

Asserting array properties

Check if an array is empty and if it contains particular items:

```
{  
    "id": 1,  
    "name": "John",  
    "location": "india",  
    "phone": "1234567890",  
    "courses": [  
        "Java",  
        "Selenium"  
    ]  
  
}  
  
pm.test("Test array properties", () => {  
    //courses includes "Java"  
    pm.expect(jsonData.courses).to.include("Java");  
    //courses array must include all listed  
    pm.expect(jsonData.courses)  
        .to.have.members(["Java", "Selenium"]);  
});
```

Response

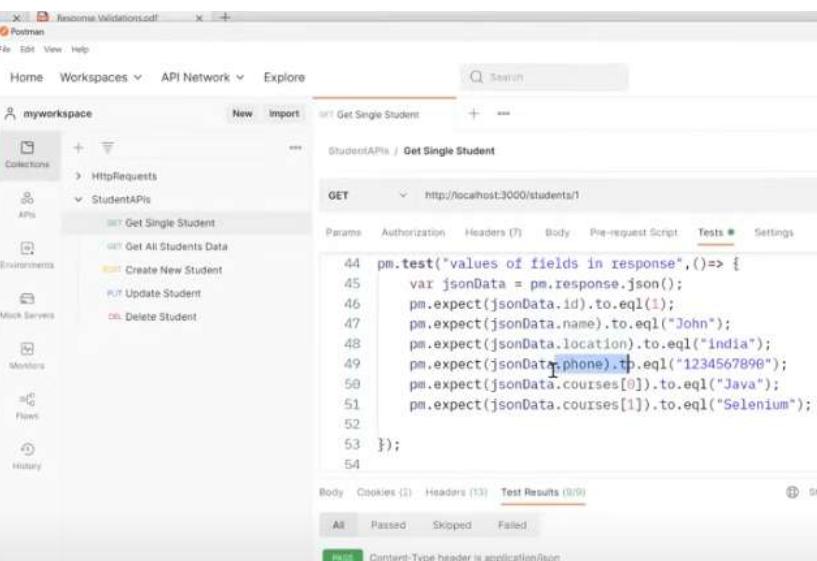
```
{  
    "id": 1,  
    "name": "John",  
    "location": "india",  
    "phone": "1234567890",  
    "courses": [  
        "Java",  
        "Selenium"  
    ]  
}
```

JSON schema

```
var schema={  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "type": "object",  
    "properties": {  
        "id": {  
            "type": "integer"  
        },  
        "name": {  
            "type": "string"  
        },  
        "location": {  
            "type": "string"  
        },  
        "phone": {  
            "type": "string"  
        },  
        "courses": {  
            "type": "array",  
            "items": [  
                {  
                    "type": "string"  
                },  
                {  
                    "type": "string"  
                }  
            ]  
        },  
        "required": [  
            "id",  
            "name",  
            "location",  
            "phone",  
            "courses"  
        ]  
    }  
};
```

Validating JSON fields in Response

```
{  
    "id": 1,  
    "name": "John",  
    "location": "india",  
    "phone": "1234567890",  
    "courses": [  
        "Java",  
        "Selenium"  
    ]  
  
}  
  
pm.test("value of location field is India", ()=> {  
    var jsonData = pm.response.json();  
    pm.expect(jsonData.id).to.eql(1);  
    pm.expect(jsonData.name).to.eql("John");  
    pm.expect(jsonData.location).to.eql("india");  
    pm.expect(jsonData.phone).to.eql("1234567890");  
    pm.expect(jsonData.courses[0]).to.eql("Java");  
    pm.expect(jsonData.courses[1]).to.eql("Selenium");  
});
```



The screenshot shows the Postman interface with a test script for validating JSON response fields. The script uses the `pm` object to access the response JSON and make assertions using the `expect` function from the `chai` library. It checks the value of the `location` field to ensure it is "India".

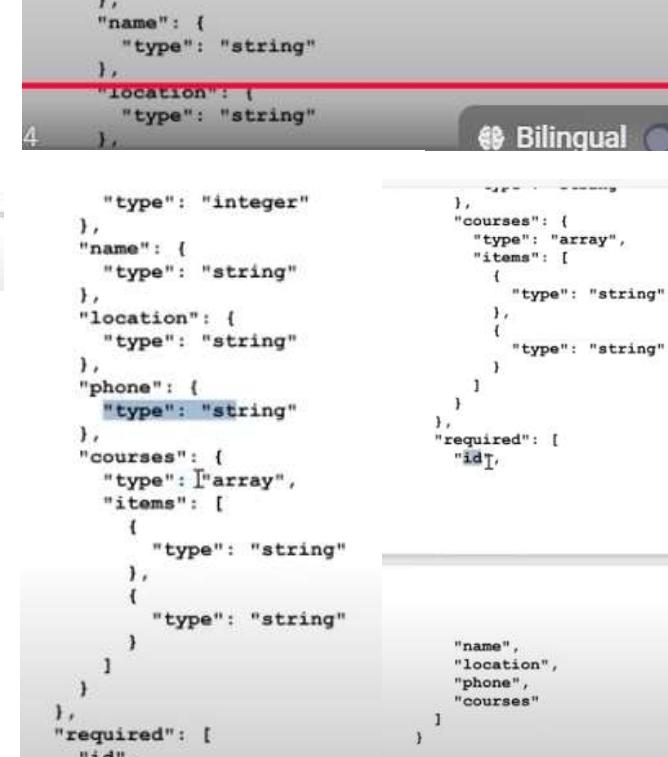
Validating JSON Schema

Response

```
{  
    "id": 1,  
    "name": "John",  
    "location": "india",  
    "phone": "1234567890",  
    "courses": [  
        "Java",  
        "Selenium"  
    ]  
}
```

JSON schema

```
var schema={  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "type": "object",  
    "properties": {  
        "id": {  
            "type": "integer"  
        },  
        "name": {  
            "type": "string"  
        },  
        "location": {  
            "type": "string"  
        },  
        "phone": {  
            "type": "string"  
        },  
        "courses": {  
            "type": "array",  
            "items": [  
                {  
                    "type": "string"  
                },  
                {  
                    "type": "string"  
                }  
            ]  
        },  
        "required": [  
            "id",  
            "name",  
            "location",  
            "phone",  
            "courses"  
        ]  
    }  
};
```



The screenshot shows the Postman interface with a test script for validating JSON schema. The script defines a schema object with properties for `id`, `name`, `location`, `phone`, and `courses`. The `courses` property is defined as an array with string items. The `required` property is set to include all these fields.

```

    "name",
    "location",
    "phone",
    "courses"
}

```

JSON schema Validation

```

pm.test('Schema is valid', function() {
  pm.expect(tv4.validate(jsonData, schema)).to.be.true;
});

```



Google

generate json schema

Q All V Videos I Images N News M Maps > More

Too

About 1,24,00,000 results (0.31 seconds)

<https://www.liquid-technologies.com/online-json-to-s...>

Free Online JSON to JSON Schema Converter

Free Online JSON to JSON Schema Converter ... Uses the sample JSON document to infer a JSON schema. ... Access the online tools directly from your desktop. Download ... JSON Schema to JSON · JSON Schema Editor · JSON Validator · JSON Formatter

<https://www.jsonschema.net/>

JSON Schema Tool

JSONSchema.net is a web application for generating JSON Schema from JSON. Read more. ... jen is a modern CLI for generating JSON Schema from JSON. Available for ...

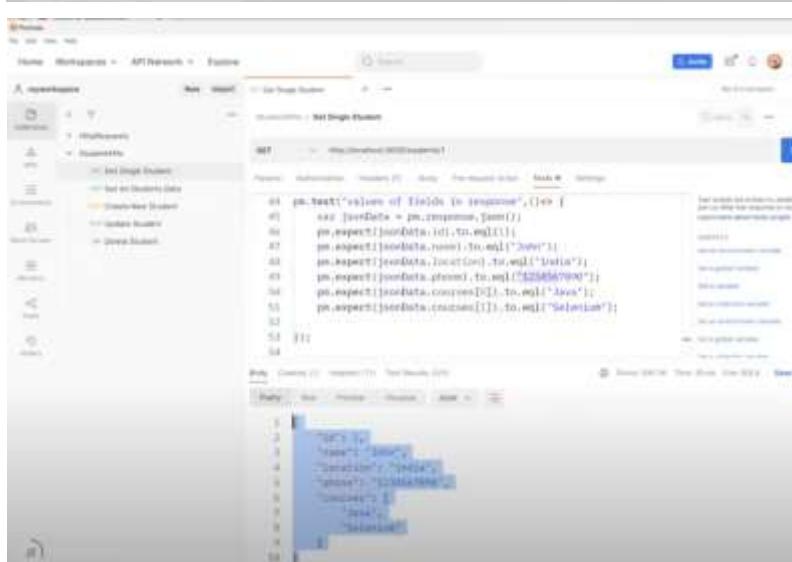
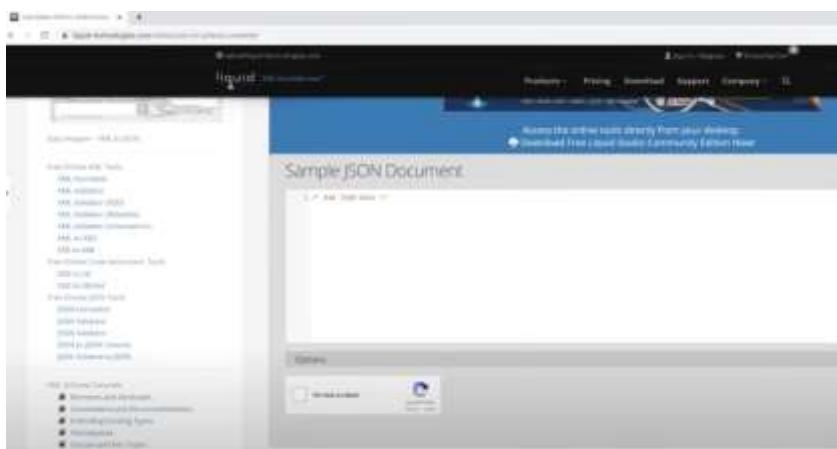
People also search for

json schema validator

json schema example

generate json schema python

generate json schema from json data java



```

        "type": "integer"
      },
      "name": {
        "type": "string"
      },
      "location": {
        "type": "string"
      },
      "phone": {
        "type": "string"
      },
      "courses": {
        "type": "array",
        "items": [
          {
            "type": "string"
          },
          {
            "type": "string"
          }
        ]
      },
      "required": [
        "id",
        "name",
        "location",
        "phone",
        "courses"
      ]
    }
  );

```

Validating JSON fields in Response

```

{
  "id": 1,
  "name": "John",
  "location": "india",
  "phone": "1234567890",
  "courses": [
    "Java",
    "Selenium"
  ]
}

pm.test("value of location field is India", ()=> {
  var jsonData = pm.response.json();
  pm.expect(jsonData.id).to.eql(1);
  pm.expect(jsonData.name).to.eql("John");
  pm.expect(jsonData.location).to.eql("india");
  pm.expect(jsonData.phone).to.eql("1234567890");
  pm.expect(jsonData.courses[0]).to.eql("Java");
  pm.expect(jsonData.courses[1]).to.eql("Selenium");
});

```

Validating JSON Schema

Response

```

{
  "id": 1,
  "name": "John",
  "location": "india",
  "phone": "1234567890",
  "courses": [
    "Java",
    "Selenium"
  ]
}

```

JSON schema

```

var schema={
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "id": {
      "type": "integer"
    }
  }
}

```

```

    "type": "integer"
},
"name": {
    "type": "string"
},
"location": {
    "type": "string"
},
"phone": {
    "type": "string"
},
"courses": {
    "type": "array",
    "items": [
        {
            "type": "string"
        },
        {
            "type": "string"
        }
    ]
},
"required": [
    "id",

```

```

    "name",
    "location",
    "phone",
    "courses"
]
}

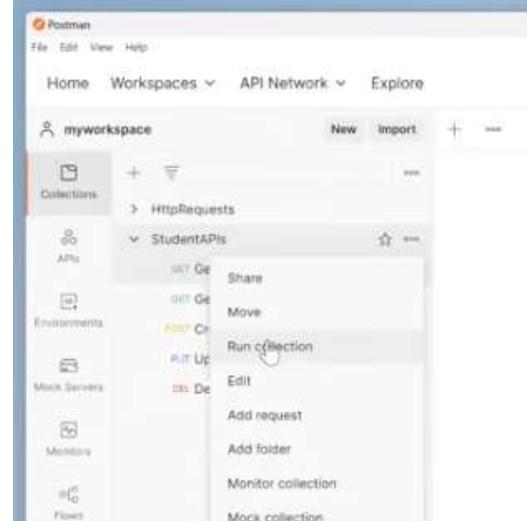
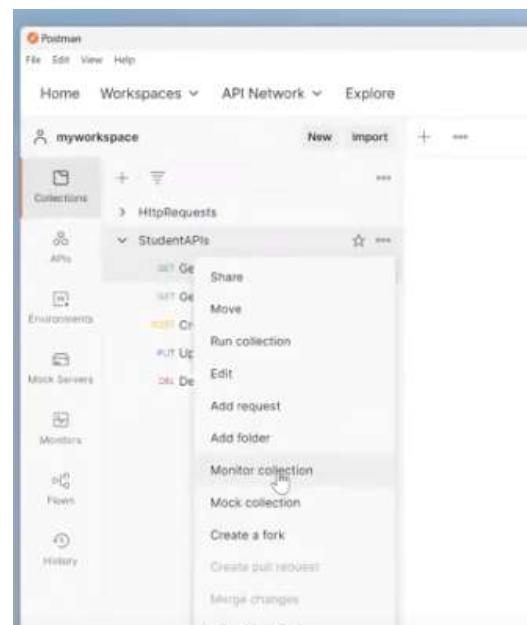
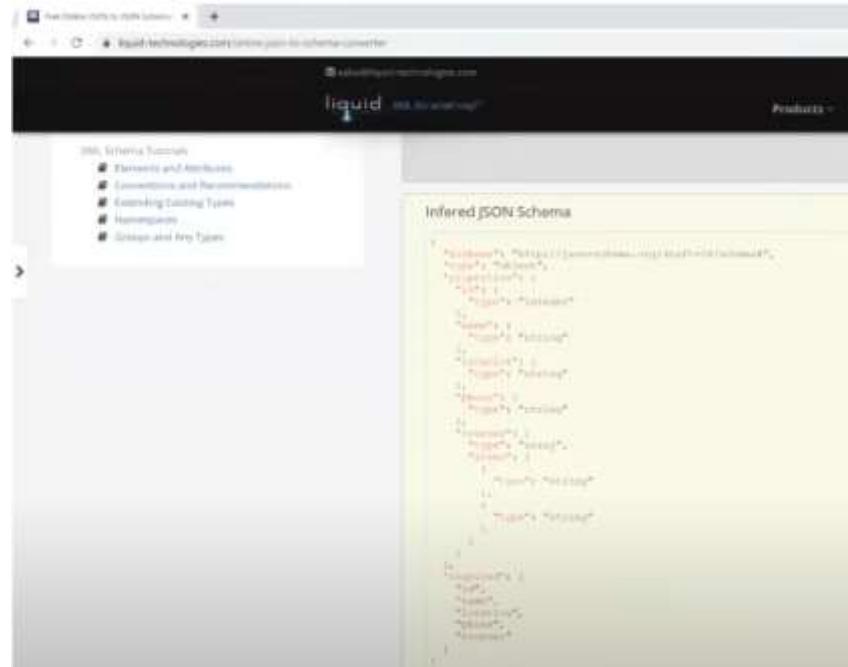
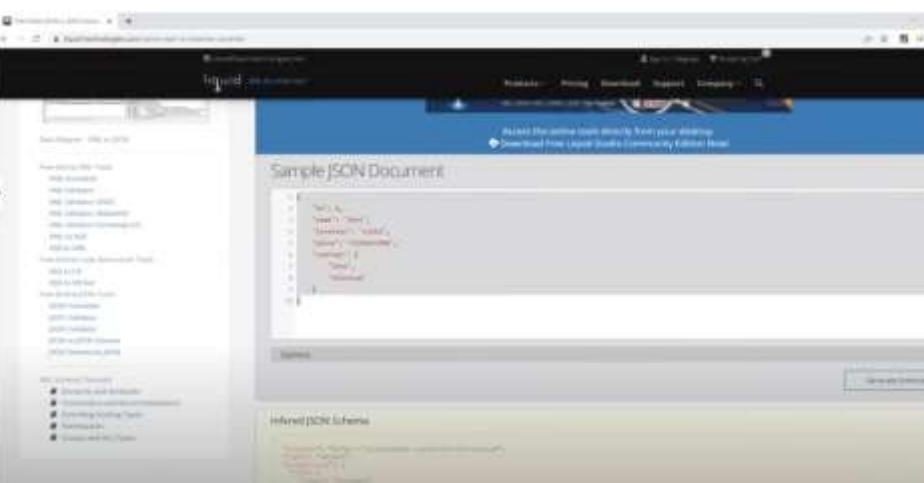
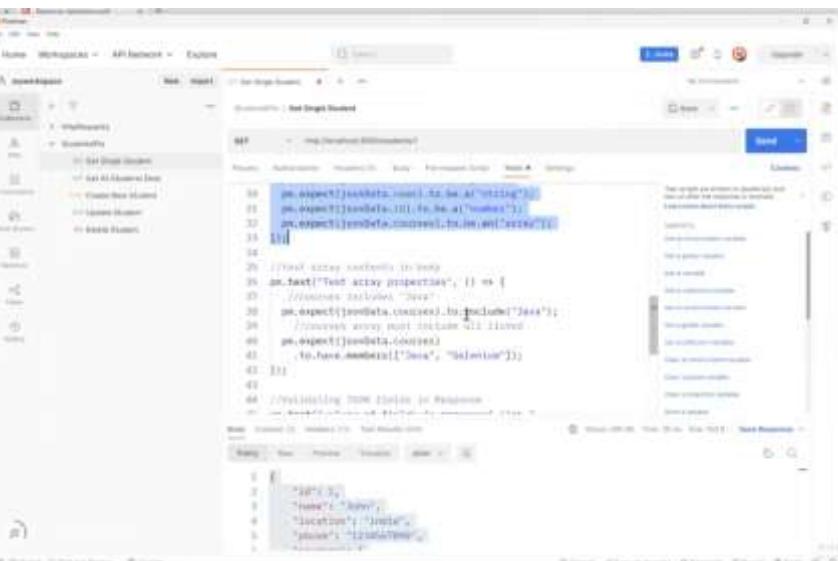
```

JSON schema Validation

```

pm.test('Schema is valid', function() {
    pm.expect(tv4.validate(jsonData, schema)).to.be.true;
});

```



Day 5

Scripts

Pre-request scripts Tests

Day 5

Scripts

Pre-request scripts

Tests

Pre-requestScript ----> Request---->Response-- Tests

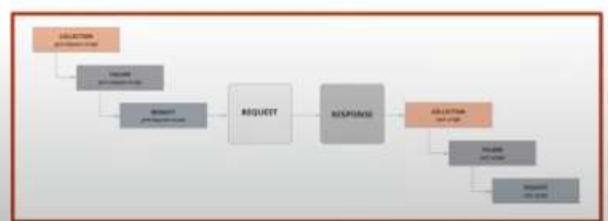
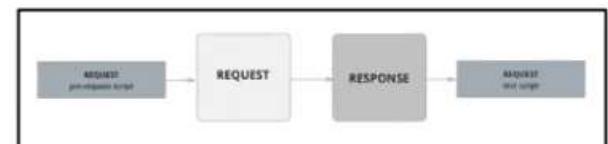
Collection

Folder

Request

Scripts in Postman

- Pre-request Scripts
- Tests Scripts



The screenshot shows the 'Recommendations' collection selected in the left sidebar. On the right, a 'Pre-request Script' step is visible in the 'Script' tab, containing the code: `console.log("pre-request script at collection level")`.

The screenshot shows the Azure portal's API Management section. The top navigation bar includes 'Home', 'Workspaces', 'API Networks', 'Experiments', a search bar, and account settings. Below the navigation is a toolbar with 'New', 'Import', and other icons. The main area displays the 'myapi' collection under 'Collections'. It contains several items: 'Request/Response' (with 'Get-Request' and 'Get-Response' sub-items), 'myapis', and 'myapis_Requirements'. A 'Script' tab is selected, showing the following code:

```
function pre-request(context) { context.log("pre-request script at collection level")}
```

The screenshot shows the 'API Network' interface with the 'Workspaces' tab selected. A search bar at the top right contains the placeholder 'Search workspace...'. Below it, a 'New' button and a 'Import' button are visible. The main area displays a workspace named 'myworkspace'. On the left, a sidebar lists several items under 'Collections': 'APIs', 'HTTP Resources', 'HTTP Request Methods', 'APIs', 'Get Request', 'System APIs', and 'Shared APIs, Request Methods'. The 'HTTP Request Methods' item is expanded, showing 'GET' and 'POST' options. The 'GET' option is selected, revealing its configuration details. The configuration includes a URL template: 'https://api-test.ingress.kube:443/{stage}/{path}'. Below the URL, there are tabs for 'Params', 'Authentication', 'Headers (0)', 'Body', 'Pre-request Script', and 'Tests'. The 'Pre-request Script' tab is active, containing the code: 'console.log("pre-request script at request level");'. At the bottom right of the configuration panel, there are 'Save' and 'Cancel' buttons.

The screenshot shows the Postman application interface. On the left, there's a sidebar with a tree view of collections: 'API Requests' is expanded, showing 'All Requests', 'Put Requests', 'Post Requests', 'Patch Requests', and 'Delete Requests'. Below these are 'Environment', 'Variables', and 'Collection Variables'. The main workspace has a header with tabs: 'Home', 'Responses', 'API Network', 'Editor', and 'Collections'. A search bar is at the top right. The central area displays a request titled 'GET /api/variables'. The 'Headers' tab is selected, showing 'Content-Type: application/json'. The 'Body' tab is also visible. Below the tabs, there's a table with columns: 'Index', 'Name', 'Value', 'Type', 'Format', and 'Preview'. The table contains several rows of environment variables. At the bottom of the table, there are buttons for 'Cancel' and 'Send'.

```
//Global variable  
//pm.globals.unset("userid_global");  
  
//Environment variable  
//pm.environment.unset("userid_qa_env");  
  
//Collection variable  
pm.collectionVariables.unset("userid_collect");
```

capture the values from variables

```
console.log(pm.globals.get("userid_global"));
console.log(pm.environment.get("userid_qa_env"));
console.log(pm.collectionVariables.get("userid_col"));
console.log(pm.variables.get("url_local"));
```



```
1 console.log(pe_globals.get("user1_id_global"));
2 console.log(pe_environment.get("user1_qs_env"));
3 console.log(pe_collectionvariable.get("user1_cv_name"));
4 console.log(pe_variables.get("user1_local"));
5
6
7 pe_globals.set("user1_global");
8
9
10 pe_environment.set("user1_qs_env");
11
12 pe_collectionvariable.set("user1_cv_name");
13 pe_variables.set("user1_local");
```

The screenshot shows the 'API Network' interface with a selected 'GET' request to 'https://api.github.com/repos/microsoft/TypeScript/branches'. The 'Headers' tab displays 'User-Agent: node-fetch/1.0 [es6]'. The 'Body' tab is empty. The 'Script' tab contains a 'Pre-request Script' with the following code:

```
1 //Global variable
2 process.env.NODE_ENV = "test"
3
4 //Development variable
5 process.env.NODE_ENV = "development"
6
7 //Collection variable
8 process.env.COLLECTION_NAME = "node_collect"
9
```

Creating variables using pre-request scripts

```
//Local variables  
pm.variables.set("url_local","https://regres.in");
```

```
//Global variable  
pm.globals.set("userid_global","2");
```

```
//Environement variable  
pm.environment.set("userid_qa_env","2");
```

```
//Collection variable  
pm.collectionVariables.set("userid_collect","2");
```

what?
why?
where?

Scope

Global -- accessible in workspace
Collection -- accessible within collection
Environment -- accessible in all collections

`local` -- accessible only within request (specific to equest)

Local — accessible only within a specific geographic area

```
//Local variables  
pm variables set("url_local" "https://recoos.io")
```

www.nature.com/scientificreports/

url_global
url_collect
url_sg_env

refering variables {{variable}}
creating variables using pre-request scripts

Postman Environment Variables

```
//pm.environment.unset("userid_qa_env");
//Collection variable
pm.collectionVariables.unset("userid_collect");
capture the values from variables
-----
console.log(pm.globals.get("userid_global"));
console.log(pm.environment.get("userid_qa_env"));
console.log(pm.collectionVariables.get("userid_collect"));
console.log(pm.variables.get("url_local"));

set - global, env, collection, local
unset - global, env, collection, local
get - global, env, collection, local
```

Day-6

Chaining of APIs

Postman Collection Chaining

This screenshot shows the Postman interface with a collection named "StudentAPI-Chaining". The left sidebar lists collections like ReqRes_httpRequests, ReqRes_httpRequestsVariables, and StudentAPIs. The main area displays the "StudentAPI-Chaining" collection with its various requests and pre-request scripts.

Postman Collection Share

A context menu is open on the "StudentAPI" collection, showing options like "Share", "Move", "Run collection", "Edit", "Add request", "Add folder", and "Monitor collection".

Postman Collection Edit

A context menu is open on the "StudentAPI" collection, showing options like "Edit", "Add request", "Add folder", "Monitor collection", "Create a fork", "Create pull request", "Merge changes", "View changelog", "View documentation", "Rename", "Delete", and "Duplicate".

File Explorer

The Windows File Explorer is shown, displaying files in the "AutomationPractice\jsonfiles" folder. Visible files include employees.json, employees.json.txt, pondata.json, store.json, and students.json.

Terminal

```
Microsoft Windows [Version 10.0.25158.1000]
(c) Microsoft Corporation. All rights reserved.

C:\AutomationPractice\jsonfiles>json-server students.json
\{^_^\}/ hi!
Loading students.json
Done

Resources
http://localhost:3000/students

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
```

Postman Create New Student

The Postman interface shows a POST request to "http://localhost:3000/students". The "Body" tab is selected, showing a JSON payload:

```
{
  "name": "Boris",
  "location": "Russia",
  "phone": "1234567890",
  "courses": [
    "Java",
    "Selenium"
  ],
  "id": 2,
  "name": "Kim",
  "location": "US",
  "phone": "9887654321",
  "courses": [
    "Python",
    "Applum"
  ]
}
```

The screenshot shows the Postman application interface. At the top, there are navigation tabs: Home, Workspaces, API Network, and Explore. Below the navigation is a search bar labeled "Search Postman". The main area is titled "myworkspace". On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Services, and Monitors. Under "Collections", "myworkspace" is expanded, showing "ReqRes_httpRequests", "ReqRes_httpRequestsVariables", "ReqRes_httpRequestesWorkflow", "StudentAPIs", and "StudentAPIs-Chaining". "StudentAPIs-Chaining" is also expanded, showing "Create New Student", "Get Single Student", "Delete Student", and "StudentAPIs_ResponseValidations". The "Create New Student" item is currently selected. To the right of the sidebar, the URL "StudentAPIs-Chaining / Create New Student" is displayed. The main content area shows a "POST" request to "http://localhost:3000/students". Below the request, under "Params", is a pre-request script:

```
1 var jsonData=JSON.parse(responseBody);
2 pm.environment.set("id",jsonData.id);
3
```

The screenshot shows a browser window with a navigation bar at the top. The main area displays a test runner interface. On the left, there's a sidebar with icons for Home, Workspaces, API Network, and Options. Below that is a tree view of workspace contents, including 'HelloWorld', 'HelloWorldController', 'HelloWorldProperties', 'StudentAPI', 'StudentAPI-Chaining', and 'Data New Item'. Under 'StudentAPI-Chaining', there are two items: 'Data New Item' and 'Set Single Student'. The 'Set Single Student' item is selected and expanded, showing its code and a red error icon. The code is as follows:

```
    11 pm.expect(jsonData.id).to.be.a('number');
    12 pm.expect(jsonData.courses).to.be.an('array');
    13 });
    14
    15 pm.test("Test array properties", () => {
    16   //course includes "Java"
    17   pm.expect(jsonData.courses).to.include("Java");
    18   //courses array must include all listed
    19   pm.expect(jsonData.courses)
    20     .to.have.members(["Java", "Selenium"]);
    21 });
    22
```



The screenshot shows the Postman application interface. On the left, there's a sidebar with a tree view of collections and environments. The main area displays a collection named 'myapis'. Under this collection, several API endpoints are listed: 'ReadAll_Resources', 'ReadOne_Resources', 'ReadAll_Responses', 'ReadOne_Response', 'CreateNew_Student', 'GetSingle_Student', and 'Delete_Student'. The 'GetSingle_Student' endpoint is currently selected. To the right of the endpoint list, there are tabs for 'Info', 'Tests', 'Pre-request Script', 'Script', 'Body', 'Headers', 'Auth', and 'Settings'. The 'Tests' tab is active, showing a failing test script:

```
1 pm.expect("status code is 200", () => {
2   pm.response.to.have.status(200);
3 });
4
5
```

The screenshot shows the NetworkMiner interface with the following details:

- File Menu:** Home, Workspaces, AFN Sensors, Export.
- Search Bar:** Search, Filter, Find in selected, Scan Selected, Scan All, Scan All.
- Tool Buttons:** New Project, Open Project, Save Project, Save As, Print, Help.
- Left Sidebar:** A hierarchical tree view of network connections, including:
 - Media, IP Requests
 - Media, IP Requests Received
 - Media, IP Requests Received List
 - Discovery
 - Discovered Devices
 - Device New Created
 - Old Device Created
 - Old Device Deleted
 - Device Deleted
 - Devices, Discovered Devices
- Central Panel:** Shows a list of captured network packets. The first few entries are:
 - 427. https://cloud-mirror.apl.443/100-443.html
 - 428. https://cloud-mirror.apl.443/100-443.html
 - 429. https://cloud-mirror.apl.443/100-443.html?n=0
 - 427. https://cloud-mirror.apl.443/100-443.html?n=0
 - 428. https://cloud-mirror.apl.443/100-443.html?n=0
 - 429. https://cloud-mirror.apl.443/100-443.html?n=0
 - 427. https://cloud-mirror.apl.443/100-443.html?n=0
 - 428. https://cloud-mirror.apl.443/100-443.html?n=0
 - 429. https://cloud-mirror.apl.443/100-443.html?n=0
- Bottom Status Bar:** Shows file paths and file sizes.



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

pm.test("Content-Type header is present", () => {
    pm.response.to.have.header("Content-Type");
});

pm.expect(pm.response.headers.get('Content-Type')).to.eql('application/json; charset=utf-8');
```

Day-6

Chaining of API's

Student API

Gorest API

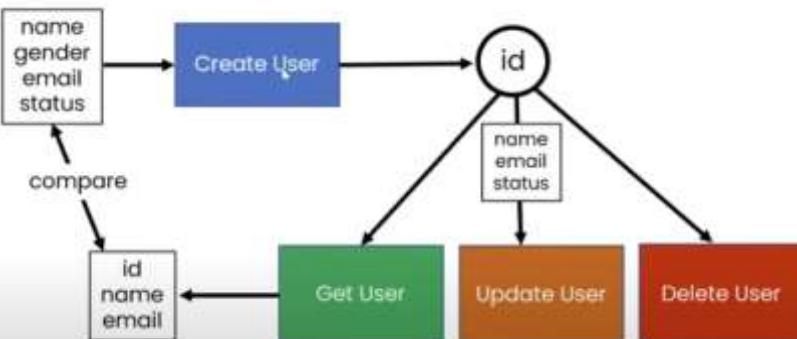
url: <https://gorest.co.in>



PQST	/public/v2/users	Create a new user
GET	/public/v2/users/{id}	Get user details
PUT PATCH	/public/v2/users/{id}	Update user details
DELETE	/public/v2/users/{id}	Delete user

Token: c35e10e748c6f13775527bcf204e9929b4c9f4b995a8ee253eec46ded57b06

API Version 2
<ul style="list-style-type: none">PathResponse header: <code>Content-Type: application/json</code>HTTP Response code contains the actual response code + <code>_200</code>, containing the application information, response body contains the results
API Version 1
<ul style="list-style-type: none">PathResponse header: <code>Content-Type: application/json</code>HTTP status code contains the actual response code + <code>_200</code>, containing the application information, response body contains the actual results
API Version 0
<ul style="list-style-type: none">PathResponse header: <code>Content-Type: application/json</code>HTTP status code is always <code>200</code>, containing the actual response code + <code>_200</code>, containing the application information, response body contains the actual results
authentication
Under RESTful applications, RESTful APIs are usually designed which require authentication or authorization to access user. This <code>auth</code> module should come with some sort of authentication mechanism. A common way is to send a request with token header to authenticate the user. Most of the services token can be generated through login and authenticated to user. API request should include header like <code>HTTP Authorization</code> for the request to make it secure.
There are different ways to generate tokens.
<ul style="list-style-type: none">Session based: The session token is stored in the database.Client generation: The access token is used as a cookie parameter in the API URL, e.g. <code>http://api.example.com/v1/users/1?access_token=1234567890123456789012345678901234567890</code>.Server generation: The access token is generated by the controller API or application server and sent back API controller <code>HTTP 200 OK</code>, according to the client's request.



url: <https://gorest.co.in>

POST	/public/v2/users	Create a new user
GET	/public/v2/users/{id}	Get user details
PUT PATCH	/public/v2/users/{id}	Update user details
DELETE	/public/v2/users/{id}	Delete user

Token: c35e10e748c8f113775527bcdf204e9929b4c9f4b995c8ee253eac48cad57b06

Request Body

```
[  
  "name": "scott",  
  "gender": "male",  
  "email": "abc@gmail.com",  
  "status": "inactive"  
]
```

The screenshot shows the Postman application interface. In the left sidebar, under 'myworkspace', there is a 'Collections' section with a 'GorestAPI-Chaining' folder expanded, containing a 'Create User' request. The main workspace shows a 'Create User' request with the following details:

- Method:** POST
- URL:** https://gorest.co.in/public/v2/users
- Body:** form-data
- Params:** none
- Authorization:** none
- Headers (B):** none

The request body is defined as follows:

```
1 "name": "Hott",
2 "gender": "male",
3 "email": "abc@gmail.com",
4 "status": "inactive"
```

The screenshot shows the Postman application interface. The left sidebar displays a workspace named 'myworkspace' containing several collections: 'StudentAPI-Chaining', 'CreateUser', 'GetUserDetails', 'UpdateUser', and 'DeleteUser'. The 'DeleteUser' collection is currently selected and highlighted in red. Within this collection, a specific request titled 'Delete User' is selected and highlighted in red. The main panel shows the request details: Method 'DELETE', URL 'https://api.getpostman.com/collections/...', and a preview section. The preview section includes tabs for 'Type' (selected), 'Header', 'Body', 'Pre-request Script', 'Tests', and 'Variables'. Below the preview, there is a note about authorization: 'This authorization header will be automatically generated when you send the request.' and a link 'Learn more about authorization'. At the bottom of the main panel, there is a 'Resource' section.

The screenshot shows the API Network interface with the following details:

- API Network** tab is selected.
- API** dropdown is set to **myworkspace**.
- Create User** is selected in the **API** list.
- Method**: **POST**, **URL**: <https://openqa.influxdb.com>
- Headers**: **Content-Type**: application/json
- Body**:

```
var random=Math.random().toString(36).substring(2);  
var usermails="jim"+random+"@gmail.com";  
var usernam="jim"+random;  
var userpass="123456789";  
ps.environment.set("email_env",usermails);  
ps.environment.set("name_env",usernam);  
ps.environment.set("pass_env",userpass);
```

API Chaining

This section illustrates how to chain multiple API requests to perform complex operations.

Example 1: Students API

Step1: Send Post request (create student) then call get "Id" from response.

```
http://localhost:1000/students
```

Request body

```
{
  "name": "Scott",
  "location": "France",
  "phone": "123456",
  "courses": [
    "C",
    "C++"
  ]
}
```

Tests

Include below script in Tests to collect Id of environment variable.

```
//Capturing id from response
var jsonData = JSON.parse(responseBody);
pm.environment.set("id", jsonData.id);
```

Step2: Send Get request (display student) using the "Id" captured from previous request.

```
http://localhost:1000/students/100
```

API Chaining

The API Chaining feature allows you to define a sequence of requests that can be triggered by a single endpoint. It includes a visual editor for defining the flow and a preview pane for testing the chained requests.

Chained Requests

- Create User:** POST https://geniesys.co/public/v2/users
- Get User Details:** GET https://geniesys.co/public/v2/users/{userId_env}
- Update User:** PUT https://geniesys.co/public/v2/users/{userId_env}
- Delete User:** DELETE https://geniesys.co/public/v2/users/{userId_env}

Chained Requests Flow:

1. Create User
2. Get User Details
3. Update User
4. Delete User

Chained Requests Settings:

- Run Settings: Run in parallel
- Advanced settings: Save responses, Every service return, Run collection without carry status codes, Save results after collection run.

Example 2: GoRest API (Users)

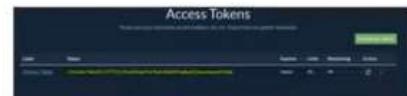
<https://gorest.co.in/>

Do not post your personal details (name, email, phone, photo etc...).
The request must contain "Authorization" header with value "Bearer <access token>".
Required methods: PUT, POST, PATCH, DELETE needs access token, which needs to be passed with "Authorization" header as Bearer token.
API version: v2
Documentation: [View documentation](#)

Authentication Required

Login with your social account:

[Google](#) [Facebook](#) [Microsoft](#)

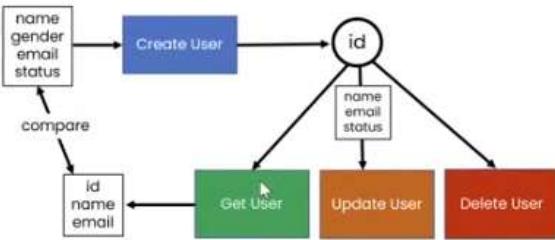


Access Token :

c35e10e748c6f13775527bcf204e9929b4c9f4b995a8ee25
3eec46aed57b06

<https://gorest.co.in>

POST	/public/v2/users	Create a new user
GET	/public/v2/users/23	Get user details
PUT PATCH	/public/v2/users/23	Update user details
DELETE	/public/v2/users/23	Delete user



Create User (Post)

<{{url}}/public/v2/users>

Request Body

```
{
  "name": "{{name_env}}",
  "gender": "male",
  "email": "{{email_env}}",
  "status": "inactive"
}
```

Create User (Post)

<{{url}}/public/v2/users>

Request Body

```
{
  "name": "{{name_env}}",
  "gender": "male",
  "email": "{{email_env}}",
  "status": "inactive"
}
```

Pre-Request script

```
var random=Math.random().toString(36).substring(2);
var useremail="Jim"+random+"@gmail.com";
var username="Jim"+random;
```

```
//console.log(useremail); // print to know value
//console.log(username); // print to know value

pm.environment.set("email_env",useremail);
pm.environment.set("name_env",username)
```

I

Tests

```
/*Capturing id from response & Set as environment var
var jsonData = JSON.parse(responseBody);
pm.environment.set("userid_env",jsonData.id);
```

Get User details (Get)

[#### Tests]({{url}}/public/v2/users/{{userid_env}}{{url}}/public/v2/users/{{userid_env}}</p></div>
<div data-bbox=)

```
/*Validating JSON fields in Response
pm.test("values of fields in response", () => {
  var jsonData = pm.response.json();

  pm.expect(jsonData.id).to.eql(pm.environment.get("userid_env"));
  pm.expect(jsonData.email).to.eql(pm.environment.get("email_env"));
  pm.expect(jsonData.name).to.eql(pm.environment.get("name_env"));
});
```

Update User details (Put)

[#### Request Body]({{url}}/public/v2/users/{{userid_env}}{{url}}/public/v2/users/{{userid_env}}</p></div>
<div data-bbox=)

```
{
  "name": "{{name_env}}",
  "email": "{{email_env}}",
  "status": "active"
}
```

Pre-Request script

```
var random=Math.random().toString(36).substring(2);
var useremail="Jim"+random+"@gmail.com";
var username="Jim"+random;

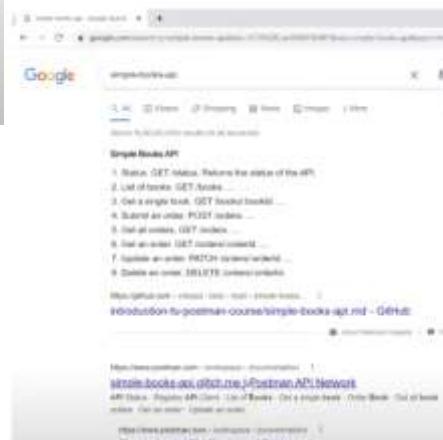
pm.environment.set("email_env",useremail);
pm.environment.set("name_env",username)
```

Delete user (Delete)

[## Day-7]({{url}}/public/v2/users/{{userid_env}}{{url}}/public/v2/users/{{userid_env}}</p></div>
<div data-bbox=)

Books API

<https://simple-books-api.glitch.me>



Books API

This API allows you to review a book.

The API is available at <https://simple-books-api.glitch.me>

API Authentication

To submit or view an order, you need to register your API client.

[POST /api-clients/](#)

The request body needs to be in JSON format and include the following properties:

- clientName - String
- clientEmail - String

Example

```
{"clientName": "Postman",
"clientEmail": "client@example.com"}
```

The response body will contain the access token. The access token is valid for 7 days.

Possible errors

Status code 409 - "API client already registered" by changing the values for clientName and clientEmail to something else.

Endpoints

Status

[GET /status](#)

Returns the status of the API.

List of books

[GET /books](#)

Returns a list of books.

Optional query parameters:

- type: fiction or non-fiction
- limit: a number between 1 and 20.

Get a single book

[GET /books/:bookId](#)

Retrieve detailed information about a book.

Status

[GET /status](#)

Returns the status of the API.

List of books

[GET /books](#)

Returns a list of books.

Optional query parameters:

- type: fiction or non-fiction
- limit: a number between 1 and 20.

Get a single book

[GET /books/:bookId](#)

Retrieves detailed information about a book.

Submit an order

[POST /orders](#)

Allows you to submit a new order. Requires authentication.

The request body needs to be in JSON format and include the following properties:

- orderId - Integer - Required
- customerId - String - Required

Example

```
orderId: 1,
customerId: 1,
customerName: "John Doe",
```

```
customerAddress: "123 Main St.",
```

```
customerEmail: "john.doe@example.com"
```

My response body will contain the order id.

Get all orders

[GET /orders](#)

Allows you to view all orders. Requires authentication.

Get an order

[GET /orders/:orderId](#)

Allows you to view an existing order. Requires authentication.

Update an order

[PATCH /orders/:orderId](#)

Updates an existing order. Requires authentication.

The request body needs to be in JSON format and allows you to update the following properties:

- customerId - String

Example

```
orderId: 1,
customerId: 1,
```

```
customerName: "John Doe",
```

```
customerAddress: "123 Main St.",
```

```
customerEmail: "john.doe@example.com"
```

Delete an order

[DELETE /orders/:orderId](#)

Deletes an existing order. Requires authentication.

Bilingual



Allows you to view all orders. Requires authentication.

Get an order

GET /orders/:orderId

Allows you to view an existing order. Requires authentication.

Update an order

PATCH /orders/:orderId

Update an existing order. Requires authentication.

The request body needs to be in JSON format and allows you to update the following properties:

- customerName - String

Example

```
PATCH /orders/PF6MF1P0cuHwobZcg0y5
Authorization: Bearer <YOUR TOKEN>

{
  "customerName": "John"
}
```

Delete an order

DELETE /orders/:orderId

Delete an existing order. Requires authentication.

The request body needs to be empty.

Example

```
DELETE /orders/PFGHfJPDcuHwobZcg0y5
Authorization: Bearer <YOUR TOKEN>
```

The screenshot shows the API Network interface with the 'myworkspace' collection selected. A 'Create Token' endpoint is highlighted under the 'BooksAPI' collection. The 'POST' method is selected, and the URL is `https://sample-books-api.gitch.me/api-client/`. The 'Body' tab shows a JSON payload with fields: `"clientName": "Postman"`, `"clientEmail": "valentin@example.com"`, and `"clientSecret": null`.

The screenshot shows the API Network interface with the 'myworkspace' collection selected. A 'Create Token' endpoint is highlighted under the 'BooksAPI' collection. The 'POST' method is selected, and the URL is `https://sample-books-api.gitch.me/api-client/`. The 'Body' tab shows a JSON payload with fields: `"clientName": "Postman"`, `"clientEmail": "valentin@example.com"`, and `"clientSecret": null`.

The screenshot shows the API Network interface with the 'myworkspace' collection selected. A 'Create Token' endpoint is highlighted under the 'BooksAPI' collection. The 'POST' method is selected, and the URL is `https://sample-books-api.gitch.me/api-client/`. The 'Body' tab shows a JSON payload with fields: `"clientName": "Postman"`, `"clientEmail": "valentin@example.com"`, and `"clientSecret": null`.

The screenshot shows the API Network interface with the 'myworkspace' collection selected. A 'Create Token' endpoint is highlighted under the 'BooksAPI' collection. The 'POST' method is selected, and the URL is `https://sample-books-api.gitch.me/api-client/`. The 'Body' tab shows a JSON payload with fields: `"clientName": "Postman"`, `"clientEmail": "valentin@example.com"`, and `"clientSecret": null`.

The screenshot shows the API Network interface with the 'myworkspace' collection selected. A 'Create Token' endpoint is highlighted under the 'BooksAPI' collection. The 'POST' method is selected, and the URL is `https://sample-books-api.gitch.me/api-client/`. The 'Body' tab shows a JSON payload with fields: `"clientName": "Postman"`, `"clientEmail": "valentin@example.com"`, and `"clientSecret": null`.

The screenshot shows the API Network interface with the 'myworkspace' collection selected. A 'Create Token' endpoint is highlighted under the 'BooksAPI' collection. The 'POST' method is selected, and the URL is `https://sample-books-api.gitch.me/api-client/`. The 'Body' tab shows a JSON payload with fields: `"clientName": "Postman"`, `"clientEmail": "valentin@example.com"`, and `"clientSecret": null`.

Books API Data Driven Test

(Data Parameters – JSON/csv)

Step 1:

Request Type: POST

URL: <https://simple-books-api.glitch.me/orders/>

Request Body:

```
{
  "bookId": "{{(BookID)}",
  "customerName": "{{(CustomerName)}}"
}
```

Tests

```
pm.test("Status code is 201", () => {
  pm.response.to.have.status(201);
});

var jsonData=JSON.parse(responseBody);
pm.environment.set("orderid_env",jsonData.orderId);
```

Step 2:

Request Type: GET

URL: https://simple-books-api.glitch.me/orders/{{orderid_env}}

Tests

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

pm.test("check orderID present in response body", () =>{
  var jsonData=pm.response.json();
  pm.expect(jsonData.id).to.eql(pm.environment.get("orderid_env"));
});
```

Step 3:

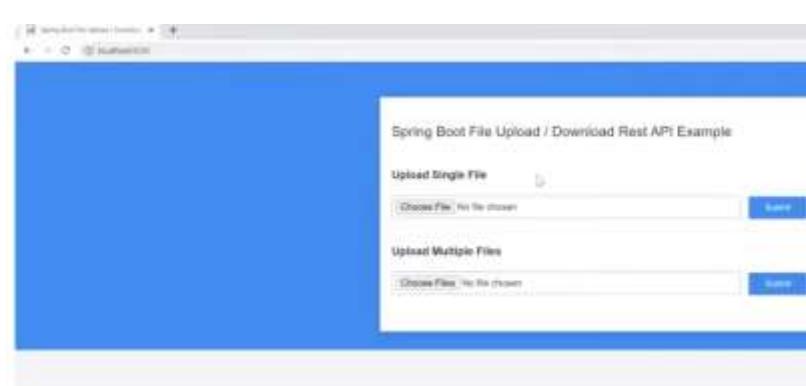
Request Type: DELETE

URL: https://simple-books-api.glitch.me/orders/{{orderid_env}}

Tests

```
pm.test("Status code is 204", () => {
  pm.response.to.have.status(204);
});

//Unset orderid environment var
pm.environment.unset("orderid_env");
```



Day-8

1) Upload file

Post : <http://localhost:8080/uploadFile>

POST: <http://localhost:8080/uploadMultipleFiles>

POST - http://localhost:8080/uploadFile

Body (x)

KEY	VALUE	DESCRIPTION
file	Select files	Description

POST - http://localhost:8080/uploadFile

Body (x)

KEY	VALUE	DESCRIPTION
file	Test1.txt	Description

POST - http://localhost:8080/uploadFile

Body (x)

KEY	VALUE	DESCRIPTION
file	Test1.txt	Description

Pretty

```
{  
  "fileName": "Test1.txt",  
  "fileDownloadUrl": "http://localhost:8080/downloadFile/Test1.txt",  
  "fileType": "text/plain",  
  "size": 70  
}
```

Home Workspaces API Network Explore

myworkspace

New Import **MultipleFileUpload** + -

Processor | MultipleFileUpload

POST → http://localhost:8080/api/multiplefiles

Params Authorization Headers (0) Body Pre-request Script Tools Settings

none form-data x-www-form-urlencoded raw binary script

KEY	VALUE
File	File

Response

Home Workspaces API Network Explore

myworkspace

New Import **MultipleFileDownload** + -

Processor | MultipleFileDownload

POST → http://localhost:8080/api/multiplefiles

Params Authorization Headers (0) Body Pre-request Script Tools Settings

none form-data x-www-form-urlencoded raw binary script

KEY	VALUE
File	File selected: None

Body Cookies (0) Headers (0) Test Results

Line	Value
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100
101	101
102	102
103	103
104	104
105	105
106	106
107	107
108	108
109	109
110	110
111	111
112	112
113	113
114	114
115	115
116	116
117	117
118	118
119	119
120	120
121	121
122	122
123	123
124	124
125	125
126	126
127	127
128	128
129	129
130	130
131	131
132	132
133	133
134	134
135	135
136	136
137	137
138	138
139	139
140	140
141	141
142	142
143	143
144	144
145	145
146	146
147	147
148	148
149	149
150	150
151	151
152	152
153	153
154	154
155	155
156	156
157	157
158	158
159	159
160	160
161	161
162	162
163	163
164	164
165	165
166	166
167	167
168	168
169	169
170	170
171	171
172	172
173	173
174	174
175	175
176	176
177	177
178	178
179	179
180	180
181	181
182	182
183	183
184	184
185	185
186	186
187	187
188	188
189	189
190	190
191	191
192	192
193	193
194	194
195	195
196	196
197	197
198	198
199	199
200	200
201	201
202	202
203	203
204	204
205	205
206	206
207	207
208	208
209	209
210	210
211	211
212	212
213	213
214	214
215	215
216	216
217	217
218	218
219	219
220	220
221	221
222	222
223	223
224	224
225	225
226	226
227	227
228	228
229	229
230	230
231	231
232	232
233	233
234	234
235	235
236	236
237	237
238	238
239	239
240	240
241	241
242	242
243	243
244	244
245	245
246	246
247	247
248	248
249	249
250	250
251	251
252	252
253	253
254	254
255	255
256	256
257	257
258	258
259	259
260	260
261	261
262	262
263	263
264	264
265	265
266	266
267	267
268	268
269	269
270	270
271	271
272	272
273	273
274	274
275	275
276	276
277	277
278	278
279	279
280	280
281	281
282	282
283	283
284	284
285	285
286	286
287	287
288	288
289	289
290	290
291	291
292	292
293	293
294	294
295	295
296	296
297	297
298	298
299	299
300	300
301	301
302	302
303	303
304	304
305	305
306	306
307	307
308	308
309	309
310	310
311	311
312	312
313	313
314	314
315	315
316	316
317	317
318	318
319	319
320	320
321	321
322	322
323	323
324	324
325	325
326	326
327	327
328	328
329	329
330	330
331	331
332	332
333	333
334	334
335	335
336	336
337	337
338	338
339	339
340	340
341	341
342	342
343	343
344	344
345	345
346	346
347	347
348	348
349	349
350	350
351	351
352	352
353	353
354	354
355	355
356	356
357	357
358	358
359	359
360	360
361	361
362	362
363	363
364	364
365	365
366	366
367	367
368	368
369	369
370	370
371	371
372	372
373	373
374	374
375	375
376	376
377	377
378	378
379	379
380	380
381	381
382	382
383	383
384	384
385	385
386	386
387	387
388	388
389	389
390	390
391	391
392	392
393	393
394	394
395	395
396	396
397	397
398	398
399	399
400	400
401	401
402	402
403	403
404	404
405	405
406	406
407	407
408	408
409	409
410	410
411	411
412	412
413	413
414	414
415	415
416	416
417	417
418	418
419	419
420	420
421	421
422	422
423	423
424	424
425	425
426	426
427	427
428	428
429	429
430	430
431	431
432	432
433	433
434	434
435	435
436	436
437	437
438	438
439	439
440	440
441	441
442	442
443	443
444	444
445	445
446	446
447	447
448	448
449	449
450	450
451	451
452	452
453	453
454	454
455	455
456	456
457	457
458	458
459	459
460	460
461	461
462	462
463	463
464	464
465	465
466	466
467	467
468	468
469	469
470	470
471	471
472	472
473	473
474	474
475	475
476	476
477	477
478	478
479	479
480	480
481	481
482	482
483	483
484	484
485	485
486	486
487	487
488	488
489	489
490	490
491	491
492	492
493	493
494	494
495	495
496	496
497	497
498	498
499	499
500	500

Home Workspaces API Network Explore

myworkspace

New Import **BasicAuthentication** + -

Processor | BasicAuthentication

GET → http://localhost:8080/api/basicauth

Params Authorization Headers (0) Body Pre-request Script Tools Settings

Type Basic Auth → Username:password → Password:

This authentication needs no pre-authentication header after you send the request. Learn more about authentication.

Body Cookies (0) Headers (0) Test Results

Line	Value
1	{ "authenticated": true }
2	}

The Internet Not secure the-internet.herokuapp.com/basic_auth

Basic Auth

Congratulations! You must have the proper credentials.

Powered by Elemental Selenium

Home Workspaces API Network Explore

myworkspace

The screenshot shows the API Management interface with a selected endpoint named "Authentication". The configuration pane displays fields for "Type" (Basic Auth), "Username" (pavan), and "Password" (password). A note indicates that the password is stored in plain text and can be copied from the browser's developer tools. There are also checkboxes for "Use this password for all requests" and "Allow Password".

Postman Authorizing requests

The screenshot shows the Postman interface with the "Authorization" tab selected. A dropdown menu lists various authentication methods: "Nil Auth", "API Key", "Bearer Token", "Basic Auth" (which is highlighted with a red box), "Digest Auth", "OAuth 1.0", "OAuth 2.0", "Hawk Authentication", "AWS Signature", "NTLM Authentication", and "Akamai EdgeGrid".

The screenshot shows the GitHub REST API documentation page. The main content area features a "Quickstart" button and sections for "Getting started with the REST API", "Resources in the REST API", "What's new in Codespaces for Organizations", and "Other authentication methods". On the left sidebar, there are links for "All products", "REST API", "Overview", "Guides", "REST API references", "Actions", "Activity", "Apps", "Billing", "Branches", "Checkins", "Codes of conduct", and "Code Scanning".

The screenshot shows the GitHub user profile page for "pavantraining". It includes a profile picture, a bio, and links for "Edit profile", "GitHub Sponsors", and "Following". The user has 1447 followers. Below the profile, there are sections for "Achievements" and "Attachments". At the bottom, there are links for "ID Connect", "Repositories", "Frontend", "Backend", and "Sites". The user has 1 repository named "spring-boot-file-upload-download-rest-api" and 1 site named "OpencartV123".

A screenshot of a web browser displaying the GitHub developer settings. The URL is https://github.com/settings/developers. The page has a header with 'Search or jump to...'. Below it are tabs for 'All requests', 'Issues', 'Marketplace', and 'Explore'. The main content area is titled 'GitHub Apps' with a sub-section 'GitHub Apps'. It contains a list with three items: 'GitHub App' (selected), 'Octokit App', and 'Personal access token'. A note below says 'Want to build something that integrates well with GitHub? Tap into the GitHub API. You can also read more about building GitHub Apps in our developer documentation.' At the bottom right is a 'New GitHub App' button.

Search or jump to... Pull requests Issues Marketplace Explore

[Settings](#) | [Developer settings](#)

GitHub Apps
 OAuth Apps
 Personal access tokens

New personal access token

Personal access tokens function like ordinary GitHub access tokens. They can be used instead of a password for Git over HTTPS, or can be used for authentication to the API over Basic Authentication.

Name

Expires in ? The token will expire on Thu, Aug 11 2020

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

<input type="checkbox"/> repo	All actions on private repositories
<input type="checkbox"/> repository	Read & write repository
<input type="checkbox"/> repository:status	Read & write repository status
<input type="checkbox"/> user	Read & write user information
<input type="checkbox"/> user:email	Read & write user email

	<input type="checkbox"/> write-public-key	Write user public keys Read user public keys
<input checked="" type="checkbox"/> adminscope_host	<input type="checkbox"/> readscope_host	Full control of necessary hosts. Manage necessary hosts.
<input type="checkbox"/> readscope_host	<input type="checkbox"/> readscope_host	Read necessary hosts
<input type="checkbox"/> adminorg_host	<input type="checkbox"/> adminorg_host	Full control of organization hosts.
<input type="checkbox"/> git	<input type="checkbox"/> clone git	Create git.
<input type="checkbox"/> notifications	<input type="checkbox"/> cancel notifications	Cancel notifications.
<input type="checkbox"/> user	<input type="checkbox"/> create user	Update all user data.
	<input type="checkbox"/> read user	Read all user profile data.
	<input type="checkbox"/> read all user addresses	Read user email addresses (read only)
	<input type="checkbox"/> update user	Update user profile data.
<input checked="" type="checkbox"/> delete_repo	<input type="checkbox"/> Delete repositories	Delete repositories.
<input type="checkbox"/> read-and-write-repository	<input type="checkbox"/> Read and write repo discussions	Read and write repo discussions.
	<input type="checkbox"/> read repository discussions	Read repo discussions.
<input type="checkbox"/> adminenterprise	<input type="checkbox"/> manage_enterprise_contacts	Full control of enterprise.
	<input type="checkbox"/> manage_enterprise_billing_data	Manage enterprise contacts and income groups.
	<input type="checkbox"/> readenterprise	Read and write enterprise billing data.
	<input type="checkbox"/> readenterpriseprofiledata	Read enterprise profile data.
<input checked="" type="checkbox"/> project	<input type="checkbox"/> full-control-project	Full control of projects.
	<input type="checkbox"/> read-project	Read access of projects.
<input type="checkbox"/> adminorg_key	<input type="checkbox"/> read-public-user-GPG-key	Full control of public user GPG keys.
	<input type="checkbox"/> write-public-user-GPG-key	Write public user GPG keys.
	<input type="checkbox"/> read-public-user-GPG-key	Read public user GPG keys.

A screenshot of the GitHub developer settings page. The URL is https://github.com/settings/developer. The page shows three tabs: 'Client OAuth Apps' (selected), 'Service Hooks', and 'Personal access tokens'. Under 'Personal access tokens', there is a table with one row. The token is 'ghp_wzDwvPfCnC2QzQz0yHvWxLJLcUOOGIj'. It has three columns: 'Scopes' (repository, admin:repo_hook, user, write:repo_hook, write:repo_admin), 'Last used' (2023-01-12 10:10:00), and 'Delete'. Below the table, a note says 'Personal access tokens handle big, sensitive data like your password. You can't use most of them for GitHub API calls, or you'll need to authenticate with the GitHub App Authorization.' A 'Copy' button is visible next to the token.

The screenshot shows the Microsoft Graph API Explorer interface. The left sidebar has a tree view of available APIs: Home, SharePoint, OneDrive, and Power BI. The main area is titled "Authentication / Beta / Authentication / {id} / tokens / me". A "GET" method is selected, and the "Headers" section includes "Authorization: Bearer <token>" and "Content-Type: application/json". The "Type" dropdown is set to "Query Name". Below the headers, there's a note about the authentication header and a link to learn more about the token. The "Body" tab is active, showing the URL: "/beta/authentication/{id}/tokens/me". The "Results" tab shows a single row with the status code "200 OK" and the following JSON response:

```
{  
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIxMjM0NTY3ODkwIiwibmFtZSI6IjRyZWVzZWQgTGFuZCIsImV4cCI6MTUxNjUwOTk4MSwiYXVkIjpbImh0dHA6Ly9leGFtcGxlLmNvbS8iXX0.JDfHq...  
",  
    "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIxMjM0NTY3ODkwIiwibmFtZSI6IjRyZWVzZWQgTGFuZCIsImV4cCI6MTUxNjUwOTk4MSwiYXVkIjpbImh0dHA6Ly9leGFtcGxlLmNvbS8iXX0.JDfHq...  
",  
    "id_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIxMjM0NTY3ODkwIiwibmFtZSI6IjRyZWVzZWQgTGFuZCIsImV4cCI6MTUxNjUwOTk4MSwiYXVkIjpbImh0dHA6Ly9leGFtcGxlLmNvbS8iXX0.JDfHq...  
",  
    "token_type": "Bearer",  
    "expires_in": 3600,  
    "not-before-policy": 0,  
    "resource": "https://graph.microsoft.com",  
    "audience": "https://graph.microsoft.com",  
    "scope": "User.read User.readBasic User.readWrite User.readWriteBasic",  
    "ext_expires_in": 3600,  
    "ext_audience": "https://graph.microsoft.com",  
    "ext_scope": "User.read User.readBasic User.readWrite User.readWriteBasic",  
    "ext_not-before-policy": 0  
}
```

 SunWedge | [Discover](#) | [Search](#) | [Dashboard](#) | [Marketplace](#) | [Pricing](#) | [Maps](#) | [Our Initiatives](#) | [Partners](#) | [Blog](#) | [For Business](#)

Weather API

Please, sign up to use our fast and easy-to-work weather APIs. As a start to use OpenWeather products, we recommend our One Call API 3.0. For more functionality, please consider our products, which are included in [professional collections](#).

One Call API 3.0 NEW

[Get API key](#) [Get started](#)

Make one API call and receive all essential weather data in one response:

- Minute forecast for 1 hour
- Hourly forecast for all hours
- Daily forecast for 6 days
- Historical data for 10+ years back by timestamp
- Extended weather警报

Pay as you call

1,000 API calls per day for free
0.0012 GBP per API call over the daily limit

[Subscribe to One Call API](#)

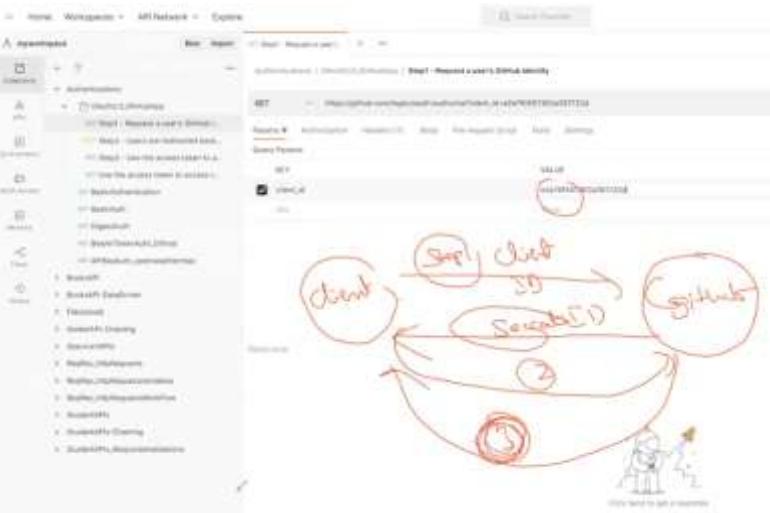
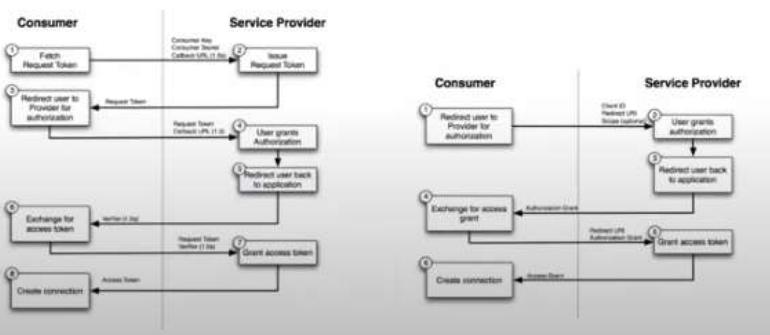
Read more about the API and subscription plans in the [FAQ](#)

This is a responsive subscription plan, which includes only One Call API.

OAuth 1.0 Vs OAuth 2.0

OAuth 1.0

OAuth 2.0



Setup OAuth2.0 – GitHub App

Login to GitHub → Go to profile → Settings → Developer Settings

Create a new GitHub App:

A screenshot of the GitHub Developer Settings page under 'GitHub Apps'. It shows a list with one item: 'Bilingual'. Below the list, it says 'No OAuth applications' and provides a link to 'Register a new application'.

Settings / Developer settings

Github Apps
OAuth Apps
Personal access tokens

No OAuth applications
OAuth applications are used to access the GitHub API. Read the docs to find out more.
Register a new application

Register a new OAuth application

Application name *

APITestingTraining

Something users will recognize and trust.

Homepage URL *

<http://localhost:3000>

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL *

<http://localhost:3000/callback>

Your application's callback URL. Read our OAuth documentation for more information.

Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow.

Read the Device Flow documentation for more information.

Something users will recognize and trust.

Homepage URL *

<http://localhost:3000>

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL *

<http://localhost:3000/callback>

Your application's callback URL. Read our OAuth documentation for more information.

Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow.

Read the Device Flow documentation for more information.

Register application

Cancel

APITestingTraining

pavanoltraining owns this application.

Transfer ownership

You can list your application in the GitHub Marketplace so that other users can discover it.

List this application in the Marketplace

0 users

Revoke all user tokens

Client ID

e2e76f497363a381720d

Generate a new client secret

Client secrets

You need a client secret to authenticate as the application to the API.

Application logo

Upload new logo

You can also drag and drop a picture from your computer.

Application logo

Upload new logo
You can also drag and drop a picture from your computer.

Application name *
APITestingTraining

Something users will recognize and trust.

Homepage URL *
http://localhost:3000

The full URL to your application homepage.

Application description
Application description is optional

This is displayed to all users of your application.

Authorization callback URL *
http://localhost:3000/callback

Your application's callback URL. Read our OAuth documentation for more information.

Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow.
Read the Device Flow documentation for more information.

Update application

After generating new client secret....

APITestingTraining

pavanoltraining owns this application.

Transfer ownership

You can list your application in the GitHub Marketplace so that other users can discover it.

List this application in the Marketplace

0 users

Revoke all user tokens

Client ID
e2e76f497363a381720d

Client secrets

Generate a new client secret

Make sure to copy your new client secret now. You won't be able to see it again.

You can list your application in the GitHub Marketplace so that other users can discover it.

List this application in the Marketplace

0 users

Revoke all user tokens

Client ID
e2e76f497363a381720d

Client secrets

Generate a new client secret

Make sure to copy your new client secret now. You won't be able to see it again.

Client secret added by pavanoltraining
Never used
You cannot delete the only client secret. Generate a new client secret first.

Client ID : **e2e76f497363a381720d**

Client secret: **de2e61ebbf38910d3924fd94fbdc22c87ffbb-a91**

Step 1: Request a user's GitHub identity

GET

https://github.com/login/oauth/authorize?client_id=e2e76f497363a381720d

GET https://github.com/login/oauth/authorize?client_id=e2e76f497363a381720d

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
<input checked="" type="checkbox"/> client_id	e2e76f497363a381720d

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

Service	Module	Port(s)	Actions
Apache		Start Stop Config Logs	
MySQL		Start Stop Config Logs	
FileZilla		Start Stop Config Logs	
Mercury		Start Stop Admin Config Logs	
Tomcat	12900 8080, 55246, 55247, 55248, 55249	Stop Start Admin Config Logs	

7:42:25 AM [main] XAMPP installation Directory: "c:\xampp\"
7:42:25 AM [main] Checking for prerequisites
7:42:25 AM [main] All prerequisites found.
7:42:25 AM [main] Initializing Modules
7:42:25 AM [Tomcat] Java is already running on port 8080!
7:42:25 AM [Tomcat] Is Tomcat already running?
7:42:25 AM [main] Starting Check-Timer
7:42:25 AM [main] Control Panel Ready

<https://localhost:8080/>

Authorize APITestingTraining

APITestingTraining by pavanoltraining wants to access your pavanoltraining account

Public data only Limited access to your public data

Cancel Authorize pavanoltraining

Authorizing will redirect to <http://localhost:8080>

Not owned or operated by GitHub Created less than a day ago Fewer than 10 GitHub users

Learn more about OAuth

localhost refused to connect.

This site can't be reached

localhost refused to connect.

Try:

- Checking the connection
- Checking the proxy and the Firewall

ERR_CONNECTION_REFUSED

Step 2 : Users are redirected back to your site by GitHub

Authentications / Sessions / Users are redirected back to your site by GitHub

POST https://github.com/login/oauth/access_token?client_id=42e10497363a387720&client_secret=d2e01aabb389103924d847fb2c29c67fbba9f4code=53d45293db8f8ctcc0

Params Authorization Headers (0) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
code	53d45293db8f8ctcc0	
client_id	42e10497363a387720	
client_secret	d2e01aabb389103924d847fb2c29c67fbba9f4	

Body Cookies (0) Headers (0) Test Results

Pretty Raw Preview Visuals Tree

1 access_token=gho_BlkRkQCSzaokYnFsd0V5J5kMgt6Y5yH02A9P1&scope=&token_type=bearer

Status: 200 OK Time: 403

API Requests

Home Workspaces API Requests

Step 1 - Request a user's identity

GET https://api.github.com/login/oauth/authorize?access_type=offline&client_id=42e10497363a387720

Params Authorization Headers (0) Body Pre-request Script Tests Settings

Query Params

Key	Value
access_type	offline
client_id	42e10497363a387720

Pretty Raw Preview Visuals Tree

GitHub Desktop

File Import User Management Help

Settings / Manage settings / API Testing

APITestingTraining

permissions come to this application Transfer ownership

You can give permission to the GitHub application to make other users see it. Learn more about this.

Turn Turn off user consent

Client ID 42e10497363a387720

Grant Access

Grant Access Select the permissions you want to give to this application. Learn more about this.

Application logs

localhost:8080/callback?code=a96e404afece659cd18

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Jul 26 07:45:25 IST 2022

There was an unexpected error (type=Not Found, status=404).

No message available

API Requests

Home Workspaces API Requests

Step 1 - Request a user's identity

GET https://api.github.com/login/oauth/authorize?access_type=offline&client_id=42e10497363a387720

Params Authorization Headers (0) Body Pre-request Script Tests Settings

Query Params

Key	Value
access_type	offline
client_id	42e10497363a387720

Pretty Raw Preview Visuals Tree

The screenshot shows the Postman interface for testing an API endpoint. The URL is https://github.com/login/oauth/access_token?client_id=de2e70497383a381720&client_secret=dc2e8fbafef38910d3824884fbec29c67fbab1&code=s2040s223d0fb3cc5. The 'Authorization' tab is selected, displaying the access token: gho_6URkQCszaoKyFu60V5J5kMgL6ySyN02A9R1&scope=&token_type=bearer.

The screenshot shows the Postman application interface. In the left sidebar, there is a tree view of collections and items. The main workspace shows a request to 'GET https://api.github.com/users/username'. The response tab displays the following JSON data:

```

{
  "type": "User",
  "private_github": false,
  "name": "The Fuzzy Doctor",
  "company": "Rockethosting, Komodo.js, https://github.com/lukechilds",
  "blog": "https://the-fuzzy-doctor.com",
  "bio": null,
  "location": null,
  "url": null
}

```

The screenshot shows the AWS Lambda function configuration interface. The left sidebar lists various AWS services and features under 'Workspaces'. The main area is titled 'Authentication' and shows the configuration for the 'Lambda@Edge' function 'lambda@edge'. The 'Authentication' tab is selected. The 'Type' dropdown is set to 'AWS Lambda'. The 'Lambda Function' dropdown is set to 'lambda@edge'. The 'Header Prefix' dropdown is set to 'none'. On the right, there is a 'Configure New Token' section with fields for 'Configuration Options' (set to 'Access token'), 'Token Name' (set to 'myToken'), and a 'Description' field. A small cartoon character icon is in the bottom right corner.

Setup OAuth2.0 – GitHub App

[Login to GitHub](#) → [Go to profile](#) → [Settings](#) → [Developer Settings](#)

Create a new GitHub App:

GitHub Apps

GitHub Apps
 OAuth Apps
 Personal access tokens

No OAuth applications

OAuth applications are used to access the GitHub API. Read the docs to find out more.

[Register a new application](#)

I

Authorization callback URL *

Your application's callback URL. Read our OAuth documentation for more information.

 Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow.

Read the Device Flow documentation for more information.

[Register application](#)[Cancel](#)

0 users

[Revoke all user tokens](#)**Client ID****Client secrets**[Generate a new client secret](#)

You need a client secret to authenticate as the application to the API.

Application logo[Upload new logo](#)

You can also drag and drop a picture from your computer.

I

Application name *

Something users will recognize and trust.

Homepage URL *

The full URL to your application homepage.

Application description

Application description is optional

You can also drag and drop a picture from your computer.

Application name *

Something users will recognize and trust.

Homepage URL *

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL *

Your application's callback URL. Read our OAuth documentation for more information.

 Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow.

Read the Device Flow documentation for more information.

[Update application](#)

You can also drag and drop a picture from your computer.

**GitHub Apps**

Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing the GitHub API. You can also read more about building GitHub Apps in the GitHub documentation.

© 2020 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Jobs](#) [Contact Support](#) [API](#) [Help](#) [About](#)

Day-8

1) Upload file

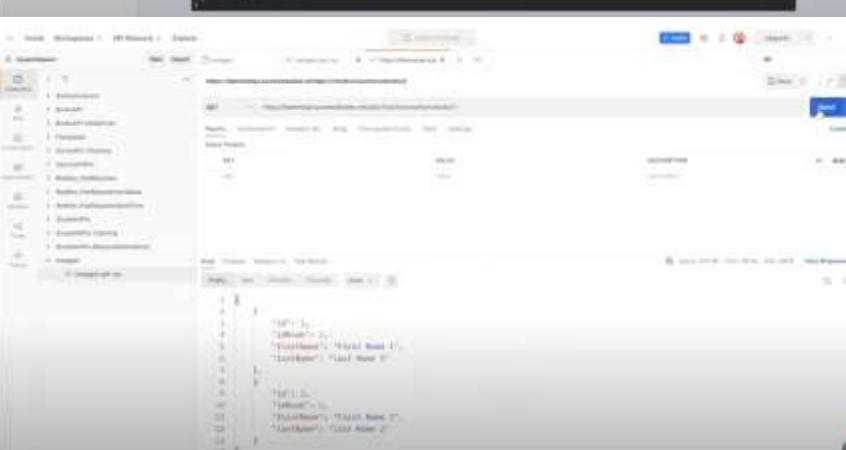
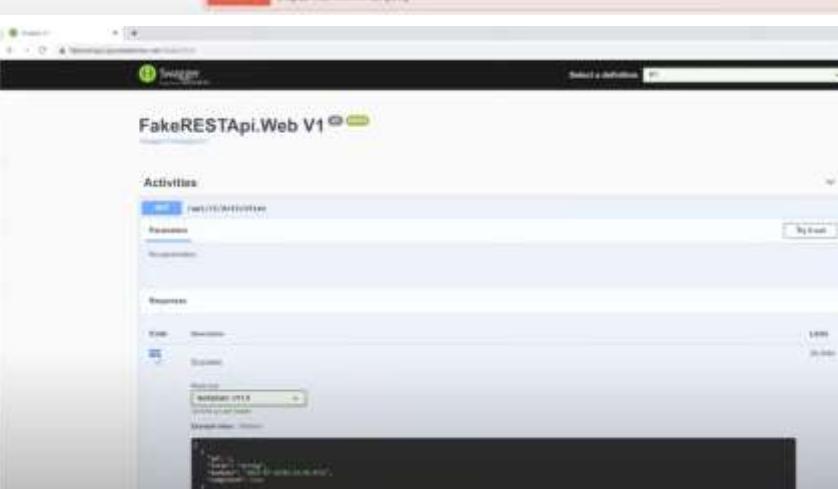
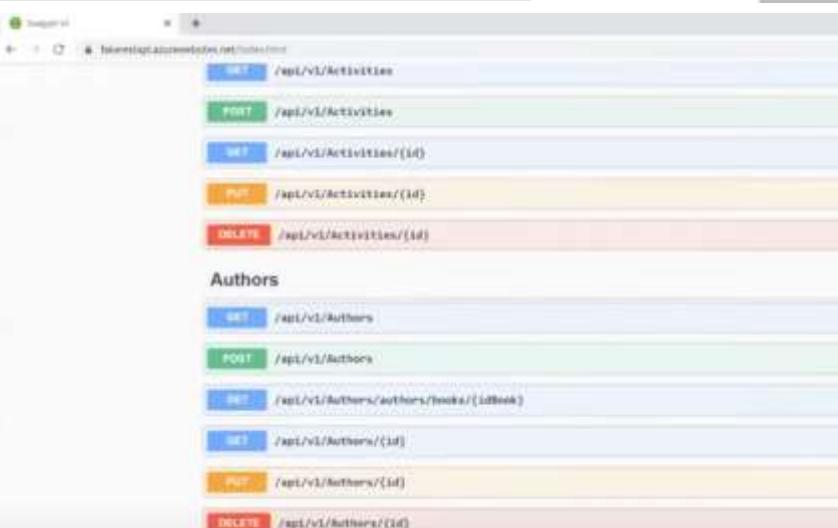
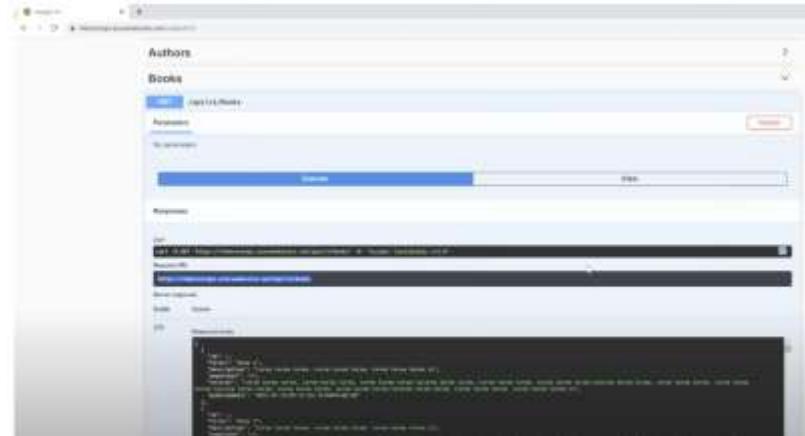
Post : http://localhost:8080/uploadFile

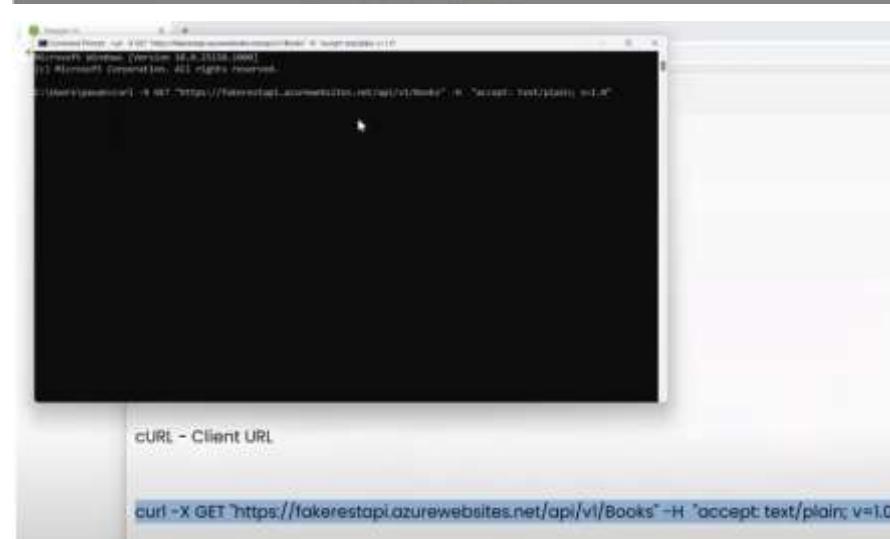
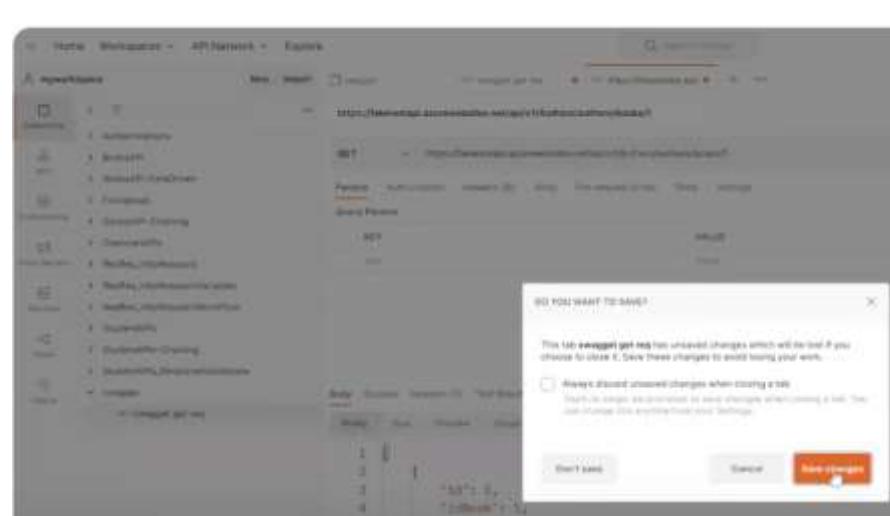
POST: http://localhost:8080/uploadMultipleFiles

Authentication types

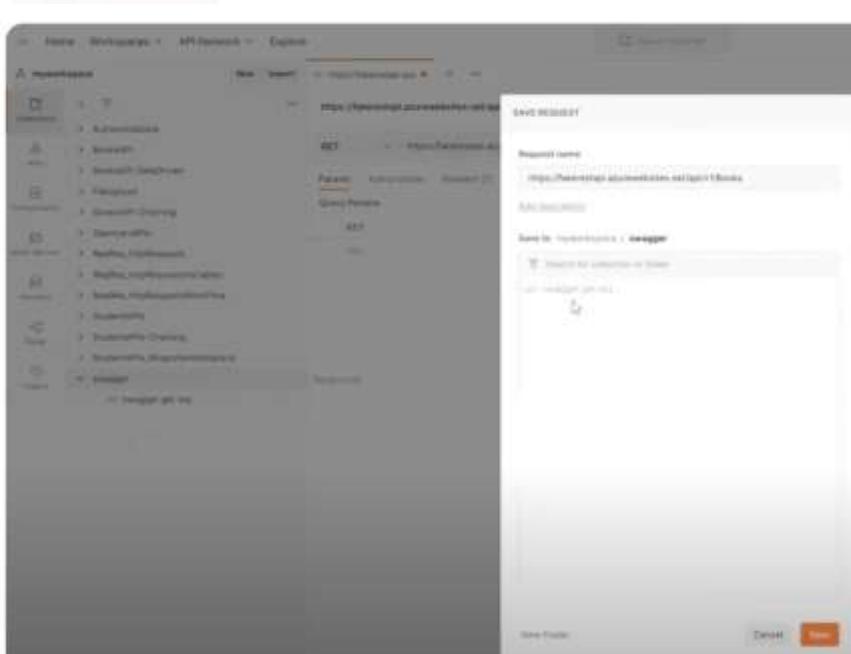
Swagger - interactive documentation

cURL





cURL - Client URL



The screenshot shows the Postman application interface. On the left, there's a sidebar with a tree view of collections and environments. The main area has tabs for 'Books' and 'Books - Response Body'. The 'Books' tab is active, showing a GET request to 'https://fakestoreapi.com/books'. The response body is displayed as a JSON array:

```
[{"id":1,"title":"Book 1","description":"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."}, {"id":2,"title":"Book 2","description":"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."}, {"id":3,"title":"Book 3","description":"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."}, {"id":4,"title":"Book 4","description":"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."}]
```

The screenshot shows the Postman application interface. At the top, there's a header bar with the title 'Mentored API Endpoints Collection'. Below the header, the collection structure is visible:

- Authors**
- Books**
- CoverPhotos**

Under the 'CoverPhotos' section, there are two requests:

- GET /api/v1/CoverPhotos**
- POST /api/v1/CoverPhotos**

Below the requests, there's a 'Parameters' section with a note 'No parameters' and a 'Try it out' button.

The 'Request body' field is set to 'application/json; charset=UTF-8'.

In the 'Body' section, under 'Preview (JSON)', the following JSON object is shown:

```
{ "name": "John", "age": 25, "city": "New York" }
```

At the bottom, there's a 'Responses' section with a table:

Code	Description	Last Run
200	Success	2023-01-12 10:30:00

Below the table, there's a dropdown menu with options: 'Next Step', 'Next Step: v1.0', 'Copy Response Body', and 'Response Headers'.

The screenshot shows the API Platform interface with the 'Import' feature open. The left sidebar displays a tree structure of entities: **Appraisals**, **Borrower**, **BorrowerDocument**, **File**, **FileContent**, **Photo**, **PhotoComment**, **PhotoDocument**, and **Project**. Below these are two generated files: `app.swagger.json` and `http://www.mysite.com/api/docs/index.html`.

The main area is titled 'Import' and includes tabs for 'File', 'Editor', 'UI', and 'Raw text'. The 'Raw text' tab is selected, showing the curl command:

```
curl -X POST "http://www.mysite.com/api/v1/uploadPhoto" -H "Accept: application/json" -H "Content-Type: multipart/form-data" -F "file=@/tmp/test.jpg" -F "comment=Test"
```

Home Workspaces API Network Explore

myworkspace

Collections +

- > Authentications
- > BooksAPI
- > BooksAPI-DataDriven
- > FileUpload
- > GorestAPI-Chaining
- > OpencartAPIs
- > ReqRes_httpRequests
- > ReqRes_httpRequestsVariables
- > ReqRes_httpRequestsWorkFlow
- > StudentAPIs
- > StudentAPIs-Chaining
- > StudentAPIs_ResponseValidations
- > swagger

Mock Servers

Monitors

D2D Flows

History

Search Postman

POST https://fakerestapi.azurewebsites.net/api/v1/CoverPhotos

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {"id":0,"idBook":0,"url":"string"}
```

Response

GET https://fakerestapi.azurewebsites.net/api/v1/Books

Home Workspaces API Network Explore

myworkspace

Collections +

- > Authentications
- > BooksAPI
 - HTTP Create Token
 - GET StatusOfBooks
 - GET ListOfBooks
 - GET SingleBook
 - POST SubmitOrder
 - GET Get All Orders
 - GET Get single Order
 - PATCH Update Order
 - DELETE Delete order
- > BooksAPI-DataDriven
- > FileUpload
- > GorestAPI-Chaining
- > OpencartAPIs
- > ReqRes_httpRequests
- > ReqRes_httpRequestsVariables
- > ReqRes_httpRequestsWorkFlow
- > StudentAPIs

Environments

Mock Servers

Monitors

D2D Flows

History

Search Postman

myworkspace

Collections +

- > Authentications
- > BooksAPI
 - HTTP Create Token
 - GET StatusOfBooks
 - GET ListOfBooks
 - GET SingleBook
 - POST SubmitOrder
 - GET Get All Orders
 - GET Get single Order
 - PATCH Update Order
 - DELETE Delete order
- > BooksAPI-DataDriven
- > FileUpload
- > GorestAPI-Chaining
- > OpencartAPIs
- > ReqRes_httpRequests
- > ReqRes_httpRequestsVariables
- > ReqRes_httpRequestsWorkFlow
- > StudentAPIs

Environments

Mock Servers

Monitors

D2D Flows

History

Search Postman

BooksAPI

Resource Tag CURRENT Language HTTP

BooksAPI

Books API provides the user functionality with 12 endpoints, covering the following:

POST Create Token

POST /CreateToken

Body (raw JSON)

```
{
  "ClientName": "Training",
  "ClientID": "Training@gmail.com"
}
```

GET StatusOfBooks

GET /StatusOfBooks

GET ListOfBooks

GET /ListOfBooks

Day-8

1) Upload file

Post : <http://localhost:8080/uploadFile>
 POST: <http://localhost:8080/uploadMultipleFiles>

Authentication types

[Swagger - interactive documentation](#)
[cURL - Client URL](#)

```
curl -X GET "https://fakerestapi.azurewebsites.net/api/v1/Books" -H "accept: text/plain; v=1.0"
```

Day-8

1) Upload file

Post : <http://localhost:8080/uploadFile>
 POST: <http://localhost:8080/uploadMultipleFiles>

Authentication types

[Swagger - interactive documentation](#)
[cURL - Client URL](#)

```
curl -X GET "https://fakerestapi.azurewebsites.net/api/v1/Books" -H "accept: text/plain; v=1.0"
```

<https://petstore.swagger.io/>
<https://httpbin.org/#/>

The screenshot shows the Swagger UI interface for the Petstore API. At the top, there's a header bar with a back arrow, forward arrow, and a search bar containing "petstore.swagger.io". Below the header, a dropdown menu titled "Schemes" is set to "HTTPS".

The main content area is titled "pet Everything about your Pets". It lists several API endpoints:

- POST /pet/{petId}/uploadImage** uploads an image
- POST /pet** Add a new pet to the store
- PUT /pet** Update an existing pet
- GET /pet/findByStatus** Finds Pets by status
- GET /pet/findByTags** Finds Pets by tags
- GET /pet/{petId}** Find pet by ID
- POST /pet/findByStatus** Returns all pets where status matches the value

Below the endpoints, there's a detailed view for the "body" parameter of the "POST /user/createUser" endpoint. The parameter is marked as required and is of type "object". It has a sub-object "(body)" with the following schema:

```
{  
    "id": 1010,  
    "username": "john",  
    "firstName": "xyz",  
    "lastName": "string",  
    "email": "abcaaa@gmail.com",  
    "password": "string",  
    "phone": "1234567",  
    "userStatus": 0  
}
```

At the bottom of the detailed view, there are buttons for "Cancel" and "Execute". A dropdown menu for "Parameter content type" is set to "application/json".

The screenshot shows the Postman application window. On the left, there's a sidebar with various icons for 'Responses', 'Collections', 'APIs', 'Workspaces', 'Environment', 'Mock Servers', 'Flows', and 'Documentation'. The main area has tabs for 'Responses', 'Collections', 'APIs', and 'Workspaces'. A central 'Create New' dialog box is open, listing several options: 'HTTP Request' (selected), 'WebSockets Request', 'gRPC Request - REST', 'Collection' (with a note about API versioning), 'Environment', 'Workspace', 'API Documentation', 'Mock Server', and 'Flows - REST'. At the bottom of the dialog, it says 'Postman allows you to share your APIs, collections, and environments published by the Postman community.' A 'Learn more' link is at the bottom right.

The screenshot shows the Microsoft Graph API Explorer interface. The left sidebar has a tree view under 'Workspaces' with items like 'Authentication', 'Scopes', 'OAuth2Client', 'Delegated Permissions', and 'API Permissions'. The main area is titled 'Authentication' and contains sections for 'Implicit' and 'Authorization code'. A note says: 'This authentication mechanism will be used for every request to this application. You can override this by specifying one or other mechanism.' Below it is a 'Type' dropdown set to 'Delegated'. At the bottom, a note says: 'This authentication needs user identity permission. Learn more about authentication »'.

The screenshot shows the AWS Lambda function configuration interface. The left sidebar lists several functions: 'HelloWorld' (selected), 'LambdaTest', 'LambdaTest2', 'LambdaTest3', 'LambdaTest4', 'LambdaTest5', 'LambdaTest6', 'LambdaTest7', 'LambdaTest8', 'LambdaTest9', 'LambdaTest10', and 'LambdaTest11'. The main panel has tabs for 'Basic', 'Code', 'Environment', 'Runtime Configuration', and 'Logs'. The 'Basic' tab is active, showing the function name 'HelloWorld', runtime 'Node.js 12.x', memory '128 MB', timeout '3 seconds', and role 'LambdaTestRole'. Below these are fields for 'Handler' (set to 'index.handler') and 'Description'. A large text area contains the Lambda function code:

```
const AWS = require('aws-lambda');
const https = require('https');

exports.handler = async (event) => {
  const response = {
    statusCode: 200,
    body: JSON.stringify({
      message: 'Hello from Lambda',
      input: event
    })
  };

  return response;
};
```

The screenshot shows the Postman application interface. The left sidebar displays a tree view of collections: 'Petitions' is expanded, showing sub-collections like 'Responses', 'Authentifications', 'Bookings', 'BookAPI-DataDriven', 'Felicities', 'GetUserAPI-Chaining', 'Deposit_WebAPI-CartMaze', and 'Petitions'. A right-click context menu is open over the 'Responses' item under 'Petitions', with options like 'Copy', 'Edit', 'Delete', 'Import', 'Export', 'Share', and 'Move'. The main workspace shows a 'Petitions' collection with a single 'Post' request highlighted. The request details are as follows:

- Method:** POST
- URL:** https://petstore.swagger.io/v2/user
- Params:** (empty)
- Authorization:** (empty)
- Headers (0):** (empty)
- Body:** (empty)
- Pre-request Script:** (empty)
- Tests:** (empty)
- Skills:** (empty)

The screenshot shows a POST request to the '/user' endpoint with the sub-operation 'Create user'. A note states: 'This can only be done by the logged in user.' Below it, the 'Parameters' section is shown with a table:

Name	Description
body	Created user object object (JSON) Example Value: <code>{ "username": "string", "password": "string", "firstName": "string", "lastName": "string", "email": "string", "address": "string", "zipCode": "string", "activation": 0 }</code>

The 'Parameter content type' dropdown is set to 'application/json'. The 'Responses' section shows a single entry with code 200 and the description 'Successful operation'.

At the bottom right, there is a 'Bilingual' button.

The screenshot shows the Microsoft Power BI Dataflows interface. On the left, there's a sidebar with sections for 'Dataflows' (containing 'PowerBI_2019_44110') and 'Recent'. The main area is titled 'PowerBI_2019_44110' and has tabs for 'Get data', 'Transform data', and 'Visualize'. The 'Get data' tab is selected, showing a connection to 'PowerBI' with the query 'PowerBI_2019_44110'. Below this, there's a table with columns 'Variable', 'Initial Value', and 'Current Value'. The table contains several variables: 'username' (initial value: prompt.username, current value: null), 'password' (initial value: prompt.password, current value: null), 'tenant' (initial value: prompt.tenant, current value: null), 'resourceURI' (initial value: prompt.resourceURI, current value: null), and 'clientID' (initial value: prompt.clientID, current value: null). A note at the bottom says 'This connection is specific to this connection and its requests. Learn more about connection variables.' At the bottom right, there's a note: 'This connection needs to make changes and extract sensitive data. This connection uses a password that needs to be stored securely. Learn more'.

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named "myworkspace". Inside this collection, there are several items: "ReqRes_httpRequestsWorkFlow", "Authentications", "BooksAPI", "BooksAPI-DataDriven", "FileUpload", "GorestAPI-Chaining", "OpenCart_RestAPI_CartModule", "PetStore", and "PetStore_JSON_UserModel". The "PetStore_JSON_UserModel" item is expanded, showing sub-items: "Create User", "Get User By Name" (which is currently selected), "Update User By Name", and "Delete User By Name". Below these are other collections like "PetStore_XML_PetModel" and "ReqRes_httpRequests". At the top, there are tabs for "New", "Import", and "POST Create User". The main content area shows the "PetStore_JSON_UserModel / Get User By Name" item with a "GET" method, URL "https://petstore.swagger.io/v2/user/{username}", and a "Tests" tab containing the following code:

```
1 pm.test("Check status code", function () {  
2     pm.response.to.have.status(200);  
3 });
```

The screenshot shows the Postman application interface. On the left, the sidebar displays the 'myworkspace' collection with several sub-folders like 'Authentications', 'BooksAPI', etc., and a list of APIs including 'PetStore', 'PetStore_JSON_UserIdentity', and 'PetStore_XML_PetModel'. The 'PetStore_JSON_UserIdentity' folder is expanded, showing four endpoints: 'Create User', 'Get User By Name' (which is currently selected), 'Update User By Name', and 'Delete User By Name'. The 'Get User By Name' endpoint is detailed in the main panel, showing a GET request to 'https://petstore.swagger.io/v2/user/{id}'. The 'Tests' tab contains a simple PM test script:

```
1 pm.test("Check status code", function () {  
2     pm.response.to.have.status(200);  
3 });
```

The 'Body' tab shows a JSON response structure:

```
1 {  
2     "id": 95,  
3     "username": "johnpjn1z3djcck",  
4     "firstName": "johnpjn1z3djcck",  
5     "lastName": "johnpjn1z3djcck",  
6     "email": "johnpjn1z3djcck@gmail.com",  
7     "password": "johnpjn1z3djcck",  
8     "phone": "1111111111",  
9     "userStatus": 0  
10 }
```

myworkspace

- Collections
 - + ReqRes_httpRequestsWorkFlow
 - > Authentications
 - > BooksAPI
 - > BooksAPI-DataDriven
 - > FileUpload
 - > GorestAPI-Chaining
 - > Opencart_RestAPI_CartModule
 - > PetStore
 - > PetStore_JSON_UserIdModel
 - POST Create User
 - GET Get User By Name
 - PUT Update User By Name
 - DELETE Delete User By Name
 - > PetStore_XML_PetModel
 - > ReqRes_httpRequests
 - > ReqRes_httpRequestsVariables
 - > StudentAPIs
 - > StudentAPIs-Chaining
 - > StudentAPIs_ResponseValidations
 - > swagger
- APIs
- Environments
- Mock Servers
- Monitors
- Flows
- History

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```
1 var randomstr = "john"+Math.random().toString(36).substring(2);
2
3 //Updating email & phone
4 pm.collectionVariables.set("email", randomstr+"@gmail.com");
5 pm.collectionVariables.set("phone", "2222222222");
6
```

Response

myworkspace

- Collections
 - + ReqRes_httpRequestsWorkFlow
 - > Authentications
 - > BooksAPI
 - > BooksAPI-DataDriven
 - > FileUpload
 - > GorestAPI-Chaining
 - > Opencart_RestAPI_CartModule
 - > PetStore
 - > PetStore_JSON_UserIdModel
 - POST Create User
 - GET Get User By Name
 - PUT Update User By Name
 - DELETE Delete User By Name
 - > PetStore_XML_PetModel
 - > ReqRes_httpRequests
 - > ReqRes_httpRequestsVariables
 - > StudentAPIs
 - > StudentAPIs-Chaining
 - > StudentAPIs_ResponseValidations
 - > swagger
- APIs
- Environments
- Mock Servers
- Monitors
- Flows
- History

DELETE https://petstore.swagger.io/v2/user/{(firstName)}

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```
1 pm.test("Check status code", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.environment.unset("id");
6 pm.collectionVariables.unset("username");
7 pm.collectionVariables.unset("firstName");
8 pm.collectionVariables.unset("lastName");
9 pm.collectionVariables.unset("email");
10 pm.collectionVariables.unset("password");
11 pm.collectionVariables.unset("phone");
12 pm.collectionVariables.unset("userStatus");
13
```

Response

myworkspace

- Collections
 - + ReqRes_httpRequestsWorkFlow
 - > Authentications
 - > BooksAPI
 - > BooksAPI-DataDriven
 - > FileUpload
 - > GorestAPI-Chaining
 - > Opencart_RestAPI_CartModule
 - > PetStore
 - > PetStore_JSON_UserIdModel
 - POST Create User https://petstore.swagger.io/v2/user / Create User
 - Pass Check status code.
 - Pass Check status code.
 - GET Get User By Name https://petstore.swagger.io/v2/user/{(firstName)} / Get User By Name
 - Pass Check status code.
 - PUT Update User By Name https://petstore.swagger.io/v2/user/{(firstName)} / Update User By Name
 - Pass Check status code.
 - DELETE Delete User By Name https://petstore.swagger.io/v2/user/{(firstName)} / Delete User By Name
 - Pass Check status code.
 - > PetStore_XML_PetModel
 - > ReqRes_httpRequests
 - > ReqRes_httpRequestsVariables
 - > StudentAPIs
 - > StudentAPIs-Chaining
 - > StudentAPIs_ResponseValidations
 - > swagger
- APIs
- Environments
- Mock Servers
- Monitors
- Flows
- History

All Tests Passed (5) Failed (0) Skipped (0)

Iteration 1

POST Create User https://petstore.swagger.io/v2/user / Create User

Pass Check status code.

Pass Check status code.

GET Get User By Name https://petstore.swagger.io/v2/user/{(firstName)} / Get User By Name

Pass Check status code.

PUT Update User By Name https://petstore.swagger.io/v2/user/{(firstName)} / Update User By Name

Pass Check status code.

DELETE Delete User By Name https://petstore.swagger.io/v2/user/{(firstName)} / Delete User By Name

Pass Check status code.

The screenshot shows the Postman application interface. The left sidebar displays the 'myworkspace' section with a tree view of collections: 'Postman_2024' (selected), 'Authentications', 'BrewAPI', 'BrewerAPI_Database', 'Brewpost', 'GeneralAPI-Chatting', 'GeneralAPI_Cartologue', and 'Postman'. The main workspace shows the 'Postman_2024' collection expanded, revealing its sub-endpoints: 'Get User By Name', 'Update User By Name', and 'Delete User By Name'. Each endpoint has a status indicator: 'Pass' for 'Get User By Name' and 'Update User By Name', and 'Pending' for 'Delete User By Name'. Below the endpoints, there are two sections: 'Pre-request Script' and 'Tests'. The 'Pre-request Script' section contains two code snippets: 'Check authz code' and 'Check status code'. The 'Tests' section also contains two code snippets: 'Check authz code' and 'Check status code'. The top navigation bar includes tabs for 'Home', 'Workspaces', 'API Network', and 'Explore', along with a 'Logout' button.

The screenshot shows the Postman application interface. On the left, there's a sidebar with a tree view of collections: 'PetStore JSON User Model' is selected. The main area displays the 'PetStore_JSON_UserModel' collection with its sub-items: 'Get User', 'Get User By Name', 'Update User By Name', and 'Delete User By Name'. Below these items, there's a 'Body' section containing a JSON object:

```
{ "id": 1, "name": "Leanne Graham", "username": "Bret", "email": "Sincere@april.biz", "address": { "street": "Kulas Light", "suite": "Apt. 556", "city": "Gwenborough", "zipcode": "92998-3872", "geo": { "lat": "43.7654", "lon": "-71.2854" } }, "phone": "1-770-736-8290", "website": "hildegard.org", "company": { "name": "Romaguera-Crona", "catchPhrase": "Multi-layered client-server interface", "bs": "User-centricruzgaran.com" } }
```

A screenshot of the Microsoft Power BI service interface. The top navigation bar includes 'Home', 'Workbooks', 'Dashboards', 'Reports', and 'My workspace'. Below this is a search bar and a 'New' button. The main area shows a dashboard titled 'Dashboard - Sales Performance' with a subtitle 'Last 30 days'. The dashboard features several visualizations: a large blue bar chart at the top, followed by a smaller bar chart, a line chart, and a table. On the left side, there is a navigation pane with a tree view of 'My content' and a 'Recently used' section. The bottom of the screen shows a dark footer with the Microsoft logo.

A screenshot of the Microsoft Power BI service interface. The top navigation bar shows 'Home', 'Workspaces', 'My workspace', and 'Logout'. Below the navigation is a search bar and a 'New' button. The main area is titled 'Public collection' with a sub-section 'Untitled_2020_yourname'. On the left, there's a 'Content' sidebar with 'Dashboards' (selected), 'Reports', and 'Visualizations'. The main workspace contains a single dashboard card with a blue circular icon and some text. To the right of the card is a 'Metrics' section with four items: 'Revenue by department', 'Profit', 'Total employees', and 'Revenue'. At the bottom, there's a 'Shareable link' section with a URL and a 'Copy' button, and a note about sharing the dashboard.

Day-9

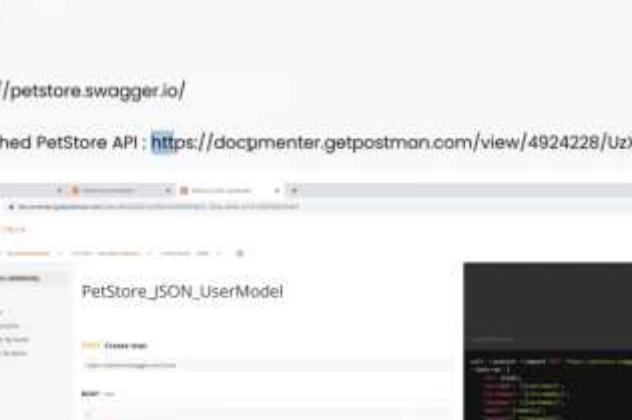
Petstore -

Json

XML

<https://petstore.swagger.io/>

published PetStore API : <https://documenter.getpostman.com/view/4924228/UzXNVdJS>

The screenshot shows the Postman application interface. On the left, there's a sidebar with a 'Petstore API' icon and a 'Petstore JSON User Model' collection. The main area displays a single endpoint: 'POST Create user'. Below the endpoint, there are several tabs: 'Body', 'Headers', 'Query String', 'Form Data', 'File', and 'Raw'. Each tab has its own input fields. To the right of the main interface, there's a dark panel showing a tree-like structure of other API endpoints under the 'Petstore API' collection.

A screenshot of the Postman application interface. The main title bar says "Postman" and the address bar shows "https://petstore.swagger.io/v2/pet/findByStatus?status=available". On the left, there's a sidebar with "Collections" and "Environments" sections. The main area has a title "PetStore_JSON_UserModel" and a sub-section "Create User". Below it, there's a "POST /user" request with a "User Model" body parameter. A context menu is open over this request, with options like "Run in...", "Postman for Web", and "Postman for Windows".

A screenshot of the Postman application interface. The left sidebar shows a collection named "PetStore_JSON_UserModel" containing several items: "Create user", "Get user", "Update user", "Delete user", and "Get users". The main workspace displays the "Create user" endpoint with a "Create user" button. A context menu is open over the "Create user" button, with options "Run in...", "Postman for Web", "Postman for Windows", and "Close this app" visible.

Day-9

Petstore -
Json
XML

<https://petstore.swagger.io/>

Swagger - Exploring API request interactive document

create documentation
publish it

published PetStore API : <https://documenter.getpostman.com/view/4924228/UzXNVdJS>

The screenshot shows the Postman application interface. On the left, there's a sidebar with various collections like 'Petstore XML PetModel', 'Petstore XML Pet', 'Petstore XML User', etc. The main area is titled 'PetStore_XML_PetModel' and contains a 'Petstore XML PetModel' collection. It includes sections for 'Add new Pet', 'Petstore XML Pet Model', 'Request Headers', 'Request Body', and 'Responses'. A preview window on the right shows a JSON response for a pet creation request.

Json
XML

<https://petstore.swagger.io/>

Swagger - Exploring API request
interactive document

create documentation
publish it

published PetStore API's :

<https://documenter.getpostman.com/view/4924228/UzXNVdJS>
<https://documenter.getpostman.com/view/4924228/UzXNVdNo>

This screenshot is identical to the one above, showing the Postman interface with the 'PetStore_XML_PetModel' collection selected. The 'Add new Pet' section is highlighted.

The screenshot shows the OpenCart website. At the top, there's a navigation bar with links like 'Home', 'About', 'Products', 'Blog', 'Contact', and 'Logout'. Below the navigation, there are two main sections: 'Store Front' and 'Administration'. The 'Store Front' section shows a product listing for a green dog bed. The 'Administration' section shows a dashboard with various metrics and a map. A blue button at the bottom center says 'View Admin Area'.

Opencart Build Deployment/Installation

1) Opencart download location:

<https://www.opencart.com/index.php?route=cms/download>

Click on [previous releases](#)

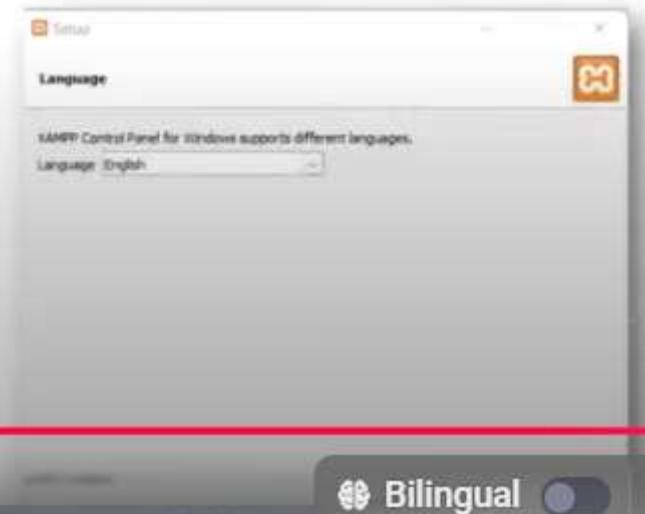
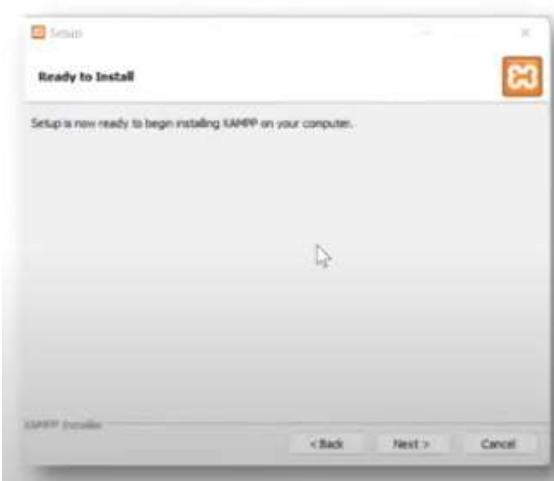
Recommended version: **opencart-3.0.3.8**

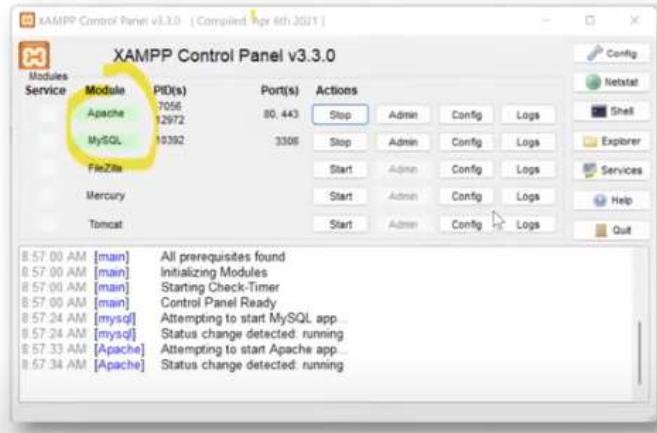
2) XAMPP for apache, mysql, and php installation.

Download Link: <https://www.apachefriends.org/download.html>

Recommended version: **7.4.29/PHP 7.4.29**

Step by step XAMPP Installation





5) Rename files

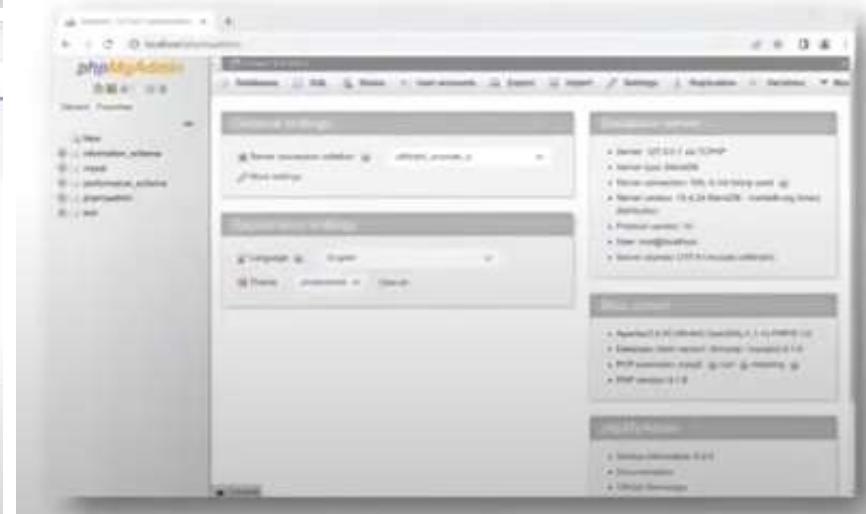
Go to C:\xampp\htdocs\opencart\upload
Rename file 'config-dist.php' to 'config.php'

Go to C:\xampp\htdocs\opencart\upload\admin
Rename file 'config-dist.php' to 'config.php'

6) Connect to the database and create DB.

DB Access URL: <http://localhost/phpmyadmin/>

Create new database 'openshop' (Refer the screenshots below to create new Database in MySQL)

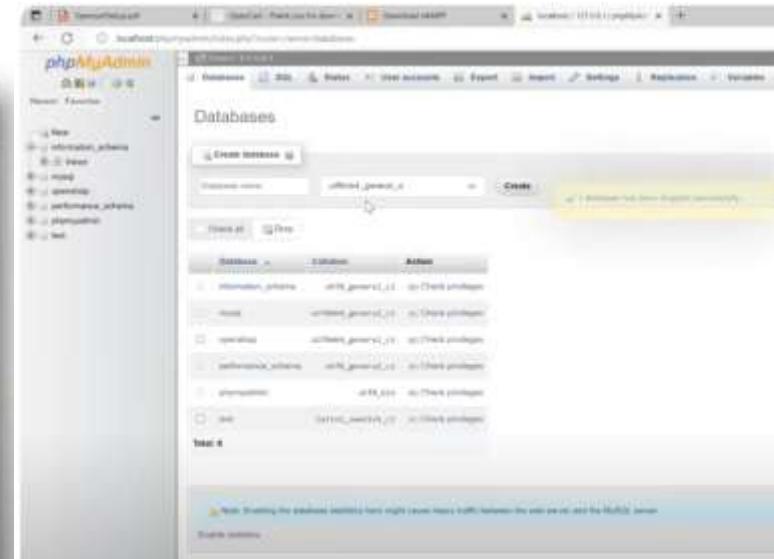
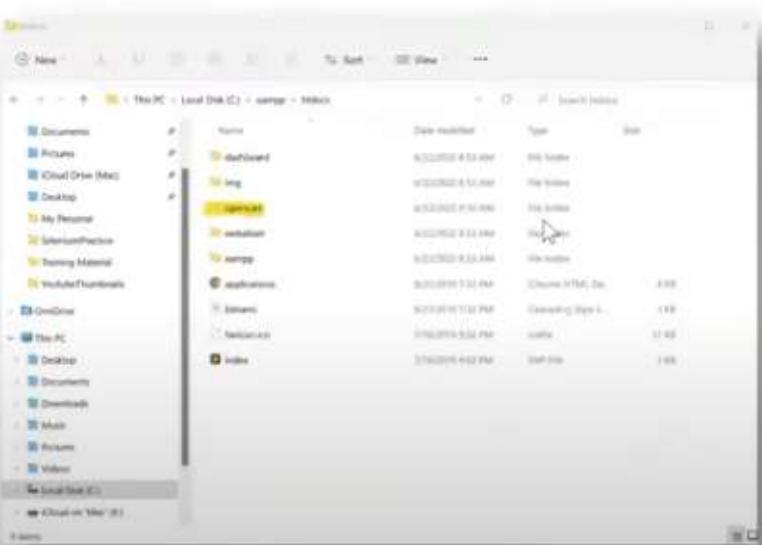


Step by step OpenCart Installation

3) Copy opencart folder(Step1) in to below location. C:\xampp\htdocs



4) After copying rename the folder " opencart "



Click on CONTINUE Button....

The screenshot shows the phpMyAdmin interface. In the General settings section, the character set is set to utf8mb4_unicode_ci. Under Database server, it shows a MySQL 8.0.28 connection. Application settings include language (English) and theme (phmadmin). The Webserver section lists Apache 2.4.53 (Win64), MySQL 8.1.6, PHP 8.1.6, and MariaDB 10.4.29. The footer indicates Version information: 5.3.0.

The screenshot shows the OpenCart license agreement page. It displays the GNU General Public License version 2, dated June 1991. It includes a note about the OpenCart license agreement, the GNU General Public License, and a preambles section. A yellow arrow points to the "CONTINUE" button at the bottom right.

The screenshot shows the phpMyAdmin databases list. It lists several databases: information_schema, mysql, performance_schema, and test. Each database has a status column indicating they are checked.

The screenshot shows the OpenCart pre-requisites configuration page. It has three main sections: 1. PHP configuration requirements (with a table showing various PHP extensions and their status), 2. PHP extensions (listing extensions like curl, gd, mbstring, etc., with checkboxes for configuration), and 3. File permissions (listing files and their required permissions).

7) Open the site <http://localhost/opencart/>
Click on upload

The screenshot shows a browser window displaying the OpenCart index page. The title is "Index of /opencart". The table lists files and folders: CHANGELOG.md, CHANGELOG_AUTO.md, INSTALL.md, UPGRADE.md, build.xml, docs/, license.txt, and upload/. The Apache server information at the bottom indicates Apache/2.4.53 (Win64) OpenSSL/1.1.1 In PHP/8.1.6 Server at localhost Port 80.

Navigate to <http://localhost/opencart/upload/install/index.php>

Click on CONTINUE Button....

If you are getting above error **GD is off** then you have to do below steps.
Open XAMPP Control panel, click on config button for Apache then open PHP(**php.ini**) file

Add new entry in the **php.ini** file **extension=gd**

The screenshot shows the XAMPP Control Panel. The Apache service is green and labeled "Apache". Other services like MySQL, FileZilla, and ProFTPD are also listed with their status.

Then stop and start Apache server. Then you will see below screen.

Configuring GD library

After above error **GD is not installed** in XAMPP Control panel, click on **Apache** in the php.ini file.

Then stop and start Apache server.

In the php.ini file:

```
; PHP's initialization file, generally called php.ini, is responsible for
; configuring many of the aspects of PHP's behavior.

; PHP attempts to find and load this configuration from a number of locations.
; The following is a summary of its search order:
; 1. SAPI module specific location.
; 2. The PHPRC environment variable. (As of PHP 5.2.0)
; 3. A number of predefined registry keys on Windows (As of PHP 5.2.0)
; 4. Current working directory (except CLI)
; 5. The web server's directory (for SAPI modules), or directory of PHP
; (otherwise in Windows):
```

Configuring GD library

After above error **GD is not installed** in XAMPP Control panel, click on **Apache** in the php.ini file.

Then stop and start Apache server.

In the php.ini file:

```
; deprecate in a future PHP major release, please
; move to the new ('extension=extname' syntax).

; Notes for Windows environments:
;
; - Many DLL files are located in the extensions/ (PHP 4) or ext/ (PHP 5+)
; extension folders as well as the separate PECL DLL download (PHP 5+).
; Be sure to appropriately set the extension_dir directive.

extension=bz2
extension=curl
extension=fifi
extension=ftp
extension=fileinfo
extension=gd2
extension=gettext
```

Check your server is set up correctly.

1. Please configure your PHP settings to match requirements listed below.

PHP Settings	Current Settings	Required Settings	Status
PHP Version	5.1.6	5.0+	✓
Register Globals	Off	Off	✓
Magic Quotes GPC	Off	Off	✓
File Uploads	On	On	✓
Session Auto Start	Off	Off	✓

2. Please make sure the PHP extensions listed below are installed.

Extension Settings	Current Settings	Required Settings	Status
Database	Off	On	✓
GD	On	On	✓
CURL	On	On	✓
OpenSSL	On	On	✓
ZLIB	On	On	✓
ZIP	On	On	✓

3. Please make sure you have set the correct permissions on the files list below.

Files	Status
C:\xampp\php\extras\phpgdconfig.php	Writable
C:\xampp\php\docs\php\phpadminconfig.php	Writable

Click on continue....

8) Provide Database connection details

Configuration

Language ▾

Enter your database and administration details

1. Please enter your database connection details.

DB Driver * Hostname

MySQL

localhost

* Username

root

Password

Information about setting up a database on different platforms:

- + CPanel DB Setup
- + Plesk DB Setup

* Database

openCart

Prefix * Port

oc_ 3306

2. Please enter a username and password for the administration.

* Username * Password

admin

admin

* E-Mail

admin@gmail.com

123456

/ 1:23:14

Bilingual



1. PLEASE ENTER YOUR DATABASE CONNECTION DETAILS.

DB Driver * Hostname

MySQL

localhost

* Username

root

Password

Information about setting up a database on different platforms:

- + CPanel DB Setup
- + Plesk DB Setup

* Database

openCart

Prefix * Port

oc_ 3306

2. Please enter a username and password for the administration.

* Username * Password

admin

admin

* E-Mail

admin@gmail.com

Back

Continue

James
* E-Mail
admin@gmail.com

admin

Back

Continue

Installation complete

Don't forget to delete your installation directory!

Ready to start selling!

9) Go to location C:\xampp\htdocs\opencart\upload

Delete **install** folder

(Mac)

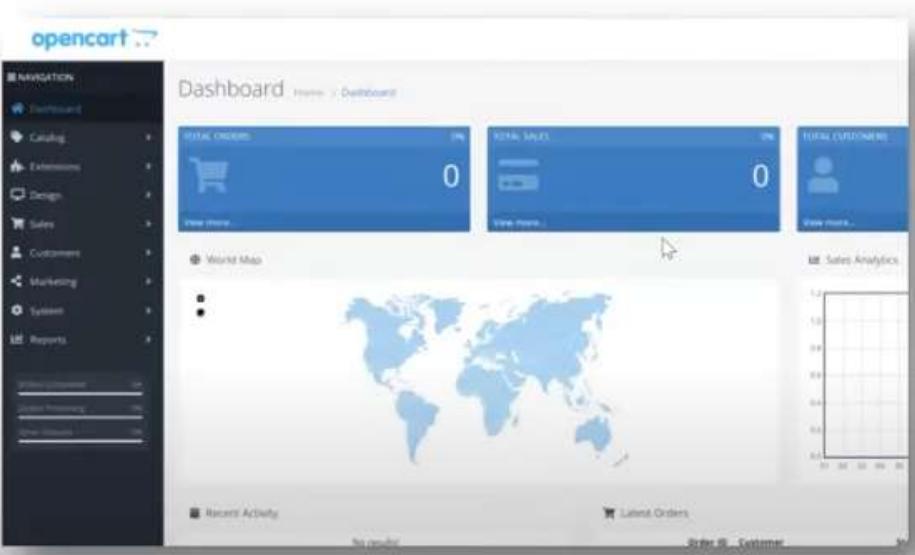
	Name	Date modified
	admin	6/22/2022 9:15
	catalog	6/22/2022 9:10
	extension	6/22/2022 9:10
	image	6/22/2022 9:10
	install	6/22/2022 9:10
	system	6/22/2022 9:10
	config.php	6/22/2022 9:54
	error.html	5/24/2022 2:40
	index.php	5/24/2022 2:40
	php.ini	5/24/2022 2:40
	robots.txt	5/24/2022 2:40

10) Go to Application URL

Frontend application URL: <http://localhost/opencart/upload/>



Backend Application url: <http://localhost/opencart/upload/admin/>



Windows 11

opencart

Please enter your login details.

Username:

Password:

Forgot Password?

OpenCart © 2008-2022 All Rights Reserved.

Windows 11

LIVEOPENCART

OpenCart Extensions Information & Articles Contact Us

OpenCart Extensions

We developed several useful OpenCart modules/extensions having wide functionality and great compatibility for using with the default OpenCart themes, but with custom themes too. Our solutions are mostly focused on advancing or extending the OpenCart functions related to premium extensions. Most of them are available for OpenCart 2 and OpenCart 3.

Choosing our extensions you are also choosing professional support, template and application service, as well as the full compatibility of the module our modules are comparable to each other, we provide adaptation to commercial themes.

All our extensions are also available on the official OpenCart marketplace <https://www.opencart.com>

PRODUCT OPTION IMAGE ULTIMATE	RELATED OPTIONS	RELATED OPTIONS PRO	PARENT- CHILD OPTIONS
MORE DETAILS	MORE DETAILS	MORE DETAILS	MORE DETAILS
LIVE PRICE	LIVE PRICE PRO	PRODUCT OPTION IMAGE PRO	IMPROVED OPTIONS
MORE DETAILS	MORE DETAILS	MORE DETAILS	MORE DETAILS

Windows 11

opencart

APIs

General

API Address:

API Key:

Version:

Status: Enabled

Information: Last API call at 2022-08-10 10:20:20

XML

<https://petstore.swagger.io/>

Swagger - Exploring API request
interactive document

create documentation
publish it

published PetStore API's :

<https://documenter.getpostman.com/view/4924228/UzXNVdJS>
<https://documenter.getpostman.com/view/4924228/UzXNVdNo>

E-Commerce application

opencart

Opencart API Documentation:

<https://docs.opencart.com/en-gb/system/users/api/>

The screenshot shows a web browser displaying the Opencart API documentation. The URL in the address bar is <http://docs.opencart.com/en-gb/system/users/api/>. The page title is "Opencart API". The main content area contains a note about choosing the appropriate controller and a "NOTES:" section with several bullet points. Below the notes is a code editor window showing a PHP script. The script starts with a class definition for "UserAPI", then defines a constructor that initializes a session and sets a timeout. It includes a method "getCustomer" which makes a cURL request to "index.php?route=user/customer&token={token}" and returns the response. A note at the bottom says "If everything was done right, you'll get your response with api_token for your session. Check this admin API page with API token just now." At the bottom of the page, there is a "Login" button and a "Logout" link.

```
class UserAPI {  
    public $session;  
    public $timeout = 10;  
  
    public function __construct() {  
        $this->session = new Session();  
        $this->session->start();  
        $this->session->timeout($this->timeout);  
    }  
  
    public function getCustomer($token) {  
        $ch = curl_init();  
        curl_setopt($ch, CURLOPT_URL, "index.php?route=user/customer&token={$token}");  
        curl_setopt($ch, CURLOPT_TIMEOUT, $this->timeout);  
        curl_exec($ch);  
        curl_close($ch);  
        return $this->session->response;  
    }  
}
```

XML

<https://petstore.swagger.io/>

Swagger - Exploring API request
interactive document

create documentation
publish it

published PetStore API's :

<https://documenter.getpostman.com/view/4924228/UzXNVdJS>

<https://documenter.getpostman.com/view/4924228/UzXNVdNo>

E-Commerce application

I

Install opencart app

API user

Opencart API Documentation: <https://docs.opencart.com/en-gb/system/users/api/>

Day-10

Opencart Application API

Opencart Build Deployment/Installation

1) Opencart download location:

<https://www.opencart.com/index.php?route=cms/download>

Click on [previous releases](#)

Recommended version: **opencart-3.0.3.8**

2) XAMPP for apache, mysql, and php installation.

Download Link: <https://www.apachefriends.org/download.html>

Recommended version: **7.4.29/PHP 7.4.29**



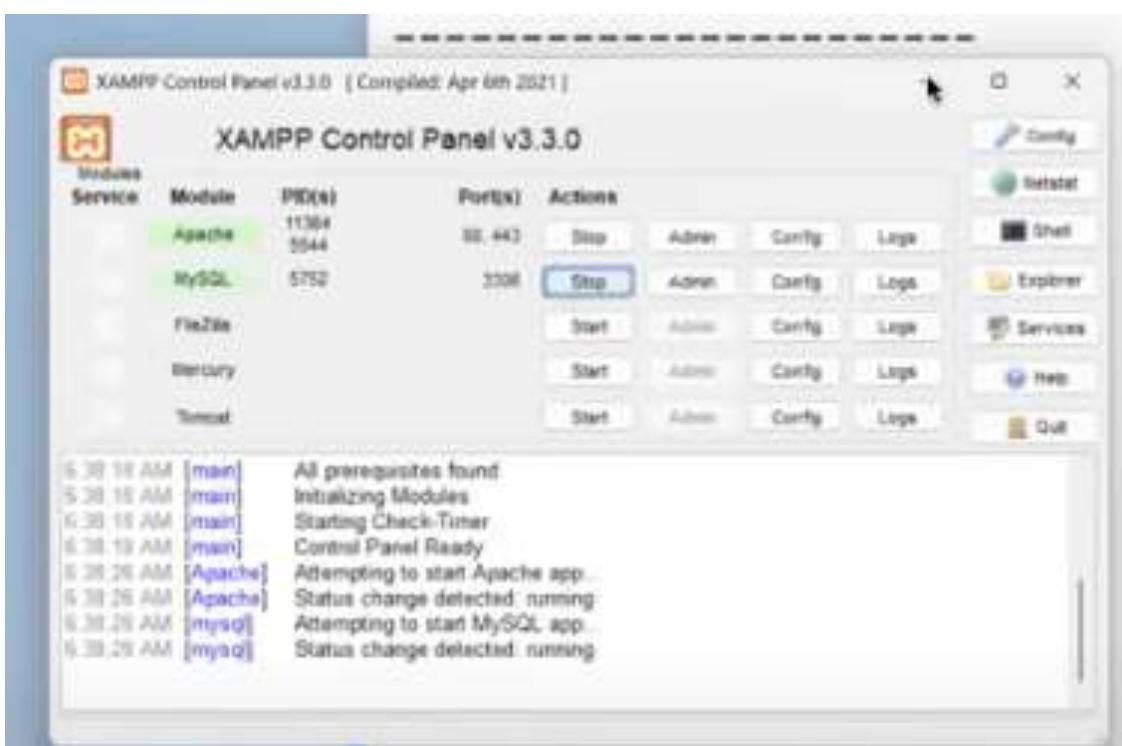
2) XAMPP for apache, mysql, and php installation.

Download Link: <https://www.apachefriends.org/download.html>

Recommended version: 7.4.29 / PHP 7.4.29

XAMPP for Windows 7.4.29, 8.0.19 & 8.1.6

Version	What's Included?	Checksum	Size
7.4.29 / PHP 7.4.29	What's Included?	md5 sha1	Download (64 bit) 159 Mb
8.0.19 / PHP 8.0.19	What's Included?	md5 sha1	Download (64 bit) 161 Mb
8.1.6 / PHP 8.1.6	What's Included?	md5 sha1	Download (64 bit) 164 Mb



A screenshot of the Postman application interface. The left sidebar shows 'OPENCART_RESTAPI_CARTMODULE' and 'Introduction'. Under 'Introduction', there are four items: 'Create SessionToken', 'Add product to cart', 'Cart content', and 'Edit product quantity in cart'. The main area shows a POST request for 'Add product to cart' with the URL 'http://192.168.0.107/opencart/upload/index.php?route=api/cart/add'. The 'BODY' tab shows 'form-data' with 'product_id' set to '(\$product_id)' and 'quantity' set to '(\$quantity)'. To the right, there is a preview of the response and a 'Cart content' tab.

```
curl -X POST "http://opencart-test.opencart-test.com/index.php?route=api/currency"
Content-Type: application/json
{
    "username": "admin",
    "key": "12345678901234567890123456789012"
}

{
    "name": "USD"
}
```

Currency

apiCurrency

DESCRIPTION: API endpoint for managing currencies.

HTTP METHODS: GET, POST, PUT, DELETE

EXAMPLE

```
curl -X POST "http://opencart-test.opencart-test.com/index.php?route=api/currency"
Content-Type: application/json
{
    "name": "USD"
}
```

Open Cart Rest API (Cart Module)

Reference Documents:

<https://documenter.getpostman.com/view/4924228/UzXMJmQ>

<https://docs.opencart.com/en-gb/system/users/api/>

Pre-Requisite: Get info from OpenCart Admin application.

Login to Admin → System → Users → API → API Username

Edit → API Key

Edit → IP Address



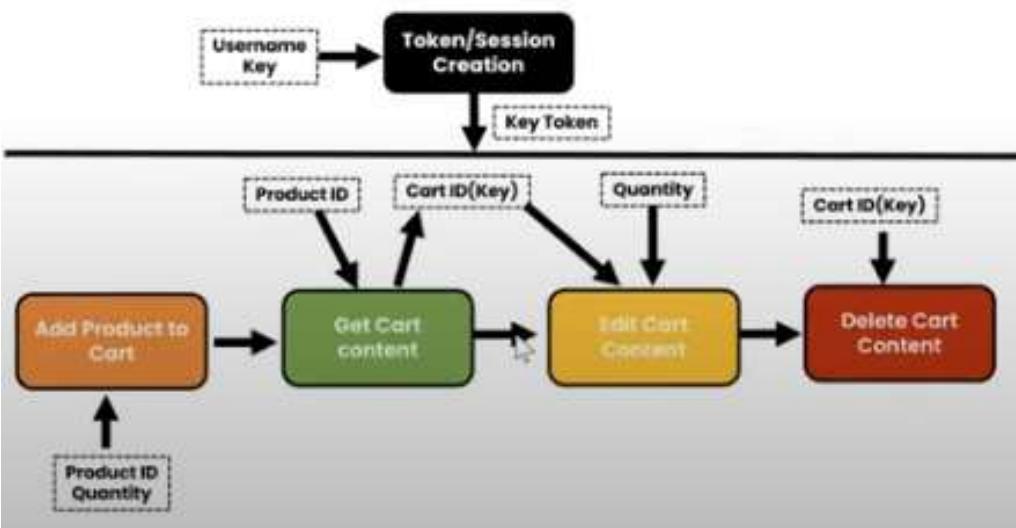
API Username: admin

API Key: 12345678901234567890123456789012

IP Address: 127.0.0.1

API Username:	admin
API Key:	12345678901234567890123456789012

Submit



Day-10

Opencart Application API

ip = 134.56.78.90

baseURL = <http://{{ip}}/index.php?route=>

`{{baseUrl}}api/login`

[{{baseURL}}api/shipping/address](#)

[{{baseURL}}api/cart/add](#)

The screenshot shows the API Tester interface with the following details:

- Collection:** OpenCart_RestAPI_CartModule
- Variables:**
 - variable: INITIAL VALUE: 102.168.0.107 CURRENT VALUE: 102.168.0.107
 - variable: BaseURL: http://102.168.0.107/api/v1/restapi/cart/lines.php?host=102.168.0.107
- Actions:**
 - Create Session Token
 - Add product to cart
 - Cart content
 - Edit product quantity in cart
 - Remove product from cart
- Related Collections:**
 - OpenCart_RestAPI_CartModule
 - OpenCart_RestAPI_UserModule
 - OpenCart_RestAPI_Referral
 - RefRes_HttpRequests
 - RefRes_Httpproxystatus
 - RefRes_JsonRequestValidator
 - RestAPIs
 - StatistAPIs
 - StatistAPIs_Chaining
 - StatistAPIs_ResponseValidation
 - Swagger
- Documentation:** Multi-step editor for your testcases with complete customer descriptions.
- Tools:**
 - ADD TO THIS COLLECTION
 - Create Session Token
 - Add product to cart
 - Cart content
 - Edit product quantity in cart
 - Remove product from cart

myworkspace

New Import Create Session/Token + ***

Opencart_RestAPI_CartModule / Create Session/Token

POST {{baseUrl}}/api/login

Params Authorization Headers (R) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY VALUE DESCRIPTION

username demo Get it from opencart admin

key skopEKSvpFPIWXFJ7qrB2vCE2wkheNX6jIHdw0hI... Get it from opencart admin

Key Value Description

Response

Authentications
BooksAPI
BooksAPI-DataDriven
FileUpload
GorestAPI-Chaining
Opencart_RestAPI_CartModule
Create Session/Token
Add product to cart
Cart content
Edit product quantity in cart
Remove product from cart
PetStore_JSON_UserModel
PetStore_XML_PetModel
ReqRes_httpRequests
ReqRes_httpRequestsVariables
ReqRes_httpRequestsWorkFlow
StudentAPIs
StudentAPIs-Chaining
StudentAPIs_ResponseValidations
swagger

The screenshot shows a browser window with several tabs open, all related to OpenCart documentation. The main content area displays the OpenCart logo and navigation links for Features, Demo, Marketplace, Blog, Showcase, and Resources. A modal window is open, showing a code snippet for logging in:

```
http://myopencart.example.com/index.php?route=api/login;
data:{'username':username, 'key':key}
).text
```

If everything was done right, you'll get json-response with api_token for you're session. Check site admin API page, edit API user and open "Sessions" tab - you can see established session.

Now, what you can do with OpenCart API?

Login

api/login

Establishing session for API user by key PARAMS:
DATA:
username:username from oc_api
key:key from oc_api

EXAMPLE:

```
session.post(
    'http://myopencart.example.com/index.php?route=api/shipping/address',
    params={"api_token":'768ef3818185cd656247ff61d2'},
    data={
        'username':username,
        'key':key
    }
)
```

Home Workspaces API Network Explore

myworkspace

New Import Create Session/Token +

Opencart_RestAPI_CartModule / Create Session/Token

POST `(baseURL)api/login`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```

1 pm.test("Status code is 200", () => {
2   pm.response.to.have.status(200);
3 });
4
5 //capturing response response
6 var jsonData = pm.response.json();
7
8 //Validating JSON response
9 pm.test("checking success msg in response", () => {
10   pm.expect(jsonData.success).to.eql("Success: API session successfully
11     started!");
12 });
13
14 //creating collection variable to store api_token
15 pm.collectionVariables.set("api_token_val",jsonData.api_token)
16

```

Home Workspaces API Network Explore

myworkspace

New Import Create Session/Token +

Opencart_RestAPI_CartModule / Create Session/Token

POST `(baseURL)api/login`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```

1 pm.test("Status code is 200", () => {
2   pm.response.to.have.status(200);
3 });
4
5 //capturing response response
6 var jsonData = pm.response.json();
7
8 //Validating JSON response
9 pm.test("checking success msg in response", () => {
10   pm.expect(jsonData.success).to.eql("Success: API session successfully
11     started!");
12 });
13
14 //creating collection variable to store api_token
15 pm.collectionVariables.set("api_token_val",jsonData.api_token)
16

```

Send

Test script was written in JavaScript and are run after the request is received. See more about test scripts.

Variables

- Set an environment variable
- Set a global variable
- Set a collection variable
- Set an environment variable
- Set a global variable
- Set a collection variable
- Clear an environment variable
- Clear a global variable
- Clear a collection variable
- Save a variable
- Status code: 200 or 200

Body

Pretty JSON Preview Header Test Results (0)

Pretty JSON Preview Header Test Results (0)

1
2 "success": "Success: API session successfully started!",
3 "api_token": "3538a1234567890a5e6f1156"

OpenCartSetup.pdf

localhost:127.0.0.1 / opencart

OpenCartAPI Reference Document API - OpenCart Documentation Opencart_RestAPI_CartModule

localhost/phpmyadmin/index.php?route=sql&db=opencart&t=oc_cart

phpMyAdmin

Information Schema

Opencart

Tables

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Tracking

Triggers

Server: 127.0.0.1 Database: opencart Table: oc_cart

Query results operations

Create view

Bookmark this SQL query

Label:

Let every user access this bookmark

Bookmark this SQL query

cart_id api_id customer_id session_id product_id recurring_id option quantity date_added

Query result set (0 rows) (Query took 0.0004 seconds)

SELECT * FROM `oc_cart`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Windows 11

JSON Path Finder

```

1 {
2   "products": [
3     {
4       "cart_id": "7",
5       "product_id": "40",
6       "name": "iPhone",
7       "model": "product 11",
8       "option": [],
9       "quantity": "2",
10      "stock": true,
11      "shipping": "1",
12      "price": "$123.20",
13      "total": "$246.40",
14      "reward": 0
15    }
16  ],
17  "vouchers": [],
18  "totals": [
19    {
20      "title": "Sub-Total",
21      "text": "$202.00"
22    },
23    {
24      "title": "Eco Tax (-2.00)",
25      "text": "$4.00"
26    },
27    {
28      "title": "VAT (20%)",
29      "text": "$40.40"
30    },
31    {
32      "title": "Total",
33      "text": "$246.40"
34    }
35  ]
36 }

```

Path: `x.products[0].cart_id`

- products:
 - 0:
 - cart_id: `7`
 - product_id: 40
 - name: iPhone
 - model: product 11
 - option:
 - quantity: 2
 - stock: true
 - shipping: 1
 - price: \$123.20
 - total: \$246.40
 - reward: 0
- vouchers:
- totals:

Home Workspaces API Network Explore

myworkspace New Import OpenCart_RestAPI_CartModule QA, Today, 7:56 am

Collections: + ***

APIs: > Authentications > BooksAPI > BooksAPI-DataDriven > FileUpload > GorestAPI-Chaining > OpenCart_RestAPI_CartModule > PetStore_JSON_UserModel > PetStore_XML_PetModel > ReqRes_httpRequests > ReqRes_httpRequestsVariables > ReqRes_httpRequestsWorkflow > StudentAPIs > StudentAPIs-Chaining > StudentAPIs_ResponseValidations > swagger

Environments: RUN SUMMARY

Mock Servers: 1 | 0

Monitors: 2 | 0

Flows: 1 | 0

History: 1 | 0

OpenCart_RestAPI_CartModule QA, Today, 7:56 am

- POST Create Session/Token
 - Pass Status code is 200
 - Pass checking success msg in response
- POST Add product to cart
 - Pass Status code is 200
 - Pass checking success msg in response
- GET Cart content
 - Pass Status code is 200
- POST Edit product quantity in cart
 - Pass Status code is 200
 - Pass checking success msg in response
- POST Remove product from cart
 - Pass Status code is 200
 - Pass checking success msg in response

How many ways we can run collection

Pre-requisite: nodejs + npm

newman

newman-reporter-html

How to run postman collections

Pre-requisites

1. Install Node.js (npm)
2. Install newman
3. Export collection

Step1: Install node.js (Refer the NodeJS installation document)

After installation, open command prompt and check if **node** and **npm** are installed

node -v

npm -v

```

C:\Users\pavan>node -v
v16.15.1

C:\Users\pavan>npm -v
npm [yellow] config global --global, --local are deprecated. Use --location=global instead.
8.11.0

```

```
Microsoft Windows [Version 10.0.25163.1010]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pavan>node -version
node: bad option: -version

C:\Users\pavan>
C:\Users\pavan>node -version
```

```
Microsoft Windows [Version 10.0.25163.1010]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pavan>node -version
node: bad option: -version

C:\Users\pavan>
C:\Users\pavan>node -v
v16.15.1

C:\Users\pavan>npm -v
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
8.11.0

C:\Users\pavan>
```

Step2: Install Newman

```
npm install -g newman
```

```
C:\Users\pavan>npm install -g newman
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 111 packages, and audited 112 packages in 9s

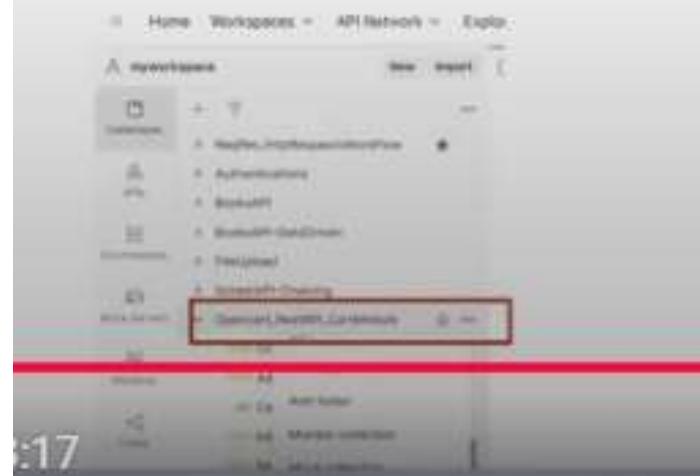
5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.11.0 -> 8.15.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.15.0
npm notice Run `npm install -g npm@8.15.0` to update!
npm notice

C:\Users\pavan>
```

Step3: Export collection

Follow below steps to export collection.



How many ways we can run collection

Pre-requisite: nodejs + npm

2 components are required to run collection in command prompt and generate report

- 1) newman
- 2) newman-reporter-html

```
npm install -g newman
npm install -g newman-reporter-html
```

Step3: Export collection

Follow below steps to export collection.

The screenshot shows the Postman interface. On the left, under 'Collections', there is a tree view with several items. One item, 'Opencart_RestAPI_CartModule', is highlighted with a red box. Below it, under 'Actions', the 'Export' option is also highlighted with a red box. A modal window titled 'EXPORT COLLECTION' is open over the interface. It contains the text 'Skip exporting! Sharing just got quicker & easier.' and 'Share this collection with people directly and work together.' There is a note that 'Opencart_RestAPI_CartModule will be exported as a JSON file. Export as:' followed by two radio button options: 'Collection v2' (selected) and 'Collection v3.1 (Recommended)'. At the bottom of the modal are 'Cancel' and 'Export' buttons, with the 'Export' button highlighted with a red box.

The screenshot shows a 'Save As' file dialog box. The path is 'This PC > Local Disk (C):'. The file name is 'Opencart_RestAPI_CartModule.postman_collection.json'. The 'Save' button is highlighted with a red box. The 'Cancel' button is also visible.

The screenshot shows a Windows File Explorer window with the path 'This PC > Local Disk (C): > mycollections'. Inside the 'mycollections' folder, there is a single file named 'Opencart_RestAPI_CartModule.postman_collection.json'. The file is a JSON file, as indicated by the file type icon. The 'Save' button from the previous dialog is also highlighted with a red box.

At the bottom of the screen, a command prompt window is open with the following text:

```
Microsoft Windows [Version 10.0.25163.1010]
(c) Microsoft Corporation. All rights reserved.

C:\mycollections>newman run Opencart_RestAPI_CartModule.postman_collection.json
```

```

Microsoft Windows [Version 10.0.25163.1018]
(c) Microsoft Corporation. All rights reserved.

C:\mycollections>newman run Opencart_RestAPI_CartModule.postman_collection.json
newman

Opencart_RestAPI_CartModule

+ Create Session/Token
POST http://192.168.0.107/opencart/upload/index.php?route=api/login [200 OK, 627ms]
✓ Status code is 200
✓ checking success msg in response

+ Add product to cart
POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/add&api_token=1505136d89db481d0beac1d685 [200 OK, 3866, 24ms]
✓ Status code is 200
✓ checking success msg in response

+ Cart contents
GET http://192.168.0.107/opencart/upload/index.php?route=api/cart/products&api_token=1505136d89db481d0beac1d685 [200 OK, 6148, 78ms]
✓ Status code is 200

+ Edit product quantity in cart
PUT http://192.168.0.107/opencart/upload/index.php?route=api/cart/edit&api_token=1505136d89db481d0beac1d685 [200 OK, 3868, 47ms]
✓ Status code is 200
✓ checking success msg in response

+ Remove product from cart
POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/remove&api_token=1505136d89db481d0beac1d685 [200 OK, 3065, 50ms]
✓ Status code is 200
✓ checking success msg in response

```

Activator Webdriver

File

```

Select C:\Windows\system32\cmd.exe
V Status code is 200
✓ checking success msg in response

+ Remove product from cart
POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/remove&api_token=1505136d89db481d0beac1d685 [200 OK, 306B, 50ms]
✓ Status code is 200
✓ checking success msg in response



|                    | executed | failed |
|--------------------|----------|--------|
| iterations         | 1        | 0      |
| requests           | 5        | 0      |
| test-scripts       | 10       | 0      |
| prerequest-scripts | 6        | 0      |
| assertions         | 9        | 0      |



total run duration: 1065ms



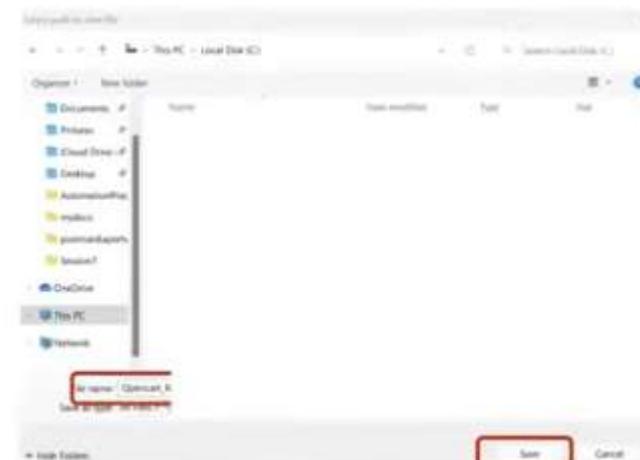
total data received: 644B (approx)



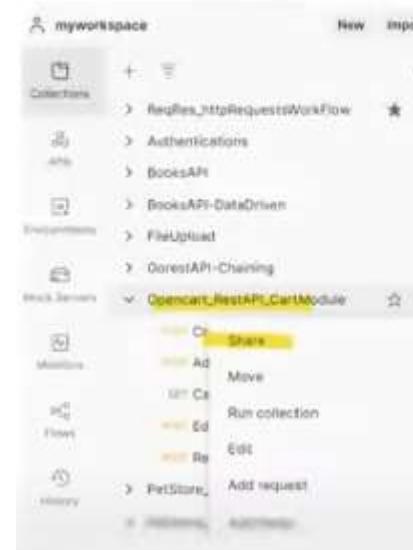
average response time: 86ms [min: 47ms, max: 185ms, s.d.: 50ms]


```

C:\mycollections>



Share Collection Link (For remote execution)



Share Opencart_RestAPI_CartModule

X

With people Via Run in Postman Via JSON link

Generate a shareable public link to a static snapshot of your collection. All your generated links are available in your profile.

This link does not update automatically when changes are made to the collection. Check out the [Postman API](#) to access your synced collections.

[Update Link](#)



<https://www.getpostman.com/collections/02078d0786a69a8fd7ec>

Share Opencart_RestAPI_CartModule

X

With people Via Run in Postman Via JSON link

Generate a shareable public link to a static snapshot of your collection. All your generated links are available in your profile.

This link does not update automatically when changes are made to the collection. Check out the [Postman API](#) to access your synced collections.

[Update Link](#)



<https://www.getpostman.com/collections/02078d0786a69a8fd7ec>

Multiple ways to Run Collection

Method 1 : Run collection locally

newman run Opencart_RestAPI_CartModule.postman_collection.json

Method 2 : Run shared collection remotely

newman run <https://www.getpostman.com/collections/02078d0786a69a8fd7ec>

Method 3 : Run collections & generate html report

Note: We need to install **newman-reporter-html** to generate html report.

Open command prompt then run below command

npm install -g newman-reporter-html

```
Microsoft Windows [Version 10.0.25143.1010]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pavan>npm install -g newman-reporter-html
npm WARN deprecate global --global, --local are deprecated. Use --location=global instead.
npm WARN deprecate global --global, --local are deprecated. Use --location=global instead.
npm WARN deprecate har-validator@0.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use cryptographically problematic random numbers. See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.3.2: Please upgrade to version 7 or higher. Older versions may use cryptographically problematic random numbers. See https://v8.dev/blog/math-random for details.
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please use the API surface has changed to use Promises in 1.x.

added 171 packages, and audited 172 packages in 17s

7 packages are looking for funding
  run `npm fund` for details

37 vulnerabilities (3 moderate, 11 high, 3 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\pavan>
```

newman run Opencart_RestAPI_CartModule.postman_collection.json -r html

newman run <https://www.getpostman.com/collections/02078d0786a69a8fd7ec> -r html

```
{{baseUrl}}api/login
{{baseUrl}}api/shipping/address
{{baseUrl}}api/cart/add
```

How many ways we can run collection

Pre-requisite: nodejs + npm

2 components are required to run collection in command prompt and generate report

- 1) newman
2) newman-reporter-html

```
npm install -g newman
npm install -g newman-reporter-html
```

Approach 1) run collection in command prompt

```
-----  
newman run Opencart_RestAPI_CartModule.postman_collection.json
newman run Opencart_RestAPI_CartModule.postman_collection.json
```

Command Prompt Sublime Text

JSON Path Finder X Postman Report X

File | C:/mycollections/newman/newman-run-report-2022-07-29-02-41-35-160-0.html

Newman Report

Collection: Opencart_RestAPI_CartModule
Time: Fri Jul 29 2022 08:11:35 GMT+0530 (India Standard Time)
Exported with: Newman v6.3.2

	Total	Failed
Iterations	1	0
Requests	5	0
Prerequest Scripts	6	0
Test Scripts	10	0
Assertions	9	0

Total run duration: 905ms
Total data received: 645B (approx)
Average response time: 68ms

Total Failures: 0

Requests

Create Session/Token

Method	URL
POST	http://192.168.0.107/opencart/upload/index.php?route=api/login

Mean time per request: 109ms
Mean size per request: 97B

Total passed tests: 2
Total failed tests: 0

Status code: 200

Tests	Name	Pass count	Fail count
Status code is 200	1	0	
checking success msg in response	1	0	

`{{baseUrl}}api/login`
`{{baseUrl}}api/shipping/address`
`{{baseUrl}}api/cart/add`

How many ways we can run collection

Pre-requisite: nodejs + npm

2 components are required to run collection in command prompt and generate report

1) newman

2) newman-reporter-html

commands to install above components..

`npm install -g newman`

`npm install -g newman-reporter-html`

I

Approach 1) run collection in command prompt

`newman run Opencart_RestAPI_CartModule.postman_collection.json`

`newman run Opencart_RestAPI_CartModule.postman_collection.json -r html`

JSON Path Finder

```

1- {
2-   "products": [
3-     (
4-       "cart_id": "7",
5-       "product_id": "40",
6-       "name": "iPhone",
7-       "model": "product 11",
8-       "option": [],
9-       "quantity": "2",
10-      "stock": true,
11-      "shipping": "1",
12-      "price": "$123.20",
13-      "total": "$246.40",
14-      "reward": 0
15-    )
16-  ],
17-  "vouchers": [],
18-  "totals": [
19-    (
20-      "title": "Sub-Total",
21-      "text": "$202.00"
22-    ),
23-    (
24-      "title": "Eco Tax (-2.00)",
25-      "text": "$4.00"
26-    ),
27-    (
28-      "title": "VAT (20%)",
29-      "text": "$40.40"
30-    ),
31-    (
32-      "title": "Total",
33-      "text": "$246.40"
34-    )
35-  ]
36- }

```

Sample Beautify Minify

Path: `x.products[0].cart_id`

- products:
 - 0:
 - cart_id: 7
 - product_id: 40
 - name: iPhone
 - model: product 11
- vouchers:
- totals:

Home Workspaces API Network Explore

myworkspace

New Import + ...

Collections

- > Authentications
- > BooksAPI
- > BooksAPI-DataDriven
- > FileUpload
- > GorestAPI-Chaining
- > Opencart_RestAPI_CartModule
- > PetStore_JSON
- > PetStore_XML_F
- > ReqRes_httpReq
- > ReqRes_httpReq
- > ReqRes_httpReq

APIs

Environments

Mock Servers

Monitors

Flows

Search Postman

Home Workspaces API Network Explore

myworkspace

New Import + ...

Collections

- > Authentications
- > BooksAPI
- > BooksAPI-DataDriven
- > FileUpload
- > GorestAPI-Chaining
- > Opencart_RestAPI_CartModule
- > PetStore_JSON
- > PetStore_XML_F
- > ReqRes_httpReq
- > ReqRes_httpReq
- > ReqRes_httpReq

APIs

Environments

Mock Servers

Monitors

Flows

History

swagger

Create a fork.

Merge changes

View changelog

View documentation

Rename Ctrl+E

Duplicate Ctrl+D

Export

Search Postman

Share Opencart_RestAPI_CartModule

With people: [Via Run in Postman](#) [Via JSON link](#)

Generate a shareable public link to a static snapshot of your collection. All your generated links are available in your profile.

This link does not update automatically when changes are made to the collection. Check out the [Postman API](#) to access your synced collections.

<https://www.getpostman.com/collections/02078d0786a69a8fd7ec>



The screenshot shows the Postman interface with two requests listed:

- Edit product quantity in cart**
 - Method: POST
 - URL: <http://192.168.1.107/opencart/upload/index.php?route=api/quantity&token=77vac18223a888e44730004210>
 - Mean time per request: 4ms
 - Mean size per request: 4KB
 - Total passed items: 2
 - Total failed items: 0
 - Status code: 200
 - Tests:

Name	Pass count	Fail count
Status code is 200	1	0
Checking success msg in response	1	0
- Remove product from cart**
 - Method: POST
 - URL: <http://192.168.1.107/opencart/upload/index.php?route=api/cart/remove&token=77vac18223a888e44730004210>
 - Mean time per request: 4ms
 - Mean size per request: 4KB

2 components are required to run collection in command prompt and generate report

1) newman

2) newman-reporter-html

commands to install above components..

npm install -g newman

npm install -g newman-reporter-html

Approach 1) run collection in command prompt (local)

newman run Opencart_RestAPI_CartModule.postman_collection.json

newman run Opencart_RestAPI_CartModule.postman_collection.json -r html

Approach 2) run the collection remotely using url.(remote)

<https://www.getpostman.com/collections/02078d0786a69a8fd7ec>

How to run collection in jenkins

```
C:\Windows\system32\cmd.exe - java -jar Jenkins.war
C:\Jenkins>java -jar Jenkins.war
Jul 29, 2022 8:19:44 AM Main verifyJavaVersion
SEVERE: Running with Java class version 54 which is not in the list of supported versions: [52, 55, 61]. Run with the --enable-future-java flag to enable behavior. See https://jenkins.io/redirect/java-support/
java.lang.UnsupportedClassVersionError: 54.0
    at Main.verifyJavaVersion(Main.java:147)
    at Main.main(Main.java:108)

Jenkins requires Java versions [17, 18, 11] but you are running with Java 10 from C:\Program Files\Java\jre-10.0.2
java.lang.UnsupportedClassVersionError: 54.0
    at Main.verifyJavaVersion(Main.java:147)
    at Main.main(Main.java:108)

C:\Jenkins>java -jar Jenkins.war --enable-future-java
Jul 29, 2022 8:20:36 AM Main verifyJavaVersion
WARNING: Running with Java class version 54 which is not in the list of supported versions: [52, 55, 61]. Argument --enable-future-java is set, so will use. See https://jenkins.io/redirect/java-support/
Running from: C:\Jenkins\jenkins.war
webroot: $user.home/.jenkins
2022-07-29 02:50:30.889+0000 [id=1] INFO org.eclipse.jetty.util.log:log#initialized: Logging initialized @943ms to org.eclipse.jetty.util.log.Java
PE
2022-07-29 02:50:31.025+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
2022-07-29 02:50:31.168+0000 [id=1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2022-07-29 02:50:31.168+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-9.4.45.v20220703; built: 2022-07-03T09:14:34.195Z; git: 4a
0be53885e3fcffdcdcc9587d53e11663db; jvm 10.0.2+13
2022-07-29 02:50:32.039+0000 [id=1] INFO o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.j
tyle.jsp.Servlet
2022-07-29 02:50:32.352+0000 [id=1] INFO o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager workerName=mode0
2022-07-29 02:50:32.152+0000 [id=1] INFO o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, using defaults
2022-07-29 02:50:32.152+0000 [id=1] INFO o.e.j.server.Session.Housekeeper#startScavenging: nodes Scavenging every 660000ms
2022-07-29 02:50:33.062+0000 [id=1] INFO Hudson.WebAppMain#contextInitialized: Jenkins home directory: C:\Users\pavan\.jenkins found at $user.h
ome/jenkins
2022-07-29 02:50:33.069+0000 [id=1] INFO o.e.j.s.handler.ContextHandler#doStart: Started ContextHandler@1.8.0_191-b12+0x0{jenkins_v2.346.1,/,file:///C:/Users/pavan/.jen
kins} for Jenkins v2.346.1
[1:36:34 / 2:03:17 - Jenkins (user)]
```

Welcome to Jenkins!

Sign in

```
C:\Windows\system32\cmd.exe - java -jar Jenkins.war --enable-future-java
2022-07-29 02:50:46.330+0000 [id=37] INFO jenkins.InitReactorRunner$1#onAttained: System config loaded
2022-07-29 02:50:46.330+0000 [id=37] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
2022-07-29 02:50:46.330+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2022-07-29 02:50:46.346+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2022-07-29 02:50:46.517+0000 [id=50] INFO hudson.model.AsyncPeriodicWork$lambda$doRun$1: Started Download metadata
2022-07-29 02:50:46.673+0000 [id=50] INFO hudson.util.Retriger#start: Attempt #1 to do the action check updates server
2022-07-29 02:50:46.830+0000 [id=30] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2022-07-29 02:50:46.939+0000 [id=24] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2022-07-29 02:51:07.604+0000 [id=50] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hu
dson.tasks.Maven.MavenInstaller
2022-07-29 02:51:25.457+0000 [id=17] INFO o.j.p.m.GlobalPipelineMavenConfig#getDao: Connect to database jdbc:h2:file
:C:\Users\pavan\.jenkins\jenkins-jobs\jenkins-jobs;AUTO_SERVER=TRUE;MULTI_THREADED=1;QUERY_CACHE_SIZE=25;JMX=TRUE with user
name sa and properties {}
2022-07-29 02:52:34.459+0000 [id=17] INFO c.zaxxer.hikari.HikariDataSource#<init>: HikariPool-1 - Starting...
2022-07-29 02:52:35.481+0000 [id=50] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hu
dson.tasks.Ant.AntInstaller
2022-07-29 02:52:36.704+0000 [id=50] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hu
dson.plugins.gradle.GradleInstaller
2022-07-29 02:52:38.875+0000 [id=50] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hu
dson.tools.JDKInstaller
2022-07-29 02:52:38.877+0000 [id=50] INFO hudson.util.Retriger#start: Performed the action check updates server success
fully at the attempt #1
2022-07-29 02:52:38.883+0000 [id=50] INFO hudson.model.AsyncPeriodicWork$lambda$doRun$1: Finished Download metadata. 1
2,333 ms
2022-07-29 02:52:39.816+0000 [id=17] INFO c.zaxxer.hikari.HikariDataSource#<init>: HikariPool-1 - Start completed.
```

Dashboard +

New Item People Build History Project Relationship Check File Fingerprint Manage Jenkins My Views New View

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

Build Queue No builds in the queue.

Build Executor Status 1 Idle 2 Idle

New Item [Jenkins] +

localhost:8080/view/all/newjob

Jenkins

Dashboard > All >

Enter an item name

opencart_postman_collection

= Required field

Freestyle project This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder

New Item [Jenkins] +

localhost:8080/view/all/newjob

Jenkins

Dashboard > All >

Enter an item name

OpenCart_Postman_Collection

= Required field

Freestyle project This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder

OpenCart_Postman_Collection Configuration

localhost:8080/job/OpenCart_Postman_Collection/configure

Jenkins

Dashboard > OpenCart_Postman_Collection >

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Description

[Plain text] Preview

Discard old builds ?
 GitHub project
 This project is parameterized ?
 Throttle builds ?
 Disable this project ?
 Execute concurrent builds if necessary ?

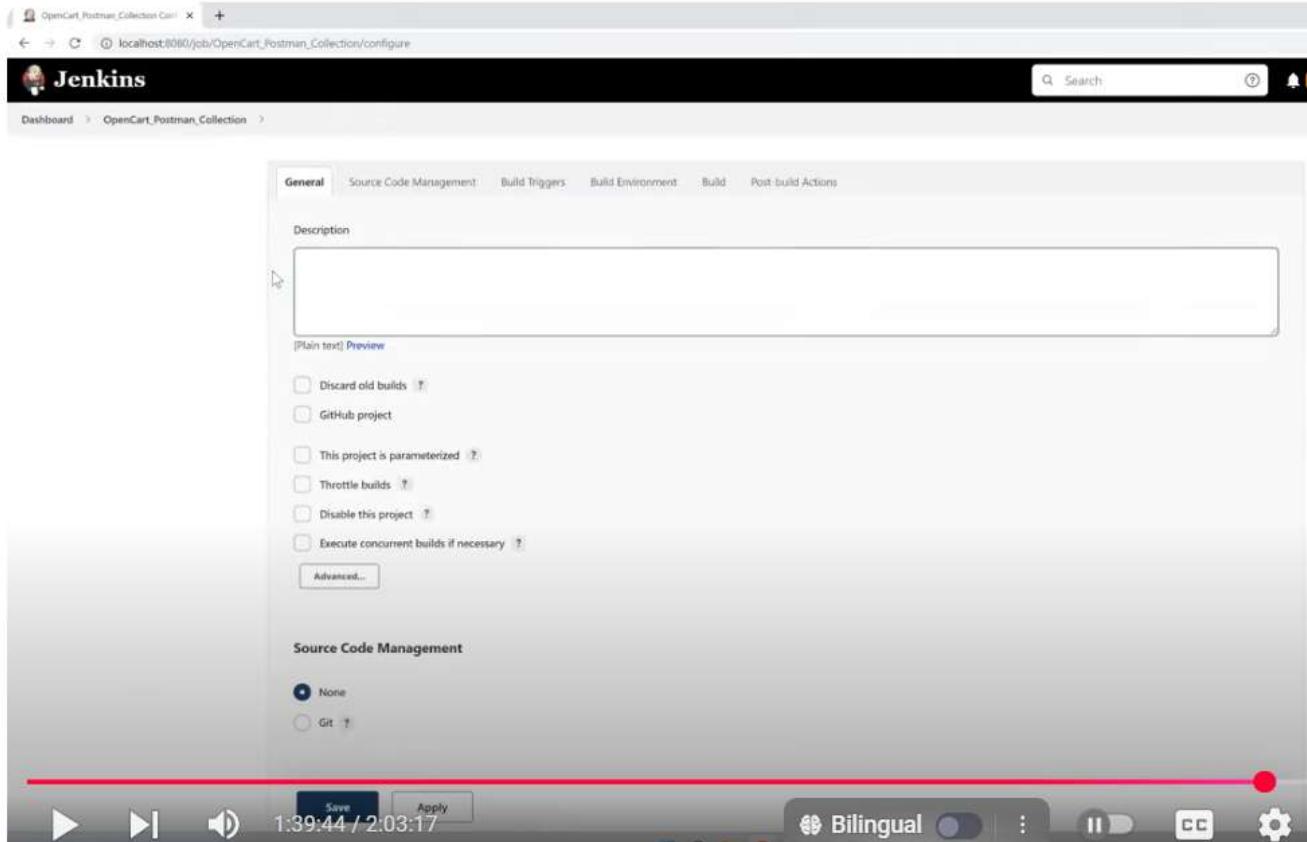
Advanced...

Source Code Management

None
 Git ?

Save Apply

1:39:44 / 2:03:17 Bilingual



OpenCart_Postman_Collection Configuration

localhost:8080/job/OpenCart_Postman_Collection/configure

Dashboard > OpenCart_Postman_Collection >

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

GitHub hook trigger for GITScm polling ?
 Poll SCM ?

Build Environment

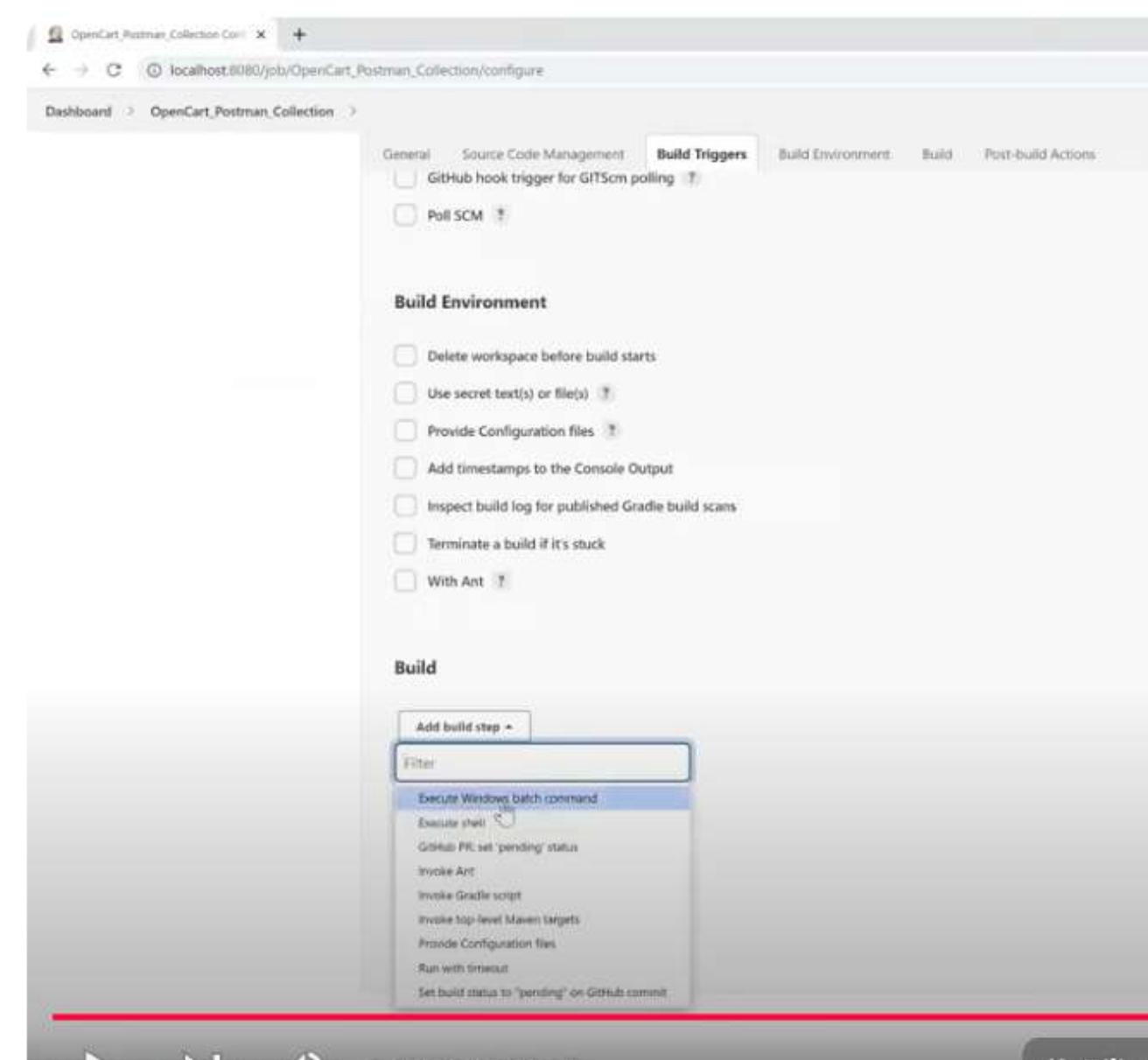
Delete workspace before build starts
 Use secret text(s) or file(s) ?
 Provide Configuration files ?
 Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 Terminate a build if it's stuck
 With Ant ?

Build

Add build step +

Filter

Execute Windows batch command ?
Execute shell ? **Execute shell**
GitHub PR: set 'pending' status
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Provide Configuration files
Run with timeout
Set build status to 'pending' on GitHub commit



OpenCart_Postman_Collection Configuration

General Source Code Management Build Triggers **Build Environment** Build Post-build Actions

Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 Terminate a build if it's stuck
 With Ant

Build

Execute Windows batch command

Command
See the list of available environment variables

```
newman run OpenCart_BestAPI_CartModule.postman_collection.json
```

Advanced...

Add build step

Post-build Actions

Add post-build action

Save Apply



OpenCart_Postman_Collection (Jenkins) localhost:8080/job/OpenCart_Postman_Collection/

Jenkins

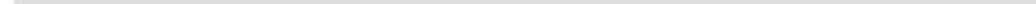
Dashboard > OpenCart_Postman_Collection >

↑ Back to Dashboard Project OpenCart_Postman_Collection

Status Changes Workspace Build Now Recent Changes

Configure Delete Project Rename

Build History trend Filter builds... No builds Atom feed for all Atom feed for failures



Dashboard >

+ New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

OpenCart_Postman_Collection

Icon: S M

Name: OpenCart_Postman_Collection

Changes

Workspace

Build Now

Configure

Delete Project

Rename

OpenCart_Postman_Collection #1

localhost:8080/job/OpenCart_Postman_Collection/1/

Jenkins

Dashboard > OpenCart_Postman_Collection > #1

Back to Project

Status

Changes

Console Output

Edit Build Information

Delete build '#1'

Build #1 (Jul 29, 2022, 8:25:14 AM)

Started by user admin

No changes.

Started by user admin

OpenCart_Postman_Collection #1

localhost:8080/job/OpenCart_Postman_Collection/1/console

Jenkins

Dashboard > OpenCart_Postman_Collection > #1

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#1'

Console Output

Started by user admin

Running as SYSTEM

Building in workspace C:\Users\pavan\.jenkins\workspace\OpenCart_Postman_Collection

[OpenCart_Postman_Collection] \$ cmd /c call C:\Users\pavan\AppData\Local\Temp\jenkins4465621210050828541.bat

C:\Users\pavan\.jenkins\workspace\OpenCart_Postman_Collection>newman run OpenCart_RestAPI_CartModule.postman_collection.json

error: collection could not be loaded

unable to read data from file "OpenCart_RestAPI_CartModule.postman_collection.json"

ENOENT: no such file or directory, open 'C:\Users\pavan\.jenkins\workspace\OpenCart_Postman_Collection\OpenCart_RestAPI_CartModule.postman_collection.json'

Build step 'Execute Windows batch command' marked build as failure

Finished: FAILURE

```
Select C:\WINDOWS\system32\cmd.exe  
C:\mycollections>newman run https://www.getpostman.com/collections/02078d0786a69a8fd7ec -r html  
C:\mycollections>newman run https://www.getpostman.com/collections/02078d0786a69a8fd7ec  
newman  
  
Opencart_RestAPI_CartModule  
  
→ Create Session/Token  
POST http://192.168.0.107/opencart/upload/index.php?route=api/login [200 OK, 627B, 180ms]  
✓ Status code is 200  
✓ checking success msg in response  
  
→ Add product to cart  
POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/add&api_token=b082451031fa454a19c439f40d [200 OK, 306B,  
✓ Status code is 200  
✓ checking success msg in response  
  
→ Cart content  
GET http://192.168.0.107/opencart/upload/index.php?route=api/cart/products&api_token=b082451031fa454a19c439f40d [200 OK, 61B  
✓ Status code is 200  
  
→ Edit product quantity in cart  
POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/edit&api_token=b082451031fa454a19c439f40d [200 OK, 306B,  
✓ Status code is 200  
✓ checking success msg in response  
  
→ Remove product from cart  
POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/remove&api_token=b082451031fa454a19c439f40d [200 OK, 306B  
✓ Status code is 200  
✓ checking success msg in response
```

The screenshot shows a Jenkins job configuration page for an 'OpenCart Postman Collection'. The top navigation bar includes links for 'Dashboard', 'Jobs', 'Cloud', 'Pipeline', 'Queue', 'New Item', and 'Help'.

The main configuration area is divided into several tabs:

- General**: Basic job details like name and description.
- Source Code Management**: Set to 'None'.
- Build Triggers**: Set to 'None'.
- Build Environment** (selected tab):
 - Provides configuration files (checkbox): Unchecked.
 - Add timestamps to the Console Output (checkbox): Unchecked.
 - Inspect build log for published Gradle build scans (checkbox): Unchecked.
 - Terminate a build if it's stuck (checkbox): Unchecked.
 - With Ant (checkbox): Unchecked.
- Build**:
 - Execute Windows batch command:
 - Command: newman run https://www.getpostman.com/collections/02078d0786a69a8fd7ec
 - Advanced... button
 - Add build step + button
- Post-build Actions**:
 - Add post-build action + button

At the bottom are 'Save' and 'Apply' buttons.

OpenCart_Postman_Collection #2

localhost:8080/jenkins/OpenCart_Postman_Collection/2/console

Jenkins

Dashboard > OpenCart_Postman_Collection > #2

Back to Project Status Changes Console Output View as plain text Edit Build Information Delete build #2 Previous Build

Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace C:\Users\pavan\jenkins\workspace\OpenCart_Postman_Collection
[OpenCart_Postman_Collection] $ cmd /c call C:\Users\pavan\AppData\Local\Temp\jenkins11685891868174768727.bat

C:\Users\pavan\jenkins\workspace\OpenCart_Postman_Collection>newman run https://www.getpostman.com/collections/82078d8786a69a8fd7ec
newman

Opencart_RestAPI_CartModule

  * Create Session/Token
    POST http://192.168.0.107/opencart/upload/index.php?route=api/login [200 OK, 6278, 138ms]
      * Status code is 200
      * checking success msg in response

  * Add product to cart
    POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/add&api_token=aa367fbe012082932cb938f27c [200 OK, 3868, 75ms]
      * Status code is 200
      * checking success msg in response

  * Cart content
    GET http://192.168.0.107/opencart/upload/index.php?route=api/cart/products&api_token=aa367fbe012082932cb938f27c [200 OK, 6158, 76ms]
      * Status code is 200

  * Edit product quantity in cart
    POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/edit&api_token=aa367fbe012082932cb938f27c [200 OK, 3068, 58ms]
      * Status code is 200
      * checking success msg in response

  * Remove product from cart
    POST http://192.168.0.107/opencart/upload/index.php?route=api/cart/remove&api_token=aa367fbe012082932cb938f27c [200 OK, 3068, 50ms]
      * Status code is 200
      * checking success msg in response
```

OpenCart_Postman_Collection #3

GitHub

Search or jump to... Pull requests Issues Marketplace Explore

Recent Repositories Find a repository...

- pavanotraining/OpencartV123
- pavanotraining/Opencart_V121
- pavanotraining/MySampleProject
- pavanotraining/TestDemo2
- pavanotraining/testdemo
- pavanotraining/inetbankingV1
- pavanotraining/nopCommerceV1

Show more

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

chandinishetty started following you 14 hours ago

chandinishetty 1 repository

nikhil-k1200 forked nikhil-k1200/JenkinsPipelineDemoProject from pavanotraining/JenkinsPipelineDemoProject 15 hours ago

pavanotraining/JenkinsPipelineDemoProject

ChangeMak started following you 18 hours ago

Viresh Borkar ChangeMak 3 repositories

abhimanyu-sun started following you yesterday

abhimanyu kumar abhimanyu-sun software engineer 6 repositories

OpenCart_Pushcart_Collection 83 · Create a New Repository

Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Repository name *

 pavanoltraining / opencart_API_testing 

Great repository names are short: opencart_API_testing is available. What's in a name? How about miniflutter-discord?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. Learn more.

Add .gitignore Choose which files not to track from a list of templates. Learn more.

.gitignore template: None ▾

Choose a license A license tells others what they can and can't do with your code. Learn more.

License: None ▾

```
npm install -g newman
npm install -g newman-reporter-html
```

Approach 1) run collection in command prompt (local)

newman run OpenCart_RestAPI_CartModule.postman_collection.json

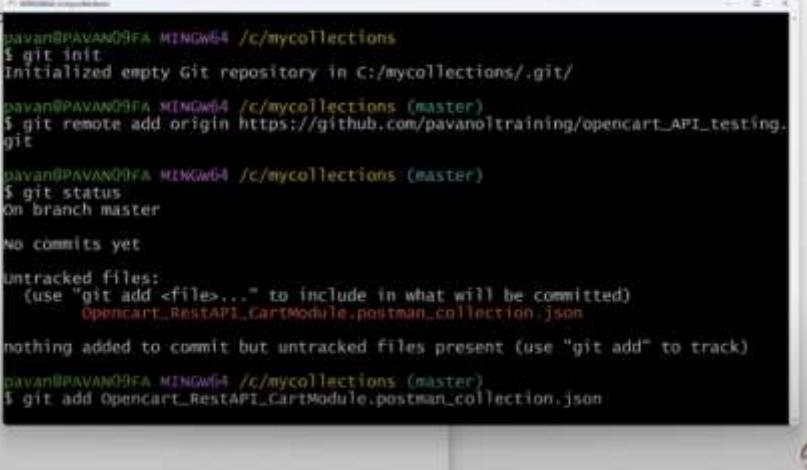
Approach 2) run collection in command prompt (remote)

https://www.getpostman.com/collections/5f3a2a2a7d8a4d0d39e0/

--> How to run

Remote repo I https://github.com/pavanoltraining/opencart_API_testing.git

Create local repository



```
pavan@PAVAN09FA MINGW64 ~/c/mycollections
$ git init
Initialized empty Git repository in C:/mycollections/.git/
pavan@PAVAN09FA MINGW64 ~/c/mycollections (master)
$ git remote add origin https://github.com/pavanoltraining/opencart_API_testing.git
pavan@PAVAN09FA MINGW64 ~/c/mycollections (master)
$ git status
On branch master

No commits yet

untracked files:
  (use "git add <file>..." to include in what will be committed)
    OpenCart_RestAPI_CartModule.postman_collection.json

nothing added to commit but untracked files present (use "git add" to track)

pavan@PAVAN09FA MINGW64 ~/c/mycollections (master)
$ git add OpenCart_RestAPI_CartModule.postman_collection.json
```

```
npm install -g newman
npm install -g newman-reporter-html
```

Approach 1) run collection in command prompt (local)

newman run OpenCart_RestAPI_CartModule.postman_collection.json

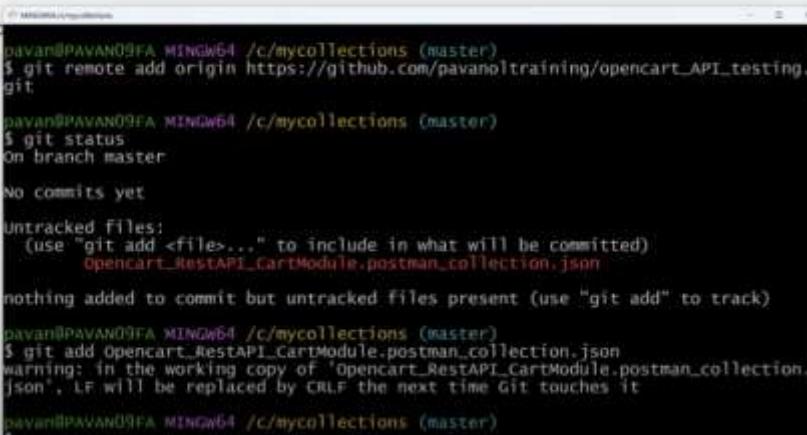
Approach 2) run collection in command prompt (remote)

https://www.getpostman.com/collections/5f3a2a2a7d8a4d0d39e0/

--> How to run

Remote repo I https://github.com/pavanoltraining/opencart_API_testing.git

Create local repository



```
pavan@PAVAN09FA MINGW64 ~/c/mycollections (master)
$ git remote add origin https://github.com/pavanoltraining/opencart_API_testing.git
pavan@PAVAN09FA MINGW64 ~/c/mycollections (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    OpenCart_RestAPI_CartModule.postman_collection.json

nothing added to commit but untracked files present (use "git add" to track)

pavan@PAVAN09FA MINGW64 ~/c/mycollections (master)
$ git add OpenCart_RestAPI_CartModule.postman_collection.json
warning: in the working copy of 'OpenCart_RestAPI_CartModule.postman_collection.json', LF will be replaced by CRLF the next time Git touches it

pavan@PAVAN09FA MINGW64 ~/c/mycollections (master)
```

File Edit View

```
npm install -g newman  
npm install -g newman-reporter-html
```

Approach 1) run collection in command prompt (local)

```
newman run Opencart_RestAPI_CartModule.postman_collection.json
```

```
warning: in the working copy of 'Opencart_RestAPI_CartModule.postman_collection.json', LF will be replaced by CRLF the next time Git touches it
```

```
pavan@PAVAN09FA MINGW64 /c/mycollections (master)
```

```
$ git status
```

```
git: 'status' is not a git command. See 'git --help'.
```

```
The most similar command is  
status
```

```
pavan@PAVAN09FA MINGW64 /c/mycollections (master)
```

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file: Opencart_RestAPI_CartModule.postman_collection.json
```

```
pavan@PAVAN09FA MINGW64 /c/mycollections (master)
```

```
$
```

Approach 2) run collection in command prompt (remote)

```
https://www.getpostman.com/collections/02078d0786a69a8fd7ec
```

```
--> How to run collection in jenkins
```

Remote repo URL..

```
https://github.com/pavanoltraining/opencart_API_testing.git
```

Create local repository

```
git init
```

1) Create local repo:

```
git init
```

2) Connected local repo with remote repository(github)

```
git remote add origin https://github.com/pavanol
```

3) add file to index/stage

```
git add filename
```

4) git commit -m "comment" filename

```
pavan@PAVAN09FA MINGW64 /c/mycollections (master)
```

```
$ git status
```

```
git: 'status' is not a git command. See 'git --help'.
```

```
The most similar command is  
status
```

```
pavan@PAVAN09FA MINGW64 /c/mycollections (master)
```

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file: Opencart_RestAPI_CartModule.postman_collection.json
```

```
pavan@PAVAN09FA MINGW64 /c/mycollections (master)
```

```
$ git commit -m "first commit"
```

newman run Opencart_RestAPI_CartModule.postman_collection.json -r html

Approach 2) run the collection remotely using url.(remote)

```
https://www.getpostman.com/collections/02078d0786a69a8fd7ec
```

--> How to run collection in jenkins

Remote repo URL..

```
https://github.com/pavanoltraining/opencart_API_testing.git
```

1) Create local repo:

```
git init
```

2) Connected local repo with remote repository(github)

```
git remote add origin https://github.com/pavanoltraining/opencart_API_testing.git
```

3) add file to index/stage

```
git add filename
```

4) git commit -m "comment"

Dashboard (jenkins) x [pavanoltraining/opencart_API_test](#) x + localhost:8080

Jenkins

New item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

New View

Build Queue

No builds in the queue.

Build Executor Status

1 idle

2 idle

New Item Creation x [pavanoltraining/opencart_API_test](#) x + localhost:8080

Jenkins

Dashboard All

Enter an item name

OpenCart_API_GitHub - Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as流水线) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores related items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

MultiBranch Pipeline
Creates a set of Pipeline projects according to selected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project substituted by scanning for repositories.

Remote repo URL..
https://github.com/pavanoltraining/opencart_API_testing.git

- 1) Create local repo:
git init
- 2) Connected local repo with remote repository(github)
git remote add origin https://github.com/pavanoltraining/opencart_API_testing.git
- 3) add file to index/stage
git add filename
- 4) git commit -m "comment"
- 5) git push origin master

```
pavan@PAVAN09FA MINGW64 /c/mycollections (master)
(use "git rm --cached <file>..." to unstage)
  new file:   Opencart_RestAPI_CartModule.postman_collection.json

pavan@PAVAN09FA MINGW64 /c/mycollections (master)
$ git commit -m "first commit"
[master (root-commit) 6646ba5] first commit
 1 file changed, 409 insertions(+)
 create mode 100644 Opencart_RestAPI_CartModule.postman_collection.json

pavan@PAVAN09FA MINGW64 /c/mycollections (master)
$ git status
On branch master
nothing to commit, working tree clean

pavan@PAVAN09FA MINGW64 /c/mycollections (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.72 KiB | 587.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

commands to install above components..

```
npm install -g newman
npm install -g newman-reporter-html
```

Approach 1) run collection in command prompt (local)

```
-----
newman run Opencart_RestAPI_CartModule.postman_collection.json
newman run Opencart_RestAPI_CartModule.postman_collection.json -r html
```

Approach 2) run the collection remotely using url.(remote)

Shared URL: <https://www.getpostman.com/collections/02078d0786a69a8fd7ec>

```
newman run https://www.getpostman.com/collections/02078d0786a69a8fd7ec -r html
```

-- > How to run collection in jenkins

Remote repo URL..

https://github.com/pavanoltraining/opencart_API_testing.git

```
newman run Opencart_RestAPI_CartModule.postman_collection.json  
newman run Opencart_RestAPI_CartModule.postman_collection.json -r html
```

Approach 2) run the collection remotely using url.(remote)
Shared URL: <https://www.getpostman.com/collections/02078d0786a69a8fd7ec>

```
newman run https://www.getpostman.com/collections/02078d0786a69a8fd7ec -r html
```

Approach 2) Run the collection in jenkins(locally)

Approach 3) Run the colelction in jenkins from Github

Remote repo URL:

https://github.com/pavanoltraining/opencart_API_testing.git

1) Create local repo:

```
git init
```

2) Connected local repo with remote repository(github)

```
git remote add origin https://github.com/pavanoltraining/opencart_API_testing.git
```

3) add file to index/stage

```
git add filename
```