“Postman Documentation”

Topic of content

Required files and software:

- Nodejs: https://nodejs.org/en/download
- Sublime Text: https://www.sublimetext.com/
- Json-server : https://github.com/typicode/json-server
- Json code checker: https://www.jsonformatter.io/
- w3schools : https://www.w3schools.com/whatis/whatis_json.asp

What is “Node.js”?

- Node.js is an open source server environment. Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc. More Details

What is “NPM”?

- NPM is a package manager for Node.js packages, or modules, if you like. The NPM program is installed on your computer when you install Node.js

What is “JSON”?

JSON stands for JavaScript Object Notation. JSON is a text format for storing and transporting data.

“Postman Documentation”

JSON Syntax Rules

1. Data is in name/value pairs **->** **"**firstName":"Marufa**"**

2. Data is separated by commas **->**
{"firstName":"Marufa"**,**"MidName":"Akter"**,**"lastName":"Eity"}

3. Curly braces hold objects **->** **{**"firstName":"John", "lastName":"Doe"**}**

4. Square brackets hold arrays **->**

```
"employees":[

   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
]
```

JSON Examples:

```
{
    "abc” :       [
             {
                   "id": 1,
                   "Name” : "XYZ”,
                 ”Address”: “zxy”
             }
                ]
}
```

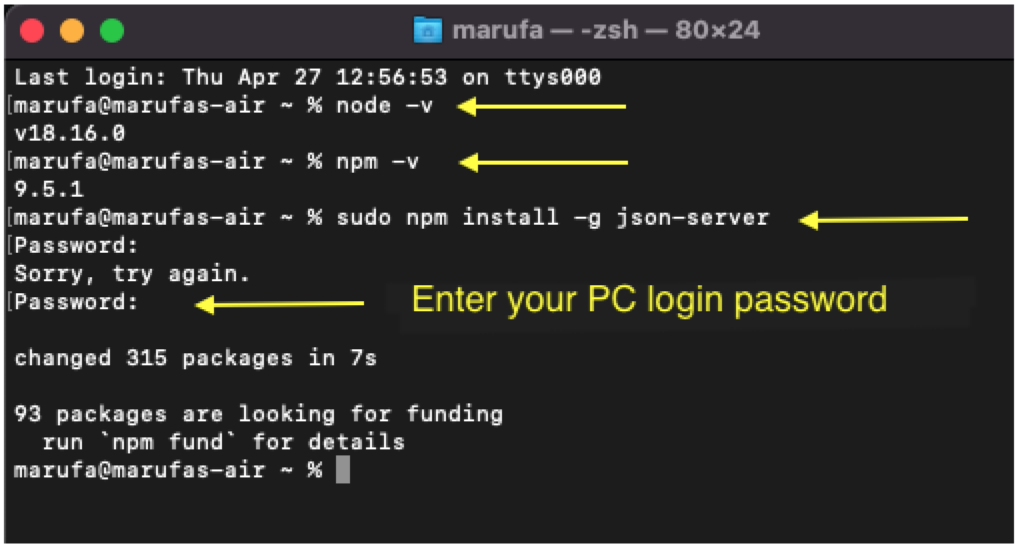
Ex 02: This example defines employees object: an array of 3 employee records (objects):

```
{
"employees":[
        {"firstName":"John", "lastName":"Doe"},
        {"firstName":"Anna", "lastName":"Smith"},
        {"firstName":"Peter", "lastName":"Jones"}
]
}
```

“Node.js” and “npm” installation:

1. Type command “**node -v**” in terminal to check that Node.js is installed or not.
2. Type command “**npm -v**” in terminal to check “npm” available or not.

3. Install JSON Server – Go through this “https://github.com/typicode/json-server ” link and follow instruction from “README.md” file> Getting started > Install JSON Server to create json server and run server.
4. Type command “**sudo npm install -g json-server**” to Install json server and give your PC login password.
5. Type command “**json-server --watch db.json**” in terminal to active json



```
Last login: Thu Apr 27 12:56:53 on ttys000
[marufa@marufas-air ~ % node -v
v18.16.0
[marufa@marufas-air ~ % npm -v
9.5.1
[marufa@marufas-air ~ % sudo npm install -g json-server
[Password:
Sorry, try again.
[Password:

changed 315 packages in 7s

93 packages are looking for funding
  run `npm fund` for details
marufa@marufas-air ~ %
```

server.

Create & run json server:

1. Type command “**json-server --watch post.json**” in terminal to active json server.

- Search **“post.json”** file from download option and pest bellow Json code.

```
{
"posts": [
{ "id": 1, "title": "json-server", "author": "typicode" }
],

"comments": [
{ "id": 1, "body": "some comment", "postId": 1 }
],

"profile": { "name": "typicode" }
}
```

Now if you go to http://localhost:3000/posts/1, And you'll get

```
> { "id": 1, "title": "json-server", "author": "typicode" }
```

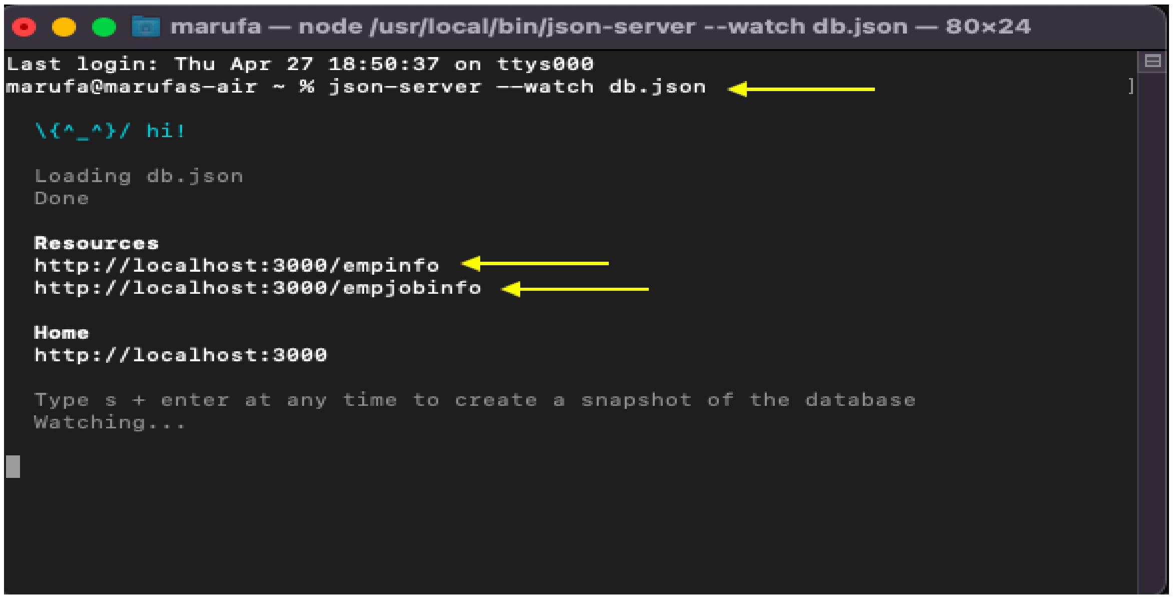
2. Example 01,

Type command “**json-server --watch db.json**” in terminal to active json server.

- Search **“db.json”** file from download option and pest bellow Json code.

Let’s edit the “db.json” file and create a json and must save the file.

```
{
  "empinfo": [

    {
      "id": 1,
      "Name": "Uzzal",
      "Address": "Mirpur 13",
      "Contact": "+880 1761724175",
      "DOB":"1-07-2001"
    },

    {
      "id": 2,
      "Name": "Sharon",
      "Address": "Mirpur Matikata",
      "Contact": "+880 1761734076",
      "DOB":"1-07-2002"
    },

     {
      "id": 3,
      "Name": "Nayeem",
      "Address": "Mohammadpur",
      "Contact": "+880 1761 794077",
      "DOB":"1-07-2004"
    },

    {
      "id": 4,
      "Name": "Tawhid",
      "Address": "Uttora",
      "Contact": "+880 1761794087",
        "DOB":"1-07-2003"
}
],



  "empjobinfo": [

     {
        "id": 101,
        "JobTitle": "Software QA Engineer"
     },

     {
        "id": 102,
        "JobTitle": "SQA Engineer"
     },

     {
        "id": 103,
        "JobTitle": "Jr.SQA Engineer"
     },

     {
        "id": 104,
        "JobTitle": "Jr.SQA Engineer"
     }


  ]
}
```



```
marufa — node /usr/local/bin/json-server --watch db.json — 80×24
Last login: Thu Apr 27 18:50:37 on ttys000
marufa@marufas-air ~ % json-server --watch db.json

  \{^_^}/ hi!

  Loading db.json
  Done

  Resources
  http://localhost:3000/empinfo
  http://localhost:3000/empjobinfo

  Home
  http://localhost:3000

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

3. Go to this link in browser “http://localhost:3000” to check the running server and check below resources.

Check output:

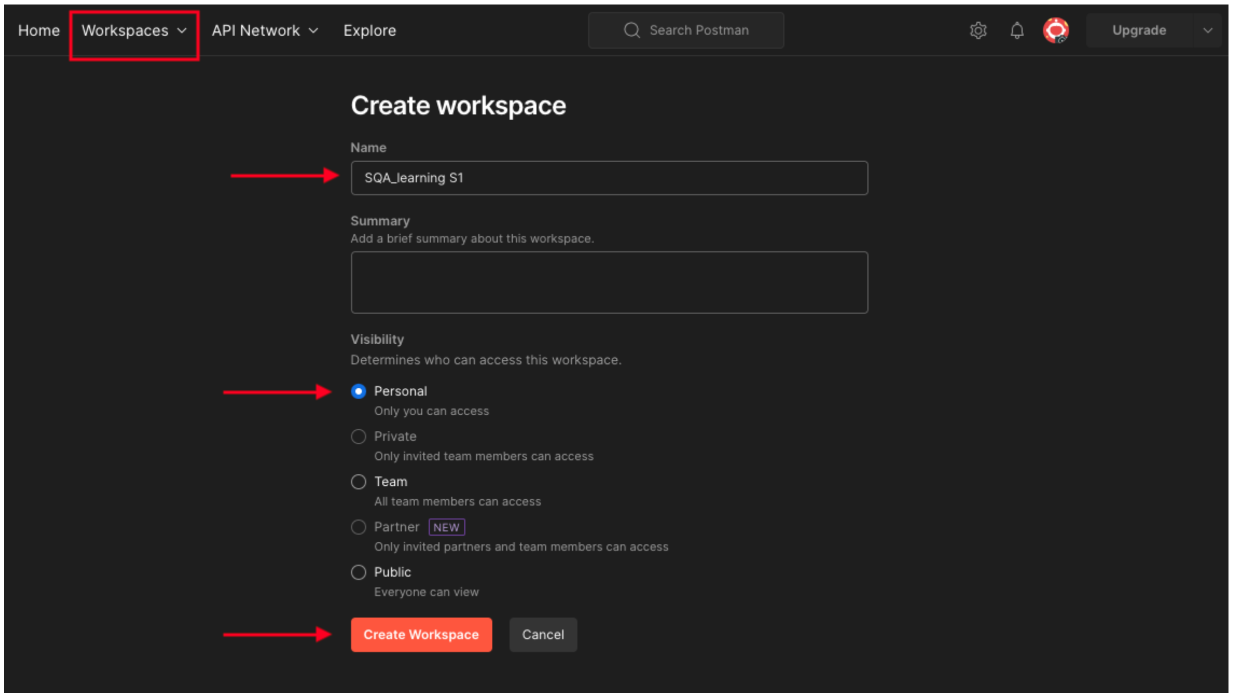
http://localhost:3000/empinfo
http://localhost:3000/empjobinfo

4. If found any error, then debug and correct the code to **Json code checker:** https://www.jsonformatter.io/

Check this json in postman

1. Go to postman and check it by Get request http://localhost:3000/
- http://localhost:3000/empinfo
- http://localhost:3000/empjobinfo

All steps with images:



Create workspace

Name

SQA_learning S1

Summary
Add a brief summary about this workspace.

Visibility
Determines who can access this workspace.

● Personal
Only you can access

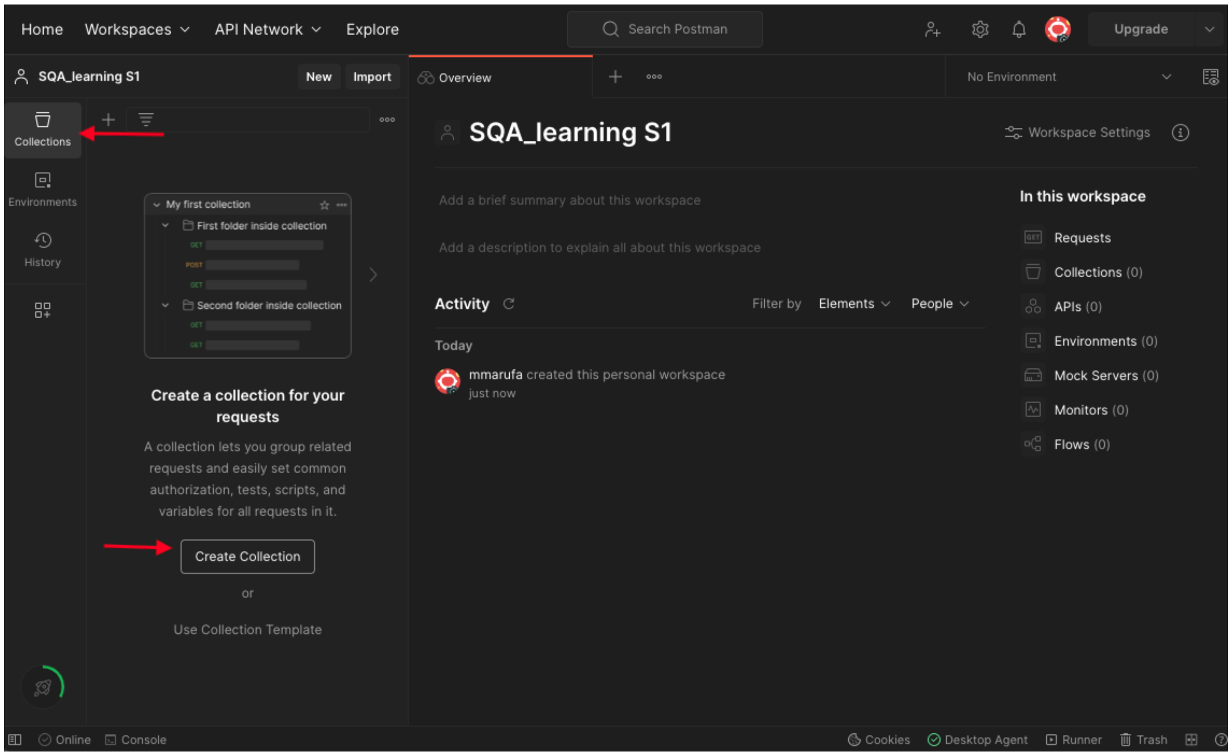
○ Private
Only invited team members can access

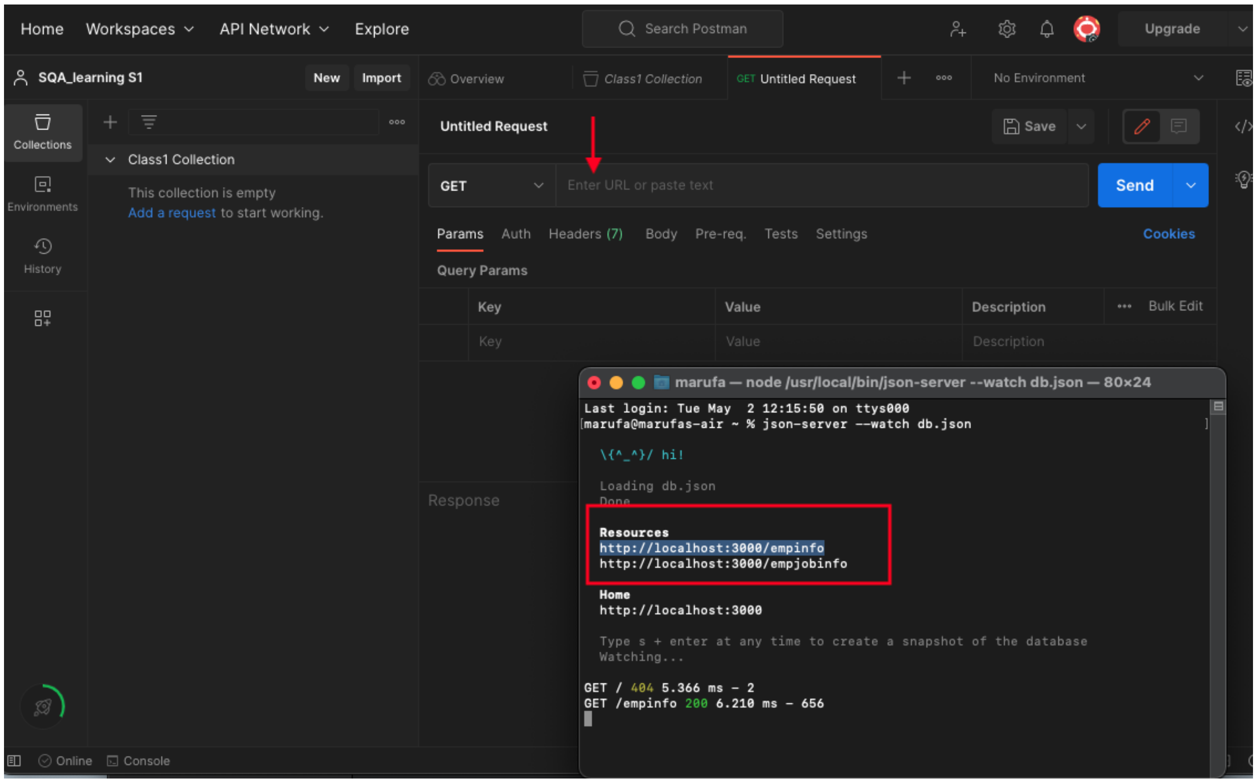
○ Team
All team members can access

○ Partner NEW
Only invited partners and team members can access

○ Public
Everyone can view

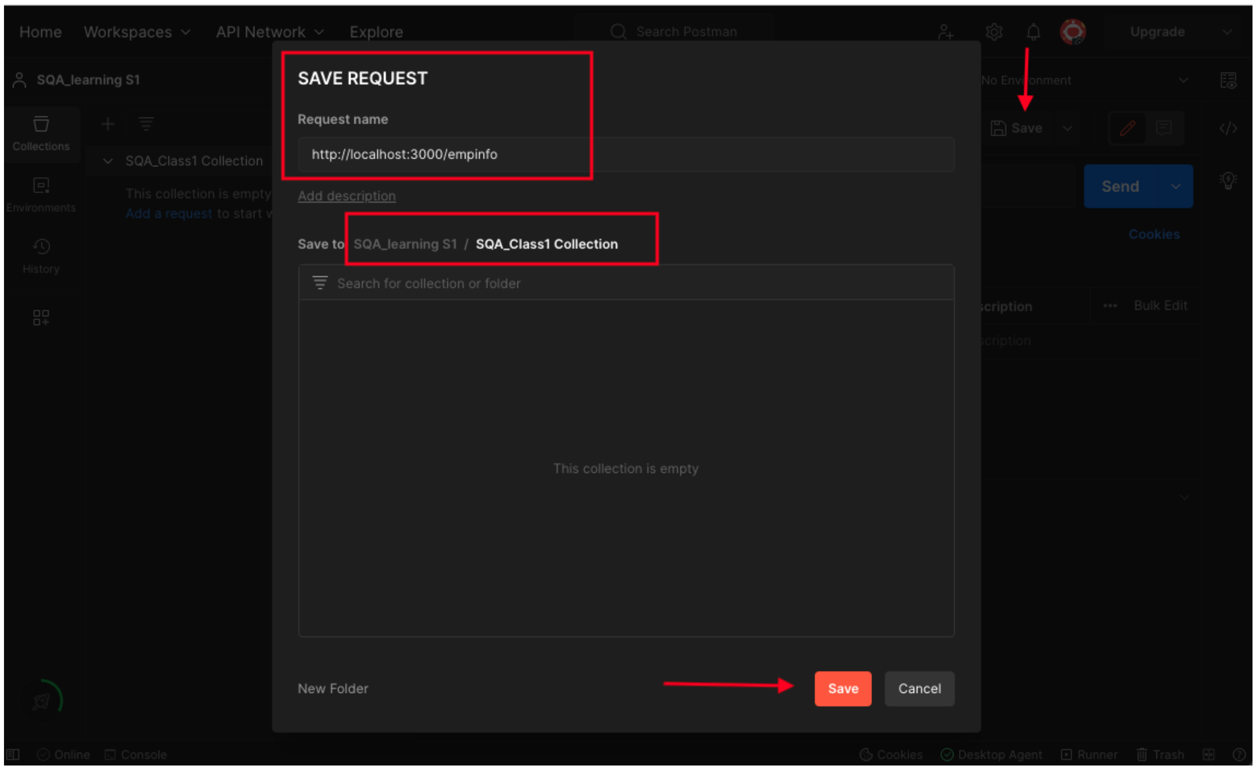
Create Workspace | Cancel





```
Last login: Tue May  2 12:15:50 on ttys000
marufa@marufas-air ~ % json-server --watch db.json

  \{^_^}/ hi!

  Loading db.json
  Done

  Resources
  http://localhost:3000/empinfo
  http://localhost:3000/empjobinfo

  Home
  http://localhost:3000

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

GET / 404 5.366 ms - 2
GET /empinfo 200 6.210 ms - 656
```

“Postman Documentation”



SAVE REQUEST

Request name

http://localhost:3000/empinfo

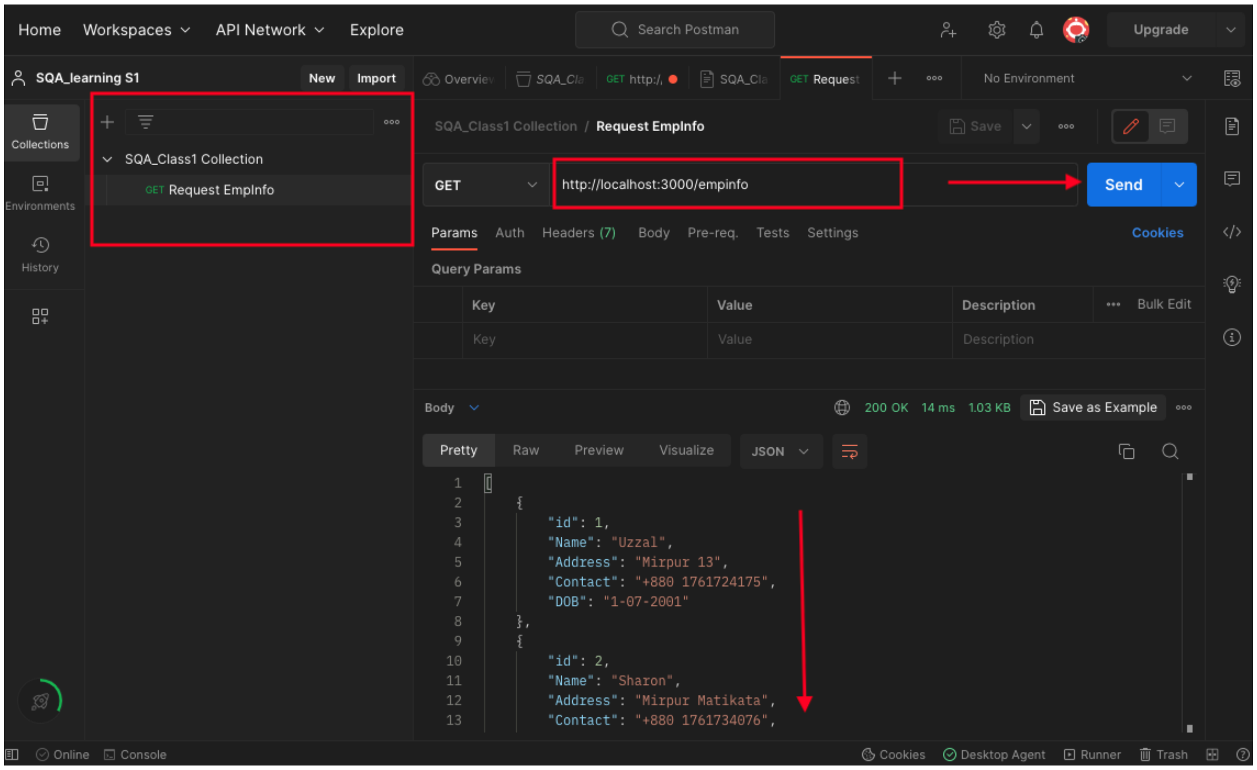
Add description

Save to SQA_learning S1 / SQA_Class1 Collection

Search for collection or folder

This collection is empty

New Folder



SQA_Class1 Collection / Request EmpInfo

GET http://localhost:3000/empinfo

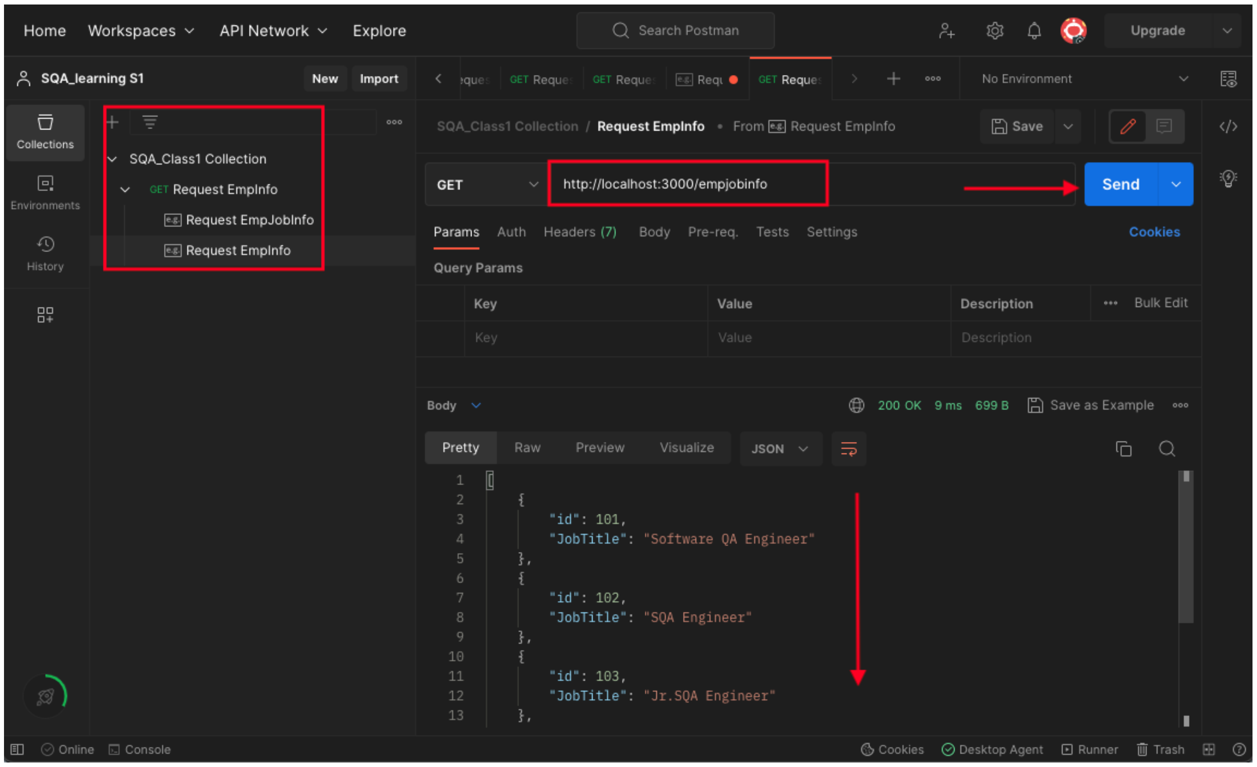
Query Params

Key	Value	Description
Key	Value	Description

Body 200 OK 14 ms 1.03 KB

```
[
    {
        "id": 1,
        "Name": "Uzzal",
        "Address": "Mirpur 13",
        "Contact": "+880 1761724175",
        "DOB": "1-07-2001"
    },
    {
        "id": 2,
        "Name": "Sharon",
        "Address": "Mirpur Matikata",
        "Contact": "+880 1761734076",
```

“Postman Documentation”



```
GET http://localhost:3000/empjobinfo
```

```
[
    {
        "id": 101,
        "JobTitle": "Software QA Engineer"
    },
    {
        "id": 102,
        "JobTitle": "SQA Engineer"
    },
    {
        "id": 103,
        "JobTitle": "Jr.SQA Engineer"
    },
```

- **Postman Documentation(2nd class)**

1. **Nested JSON:**
— A sample nested JSON format

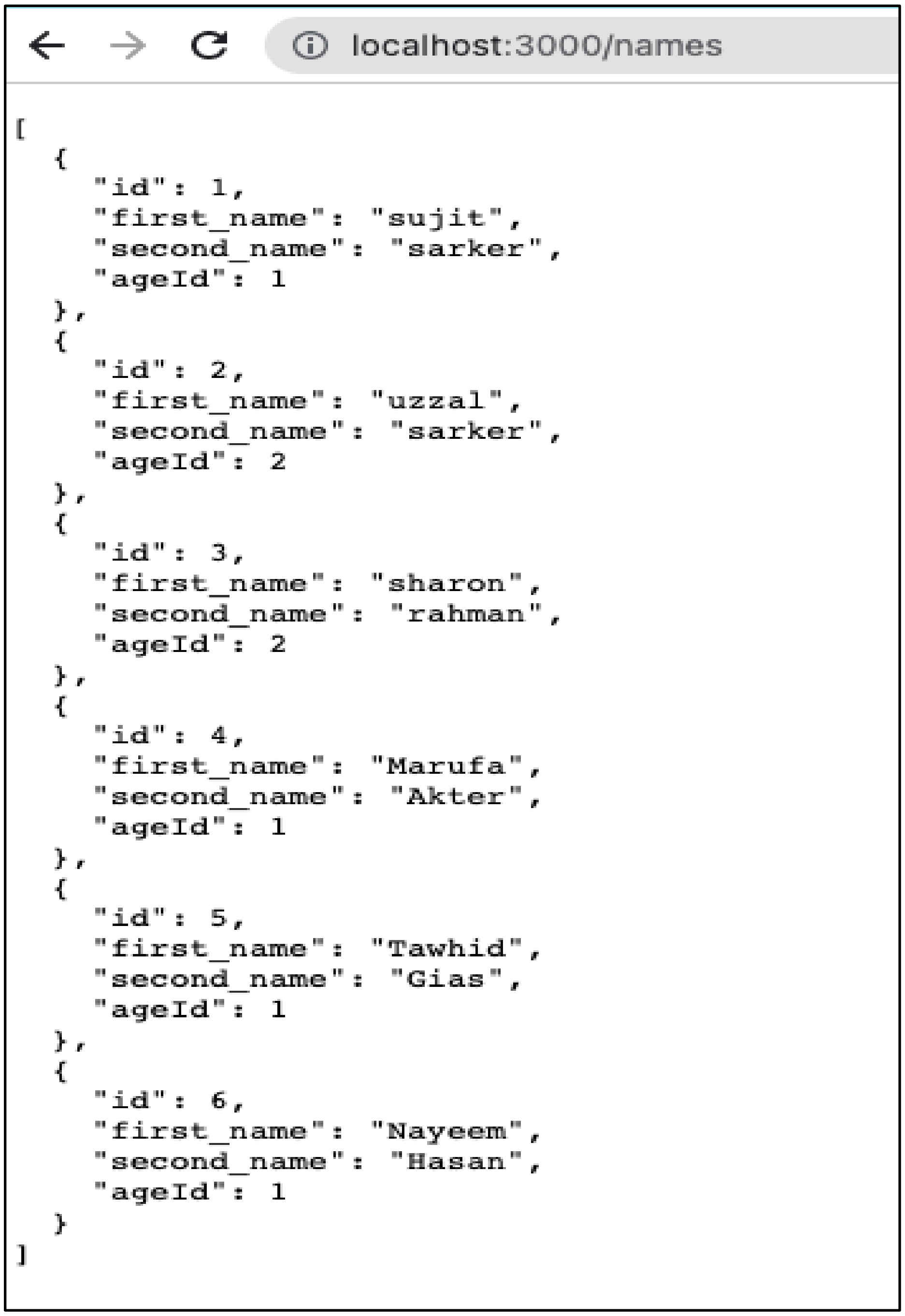
```
{
  "names": [
    {
      "id": 1,
      "first_name": "sujit",
      "second_name": "sarker",
      "ageId": 1
    },
    {
      "id": 2,
      "first_name": "uzzal",
      "second_name": "sarker",
      "ageId": 2
    },
    {
      "id": 3,
      "first_name": "sharon",
      "id": 1
    },
    {
      "number": "65",
      "id": 2
    }
  ]
}
```

```
      "second_name": "rahman",
      "ageId": 2
    },
    {
      "id": 4,
      "first_name": "Marufa",
      "second_name": "Akter",
      "ageId": 1
    },
    {
      "id": 5,
      "first_name": "Tawhid",
      "second_name": "Gias",
      "ageId": 1
    },
    {
      "id": 6,
      "first_name": "Nayeem",
      "second_name": "Hasan",
      "ageId": 1
    }
  ],
  "ages": [
    {
      "number": "70",
```

- **Uses of Get, Post, Put, Patch, Delete method:**

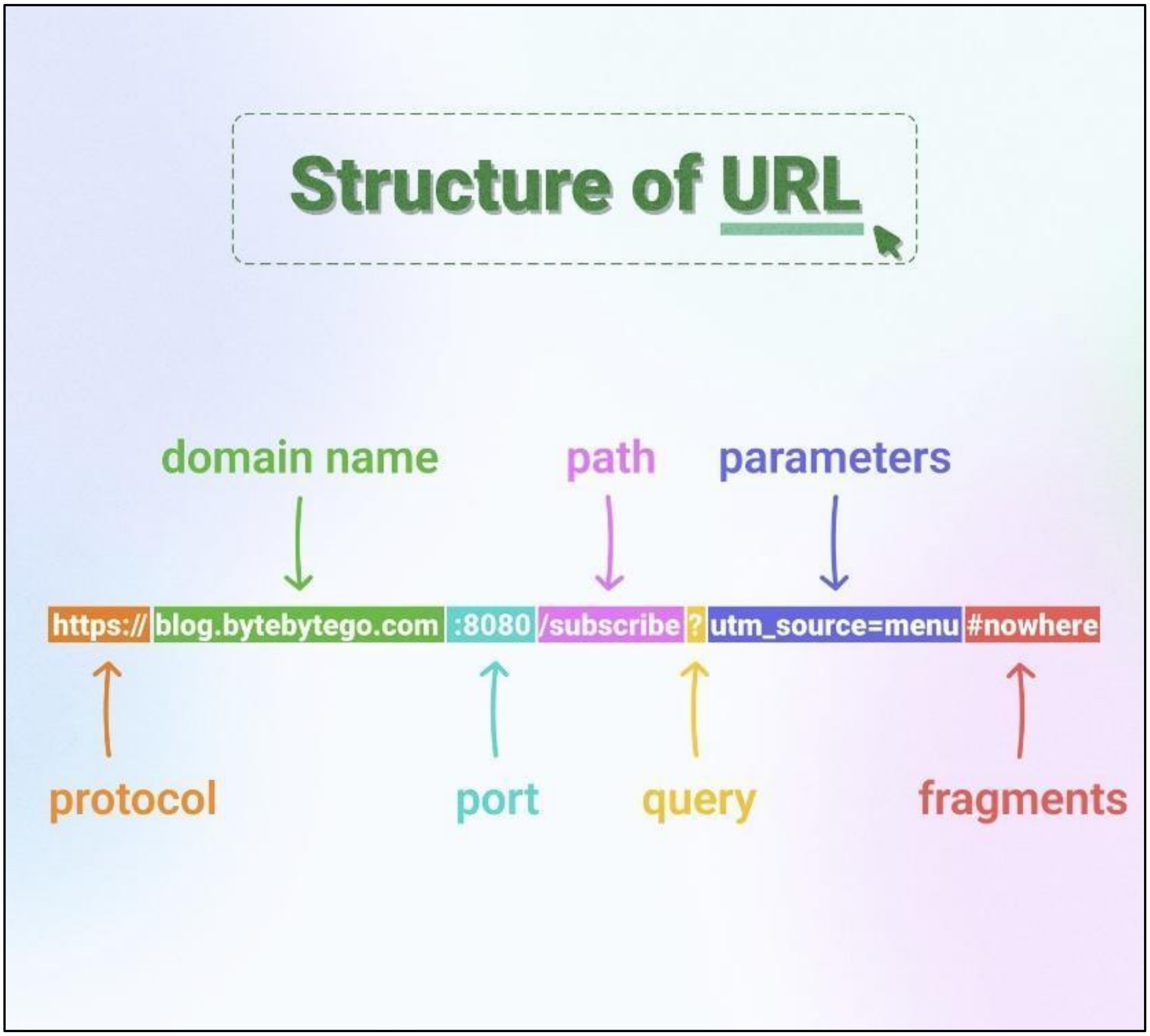
  1. Open JSON server in terminal by: `json-server --watch db.json`
  2. Go to postman and check it by Get request http://localhost:3000/

- **Result view**

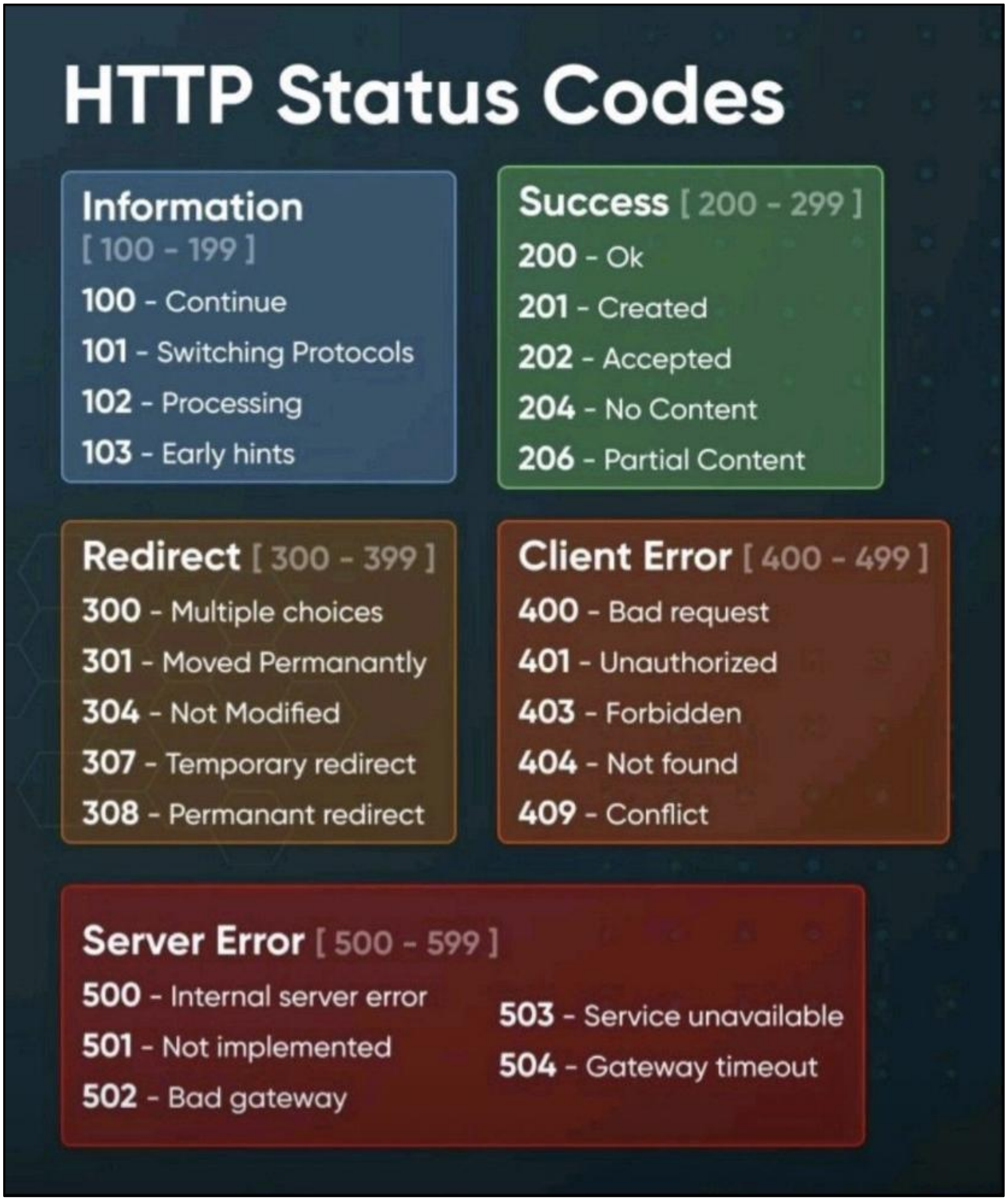

localhost:3000/names

```
[
  {
    "id": 1,
    "first_name": "sujit",
    "second_name": "sarker",
    "ageId": 1
  },
  {
    "id": 2,
    "first_name": "uzzal",
    "second_name": "sarker",
    "ageId": 2
  },
  {
    "id": 3,
    "first_name": "sharon",
    "second_name": "rahman",
    "ageId": 2
  },
  {
    "id": 4,
    "first_name": "Marufa",
    "second_name": "Akter",
    "ageId": 1
  },
  {
    "id": 5,
    "first_name": "Tawhid",
    "second_name": "Gias",
    "ageId": 1
  },
  {
    "id": 6,
    "first_name": "Nayeem",
    "second_name": "Hasan",
    "ageId": 1
  }
]
```

- **URL structure:**



- **HTTP Status code meaning:**



HTTP Status Codes

Information [100 - 199]
100 - Continue
101 - Switching Protocols
102 - Processing
103 - Early hints

Success [200 - 299]
200 - Ok
201 - Created
202 - Accepted
204 - No Content
206 - Partial Content

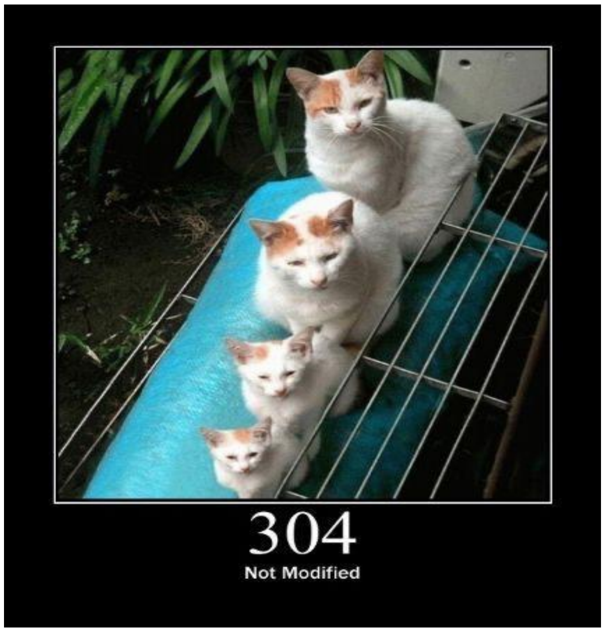
Redirect [300 - 399]
300 - Multiple choices
301 - Moved Permanantly
304 - Not Modified
307 - Temporary redirect
308 - Permanant redirect

Client Error [400 - 499]
400 - Bad request
401 - Unauthorized
403 - Forbidden
404 - Not found
409 - Conflict

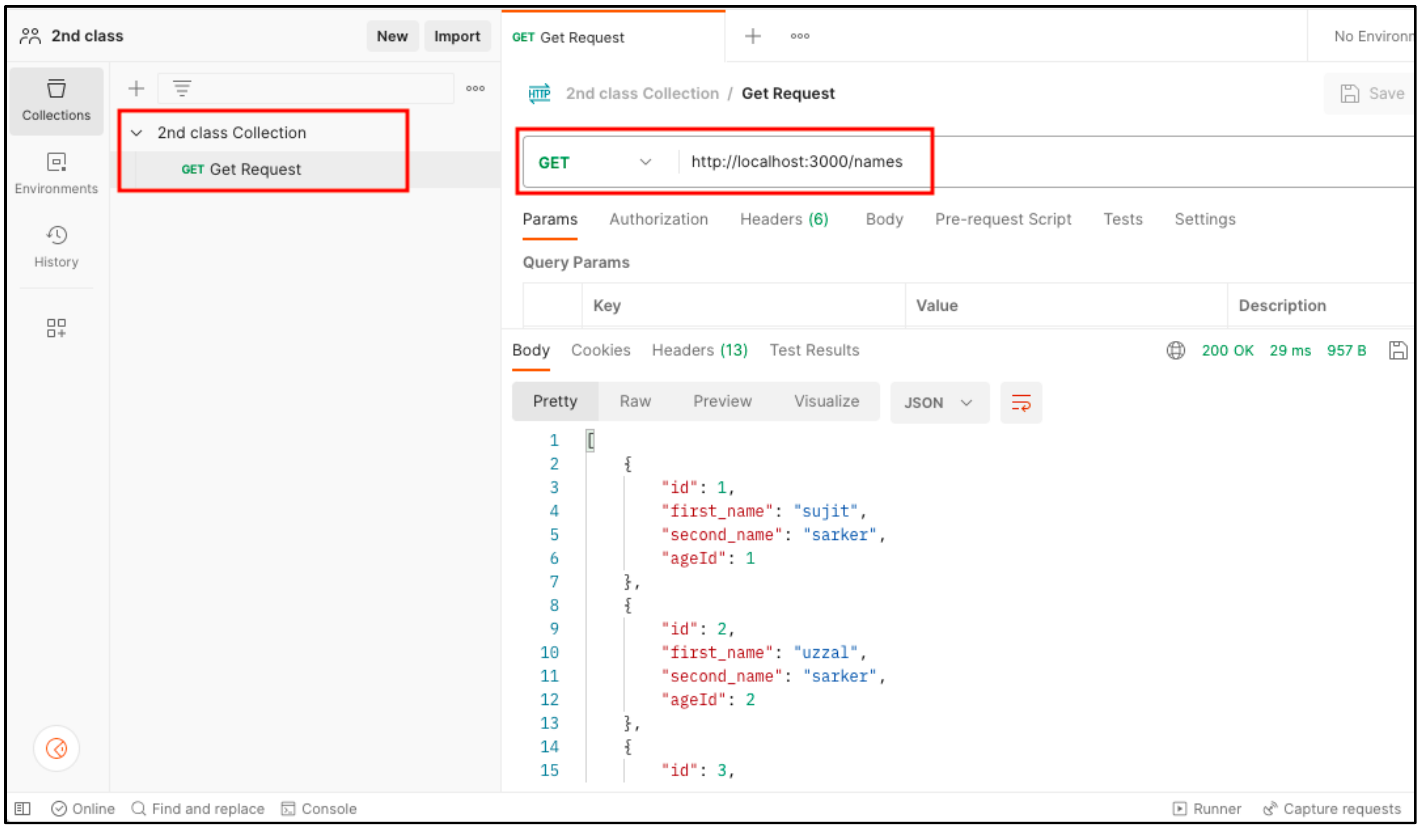
Server Error [500 - 599]
500 - Internal server error
501 - Not implemented
502 - Bad gateway
503 - Service unavailable
504 - Gateway timeout

Server Error/Response Code with **Examples:**

500- Internal Server Error	**404**- Not Found	**403**- Forbidden
401- Unauthorized	**400**- Bad Request	**304**- Not modified
300- Multiple Choices	**200**- OK	**204**-No Content

400 Bad Request	200 OK
300 Multiple Choices	304 Not Modified
204 No Content	

- **Get Method:**

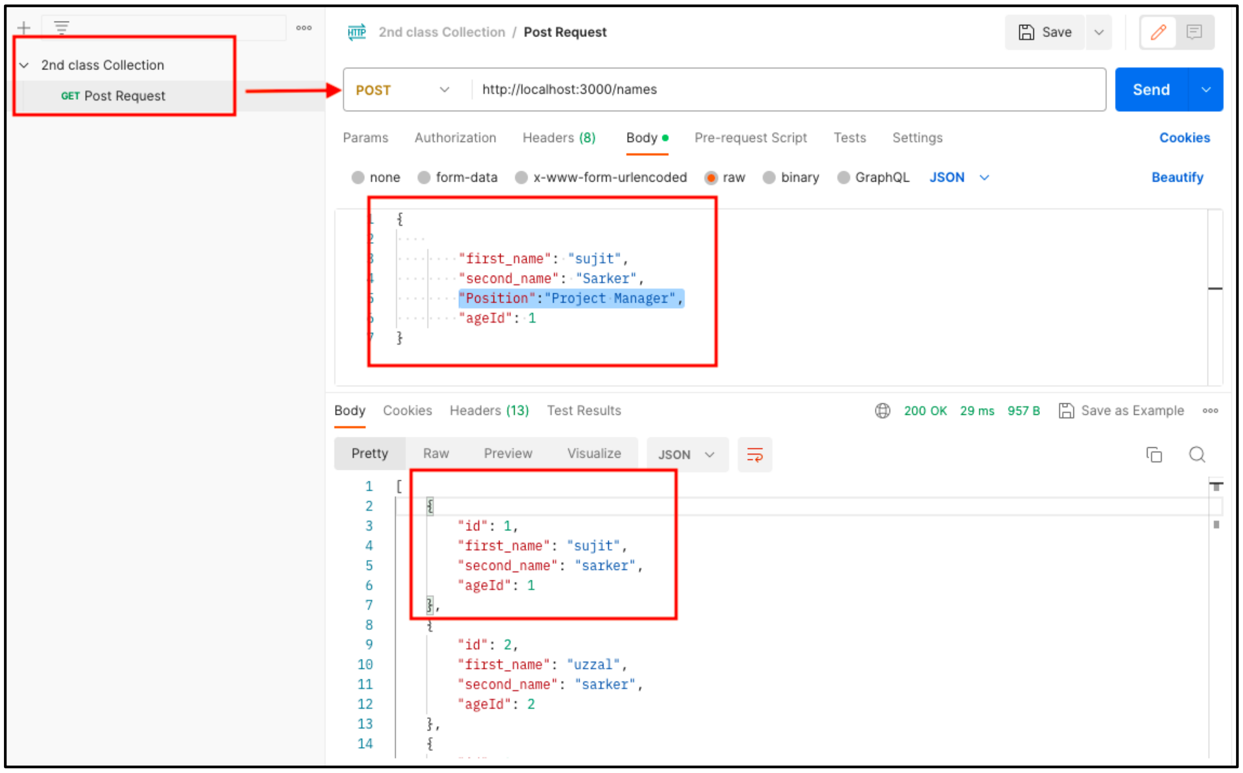


```
GET http://localhost:3000/names
```

```
[
    {
        "id": 1,
        "first_name": "sujit",
        "second_name": "sarker",
        "ageId": 1
    },
    {
        "id": 2,
        "first_name": "uzzal",
        "second_name": "sarker",
        "ageId": 2
    },
    {
        "id": 3,
```

“Postman Documentation”

- **Post Method:**



2nd class Collection / Post Request

GET Post Request

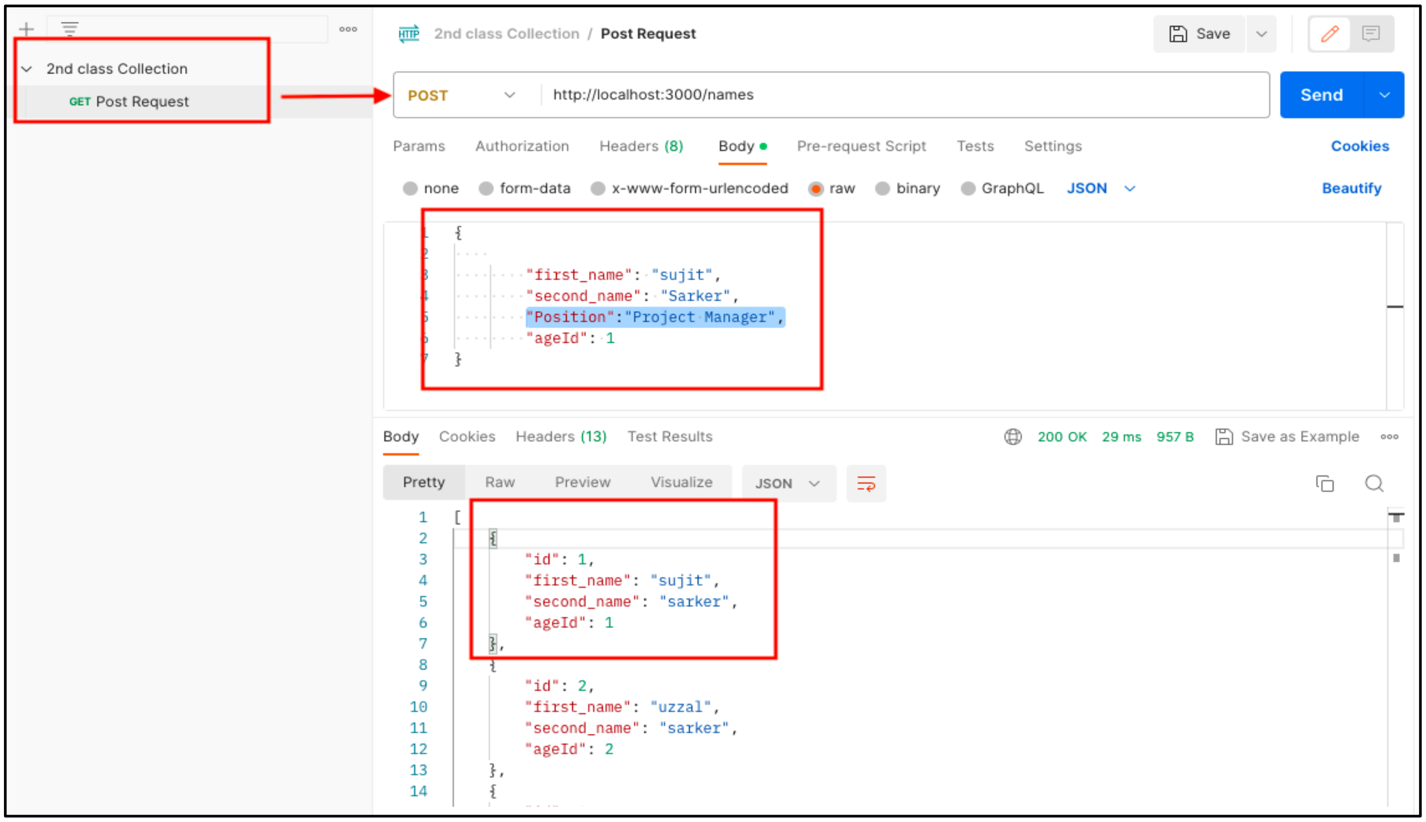
POST http://localhost:3000/names

Body: raw, JSON

```
{

    "first_name": "sujit",
    "second_name": "Sarker",
    "Position":"Project Manager",
    "ageId": 1
}
```

Body — 200 OK 29 ms 957 B

```
[
    {
        "id": 1,
        "first_name": "sujit",
        "second_name": "sarker",
        "ageId": 1
    },
    {
        "id": 2,
        "first_name": "uzzal",
        "second_name": "sarker",
        "ageId": 2
    },
    {
```



2nd class Collection / Post Request

GET Post Request

POST http://localhost:3000/names

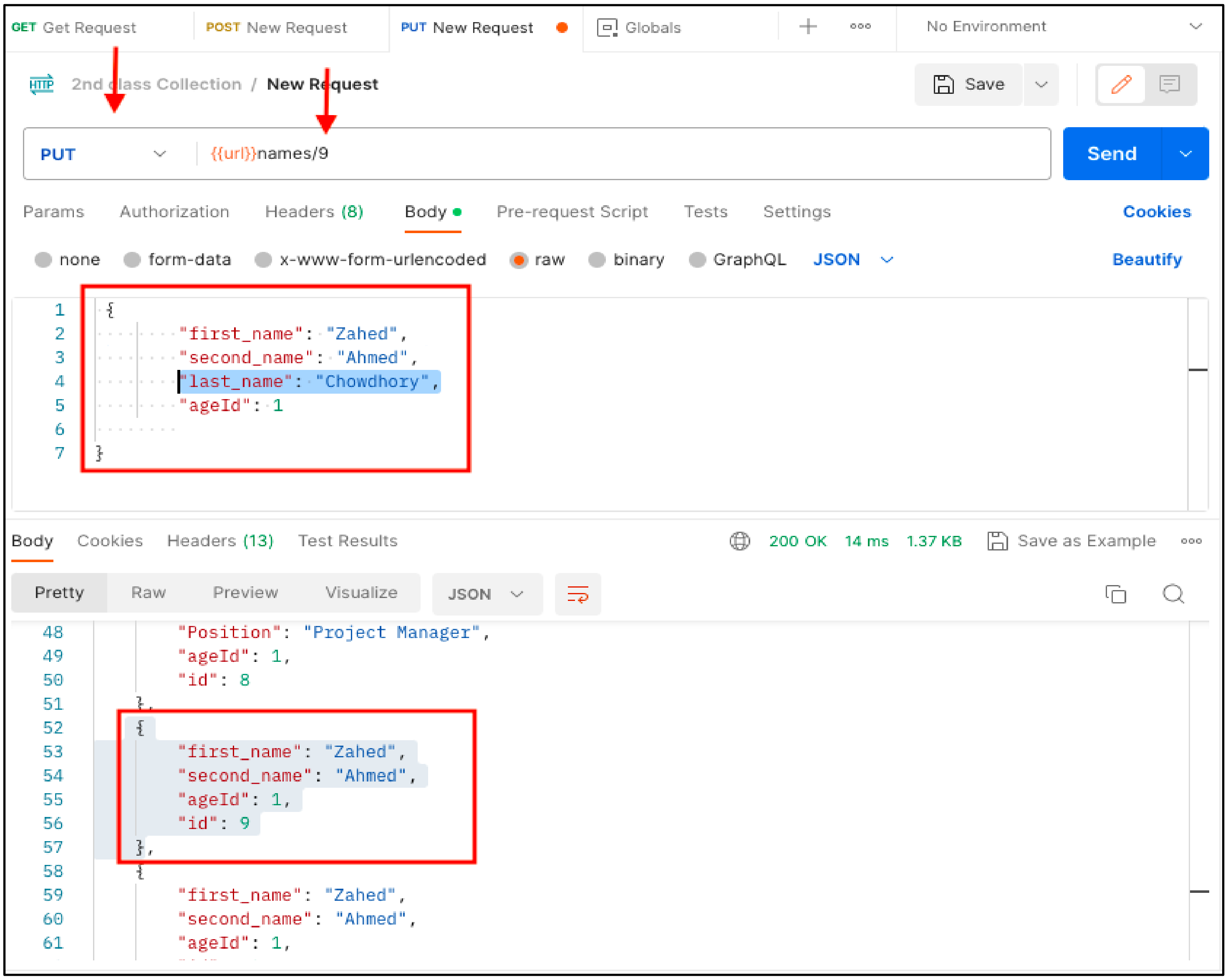
Body: raw, JSON

```
{

    "first_name": "sujit",
    "second_name": "Sarker",
    "Position":"Project Manager",
    "ageId": 1
}
```

Body — 200 OK 29 ms 957 B

```
[
    {
        "id": 1,
        "first_name": "sujit",
        "second_name": "sarker",
        "ageId": 1
    },
    {
        "id": 2,
        "first_name": "uzzal",
        "second_name": "sarker",
        "ageId": 2
    },
    {
```

“Postman Documentation”



GET Get Request | POST New Request | PUT New Request | Globals | No Environment

2nd class Collection / New Request

Save

PUT {{url}}names/9

Send

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

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

```
{
    "first_name": "Zahed",
    "second_name": "Ahmed",
    "last_name": "Chowdhory",
    "ageId": 1

}
```

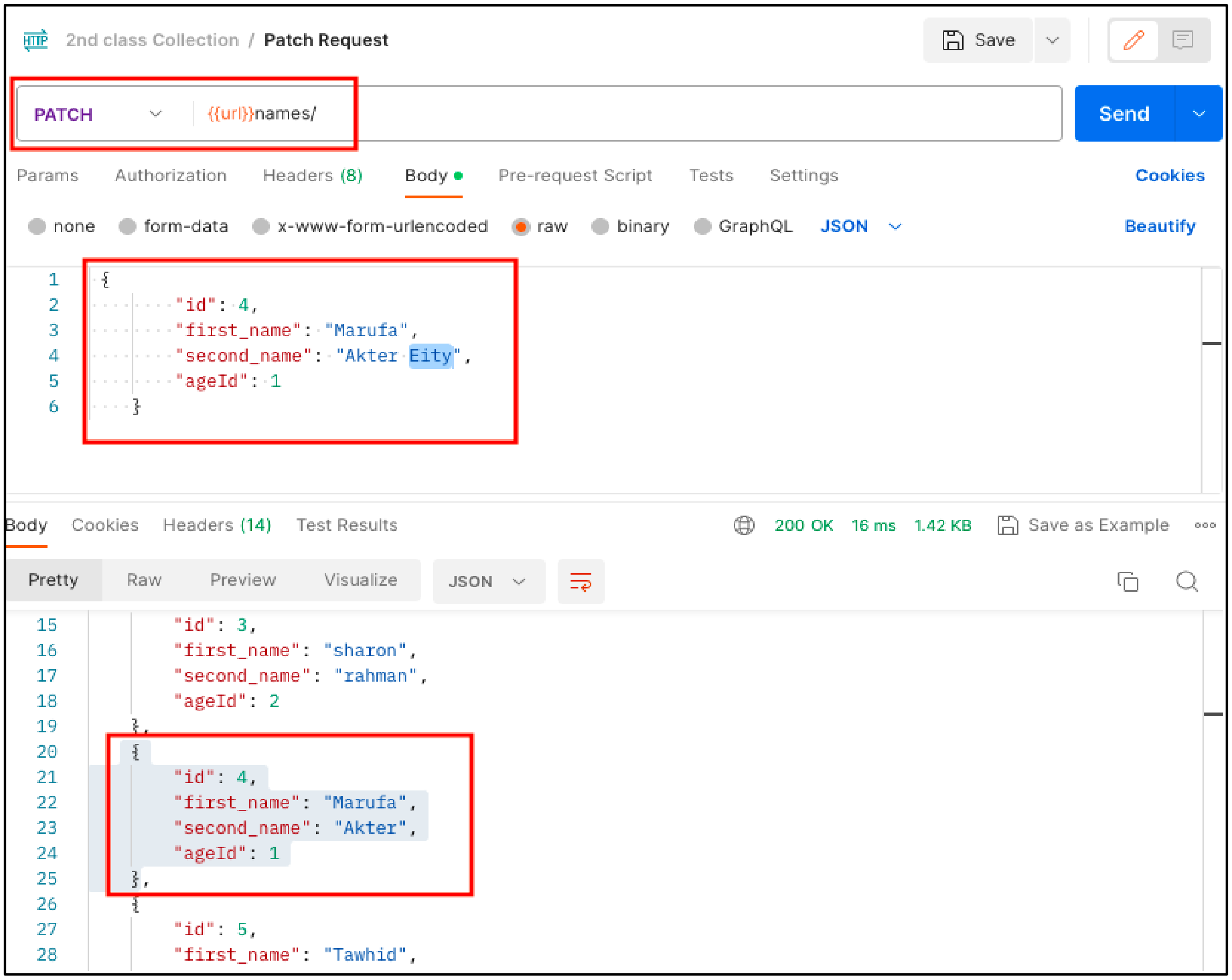
Body | Cookies | Headers (13) | Test Results | 200 OK | 14 ms | 1.37 KB | Save as Example

Pretty | Raw | Preview | Visualize | JSON

```
        "Position": "Project Manager",
        "ageId": 1,
        "id": 8
    },
    {
        "first_name": "Zahed",
        "second_name": "Ahmed",
        "ageId": 1,
        "id": 9
    },
    {
        "first_name": "Zahed",
        "second_name": "Ahmed",
        "ageId": 1,
```

“Postman Documentation”

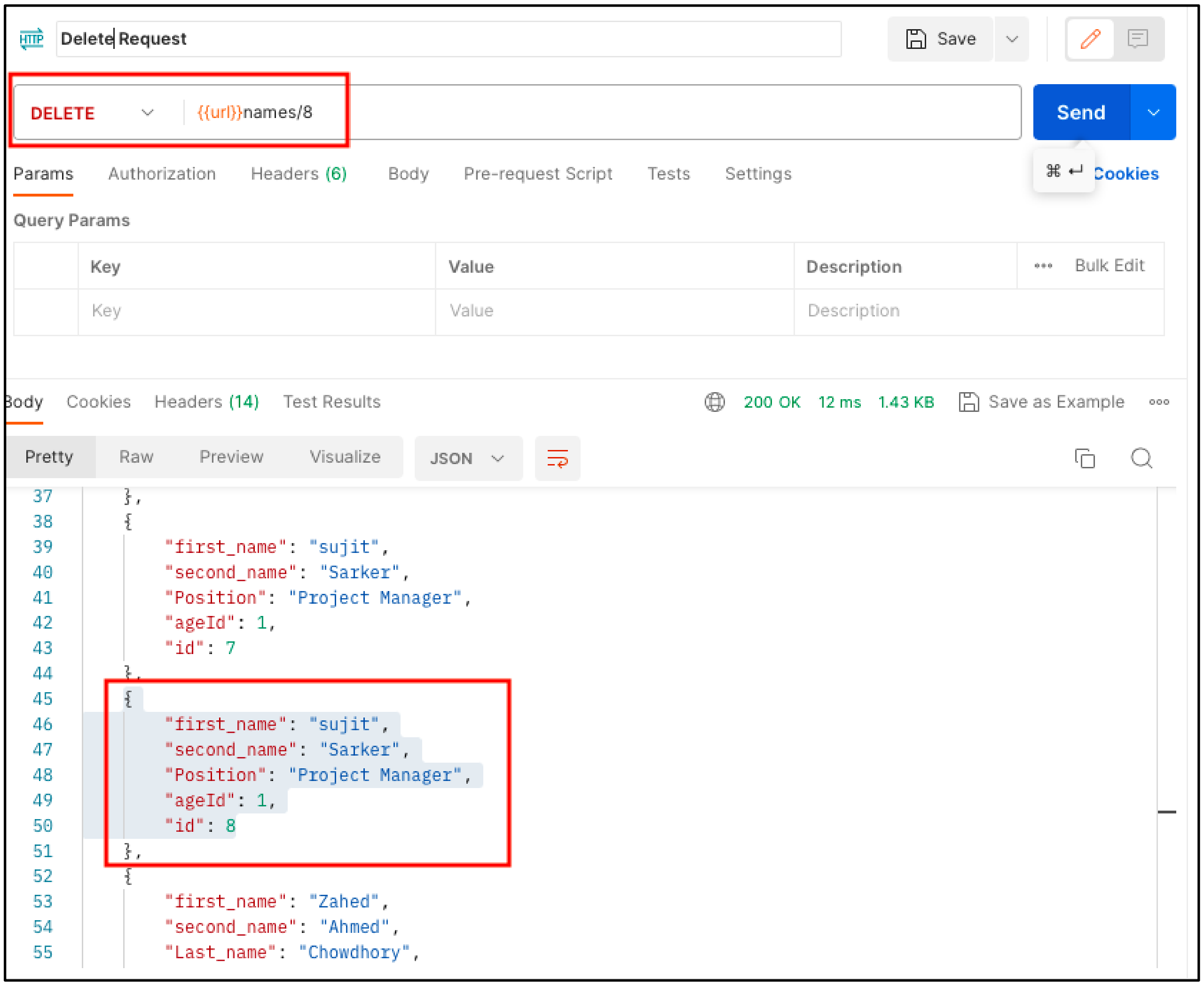
- **Patch Method:**



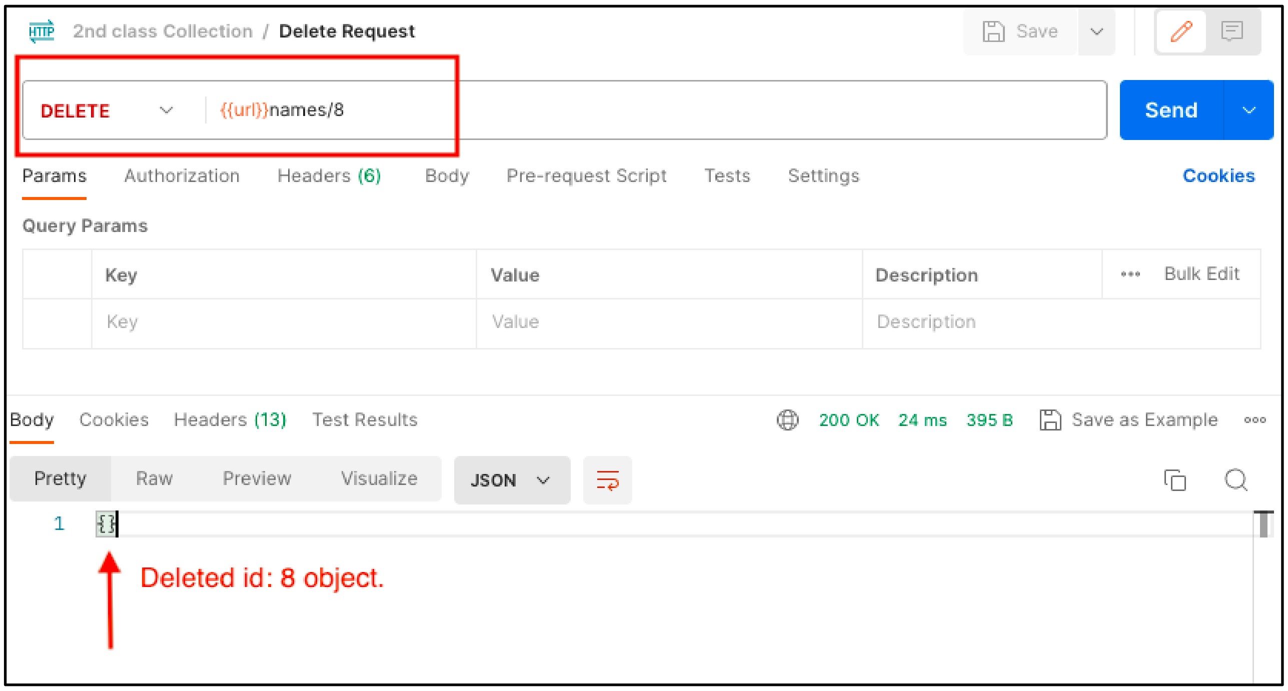
```
{
    "id": 4,
    "first_name": "Marufa",
    "second_name": "Akter Eity",
    "ageId": 1
}
```

```
        "id": 3,
        "first_name": "sharon",
        "second_name": "rahman",
        "ageId": 2
    },
    {
        "id": 4,
        "first_name": "Marufa",
        "second_name": "Akter",
        "ageId": 1
    },
    {
        "id": 5,
        "first_name": "Tawhid",
```

- **Delete Method:**



```
    },
    {
        "first_name": "sujit",
        "second_name": "Sarker",
        "Position": "Project Manager",
        "ageId": 1,
        "id": 7
    },
    {
        "first_name": "sujit",
        "second_name": "Sarker",
        "Position": "Project Manager",
        "ageId": 1,
        "id": 8
    },
    {
        "first_name": "Zahed",
        "second_name": "Ahmed",
        "Last_name": "Chowdhory",
```


2nd class Collection / Delete Request

DELETE {{url}}names/8

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (13) Test Results 200 OK 24 ms 395 B Save as Example

Pretty Raw Preview Visualize JSON

```
{}
```

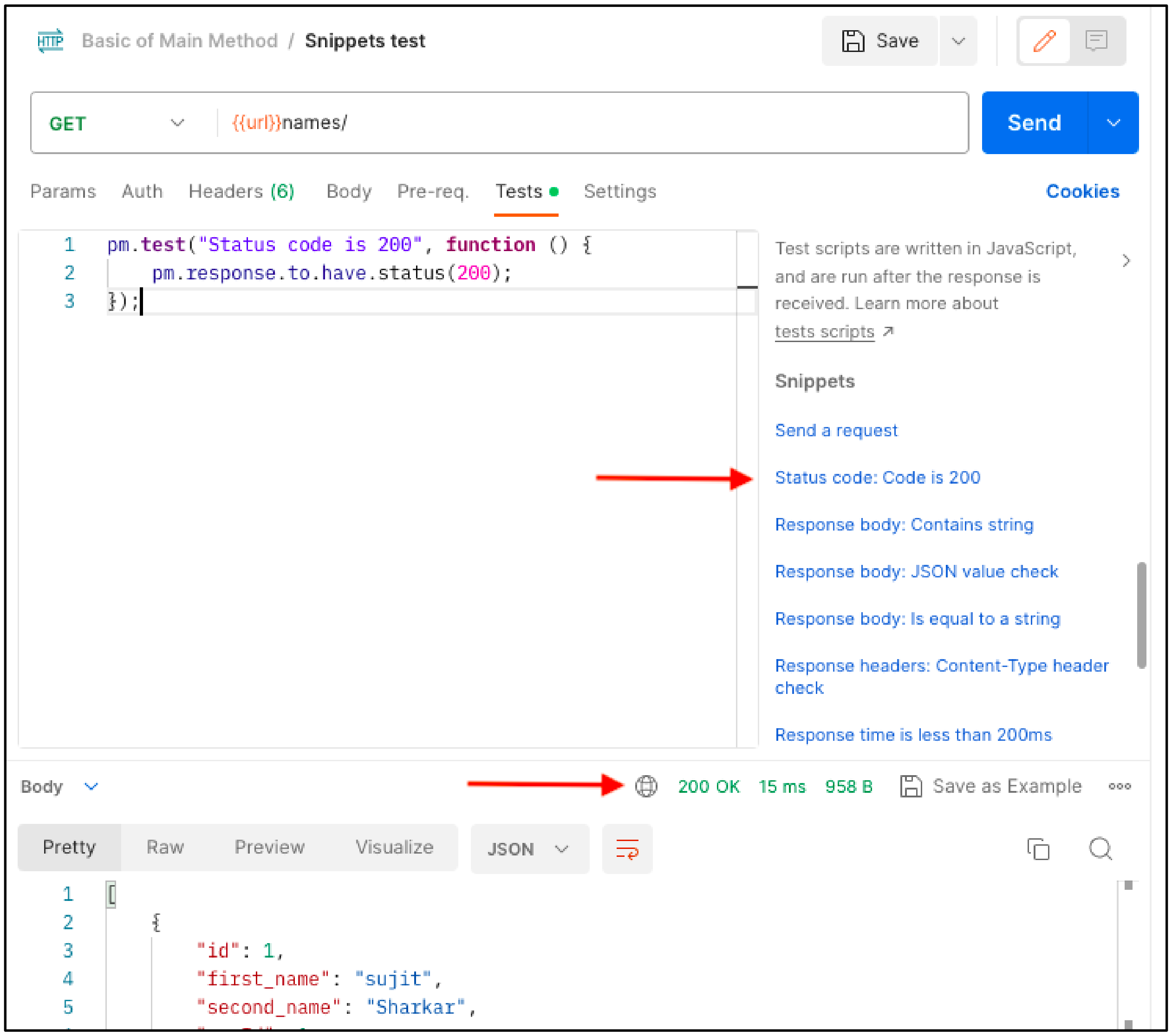
Deleted id: 8 object.

- **Test script practice using sample snippets:**

➢ **Apply Process:**
Select a **Request**
Select the **Tests** section
Click on any Script [**Snippets**]
Script will be available on the Script body
Modify the Script if needed
Click on the **Send** Button
Let’s check the **Result** on Postman

“Postman Documentation”

Script I- Status Code: Code is 200



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

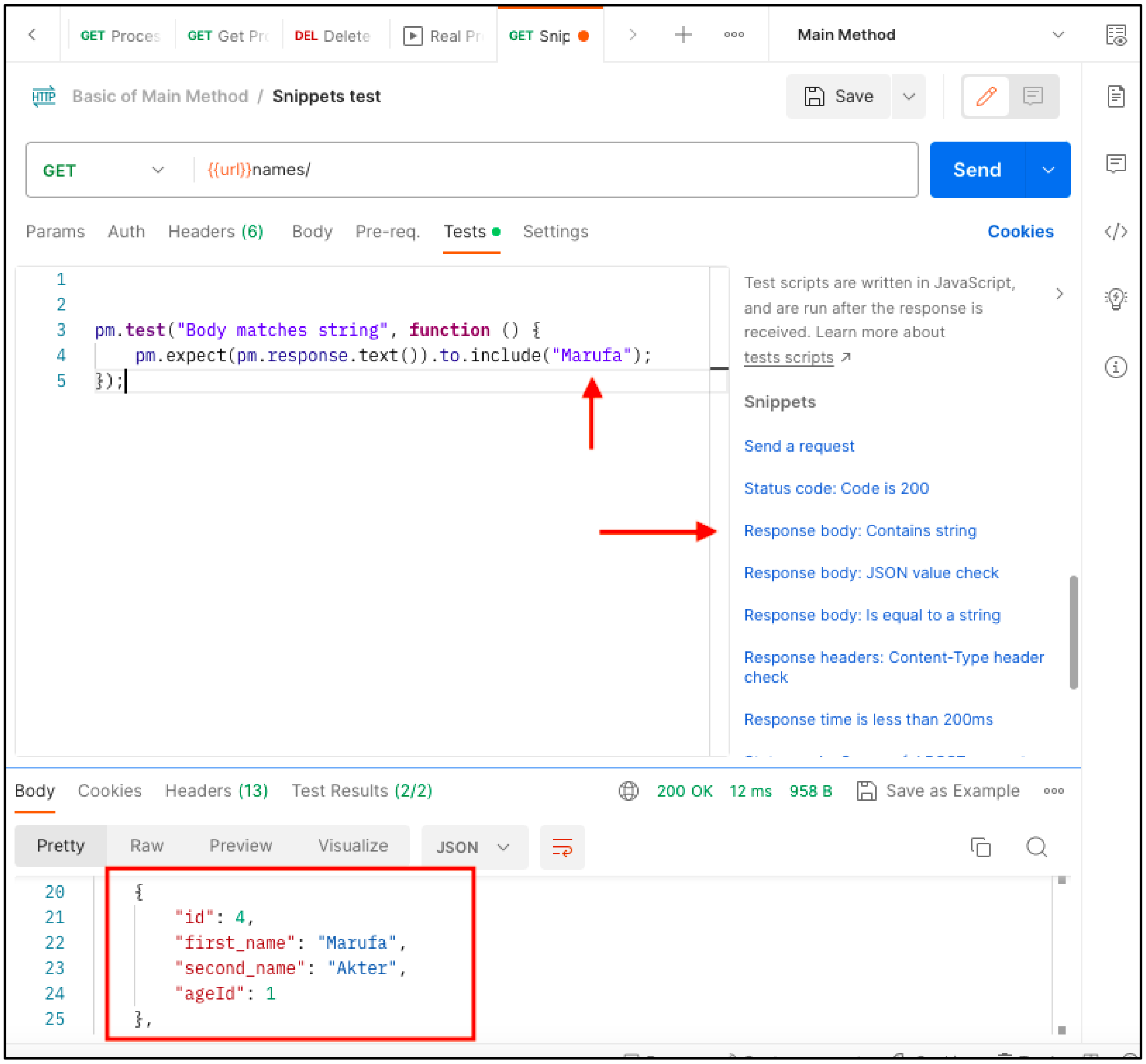
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

- Send a request
- Status code: Code is 200
- Response body: Contains string
- Response body: JSON value check
- Response body: Is equal to a string
- Response headers: Content-Type header check
- Response time is less than 200ms

```
[
    {
        "id": 1,
        "first_name": "sujit",
        "second_name": "Sharkar",
```

“Postman Documentation”



```
pm.test("Body matches string", function () {
    pm.expect(pm.response.text()).to.include("Marufa");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

Send a request

Status code: Code is 200

Response body: Contains string

Response body: JSON value check

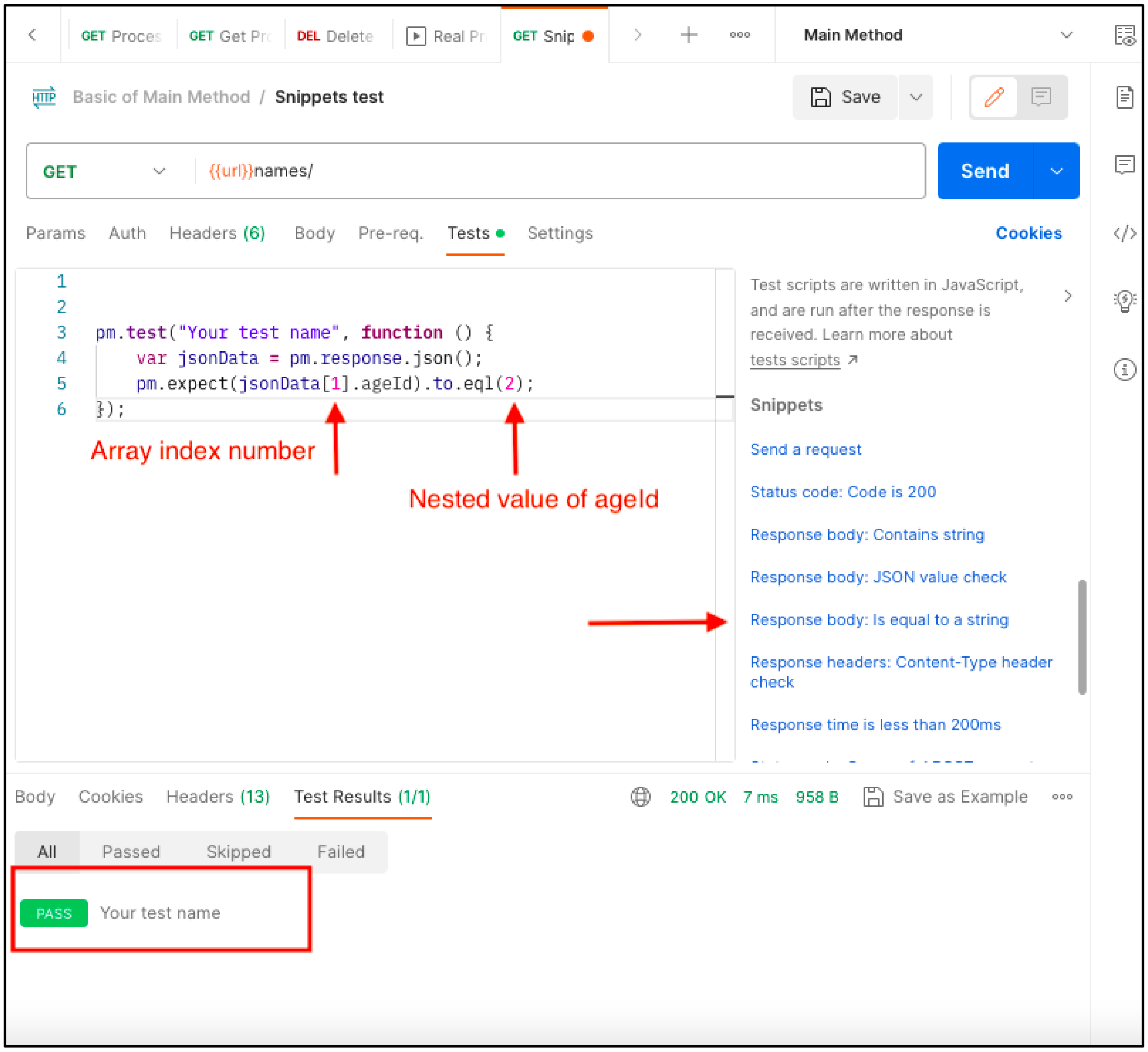
Response body: Is equal to a string

Response headers: Content-Type header check

Response time is less than 200ms

```
{
    "id": 4,
    "first_name": "Marufa",
    "second_name": "Akter",
    "ageId": 1
},
```

Script III- Response Body: JSON Value Check

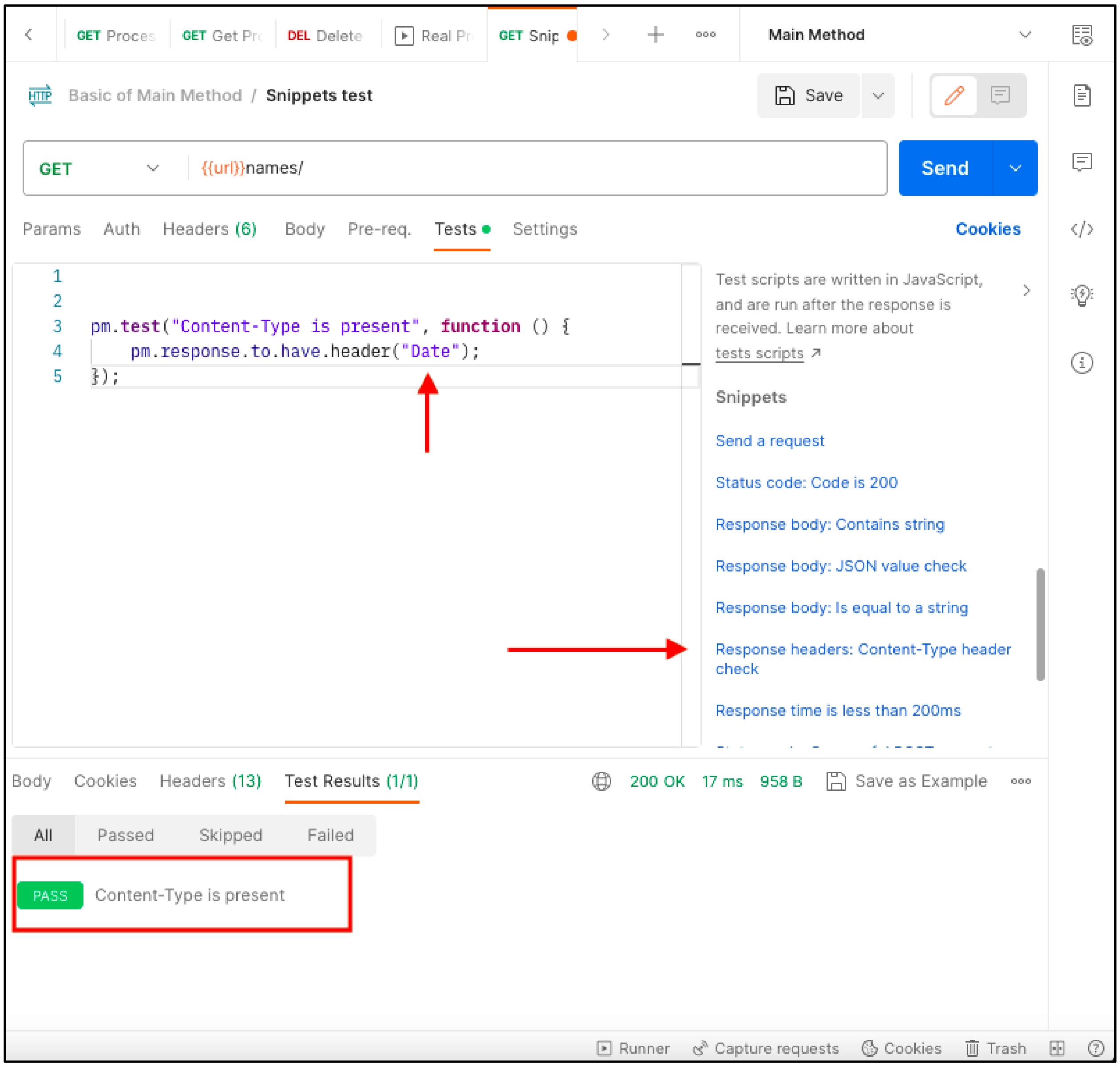


```
pm.test("Your test name", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData[1].ageId).to.eql(2);
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

PASS Your test name

Script IV- Response Headers: Content-type Header Check



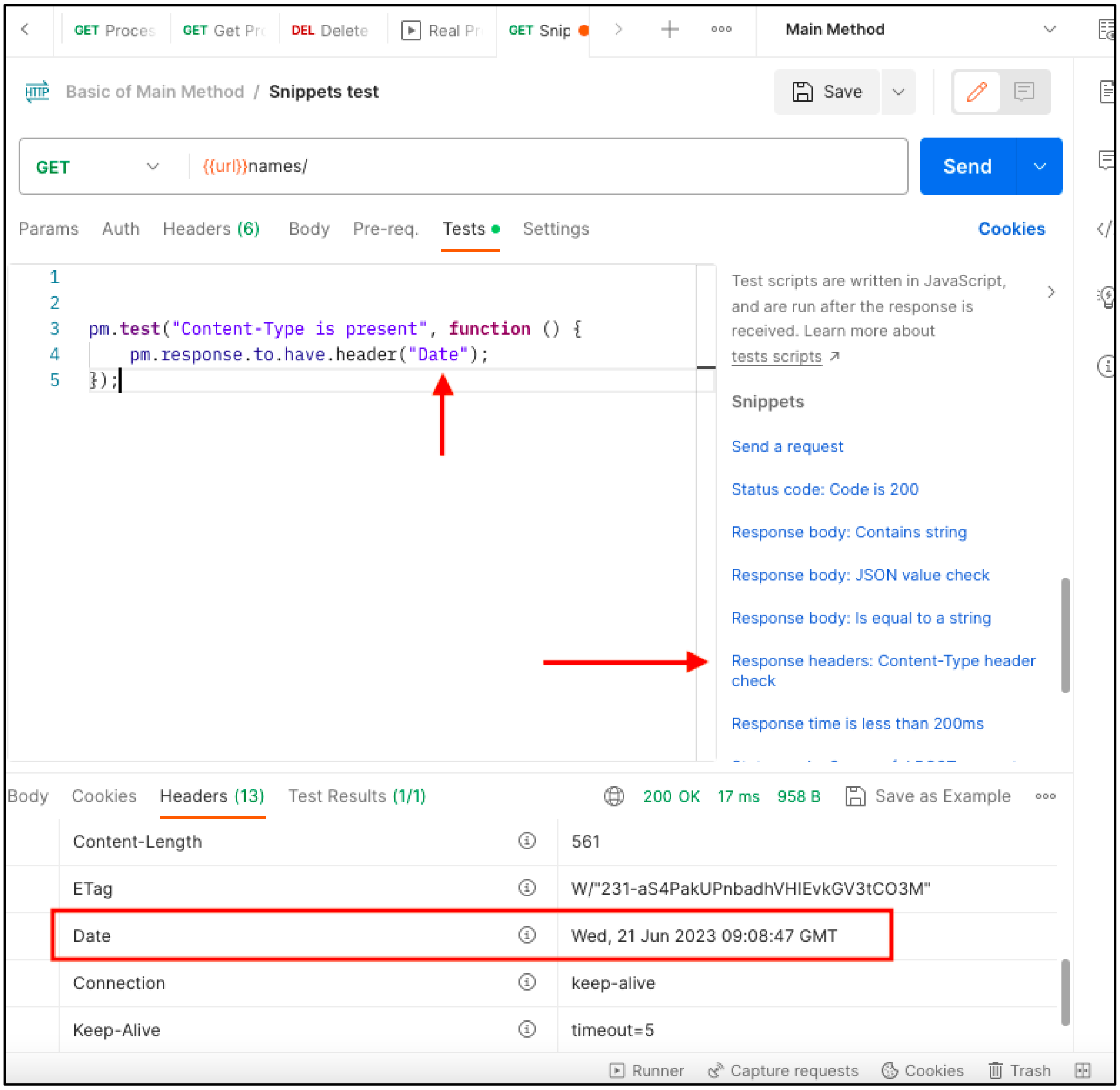
```
pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Date");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets
Send a request
Status code: Code is 200
Response body: Contains string
Response body: JSON value check
Response body: Is equal to a string
Response headers: Content-Type header check
Response time is less than 200ms
Body
Cookies
Headers (13)
Test Results (1/1)
200 OK
17 ms
958 B
Save as Example
All
Passed
Skipped
Failed

PASS Content-Type is present

“Postman Documentation”



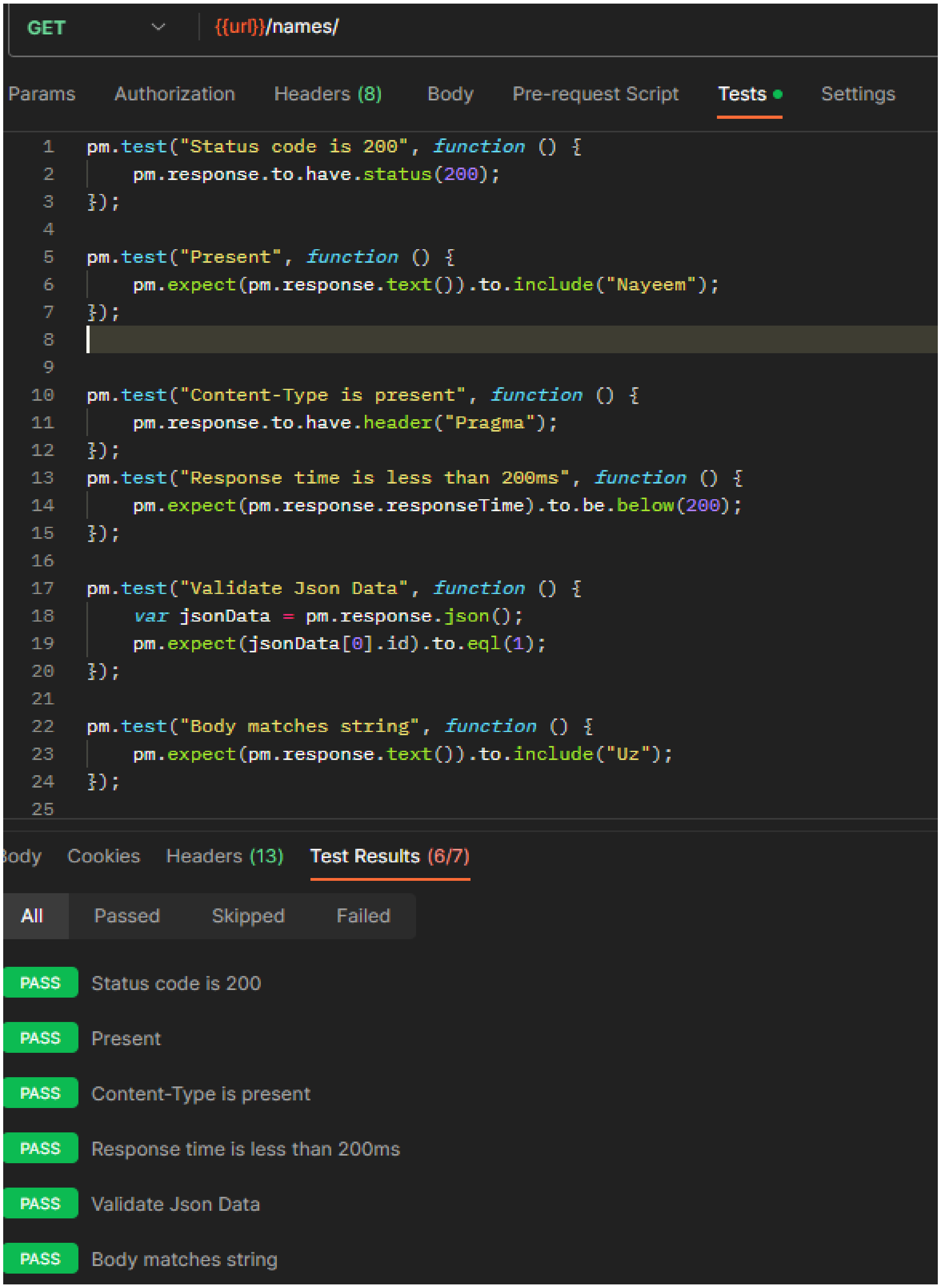
```
pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Date");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

- Send a request
- Status code: Code is 200
- Response body: Contains string
- Response body: JSON value check
- Response body: Is equal to a string
- Response headers: Content-Type header check
- Response time is less than 200ms

Content-Length	561
ETag	W/"231-aS4PakUPnbadhVHlEvkGV3tCO3M"
Date	Wed, 21 Jun 2023 09:08:47 GMT
Connection	keep-alive
Keep-Alive	timeout=5



GET {{url}}/names/

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

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

pm.test("Present", function () {
    pm.expect(pm.response.text()).to.include("Nayeem");
});


pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Pragma");
});
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});

pm.test("Validate Json Data", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData[0].id).to.eql(1);
});

pm.test("Body matches string", function () {
    pm.expect(pm.response.text()).to.include("Uz");
});

```

Body Cookies Headers (13) Test Results (6/7)

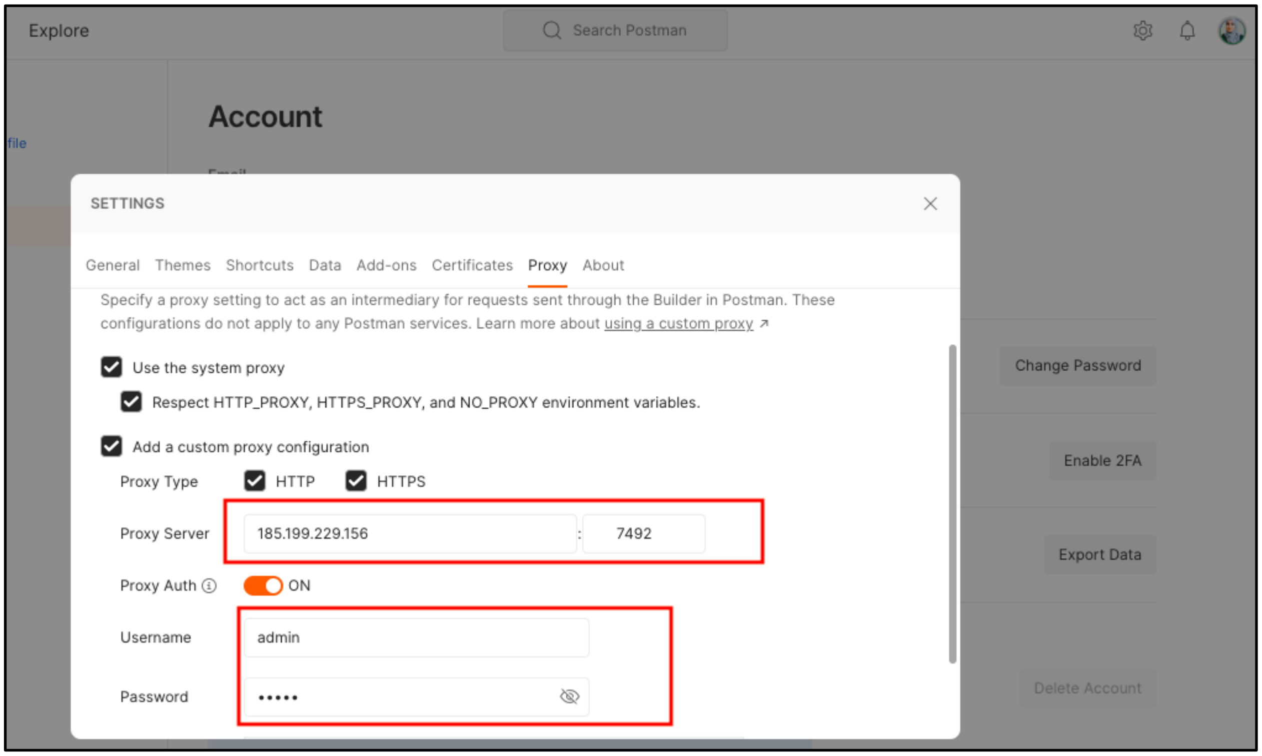
All Passed Skipped Failed

```
PASS  Status code is 200
PASS  Present
PASS  Content-Type is present
PASS  Response time is less than 200ms
PASS  Validate Json Data
PASS  Body matches string
```

“Postman Documentation”

➢ **Configure a Proxy Server**

Go to Settings
Select the Proxy tab
Let’s follow the steps according to attached image



Specify a proxy setting to act as an intermediary for requests sent through the Builder in Postman. These configurations do not apply to any Postman services. Learn more about using a custom proxy ↗

- ✓ Use the system proxy
  - ✓ Respect HTTP_PROXY, HTTPS_PROXY, and NO_PROXY environment variables.
- ✓ Add a custom proxy configuration

Proxy Type: ✓ HTTP ✓ HTTPS
Proxy Server: 185.199.229.156 : 7492
Proxy Auth: ON
Username: admin
Password: •••••

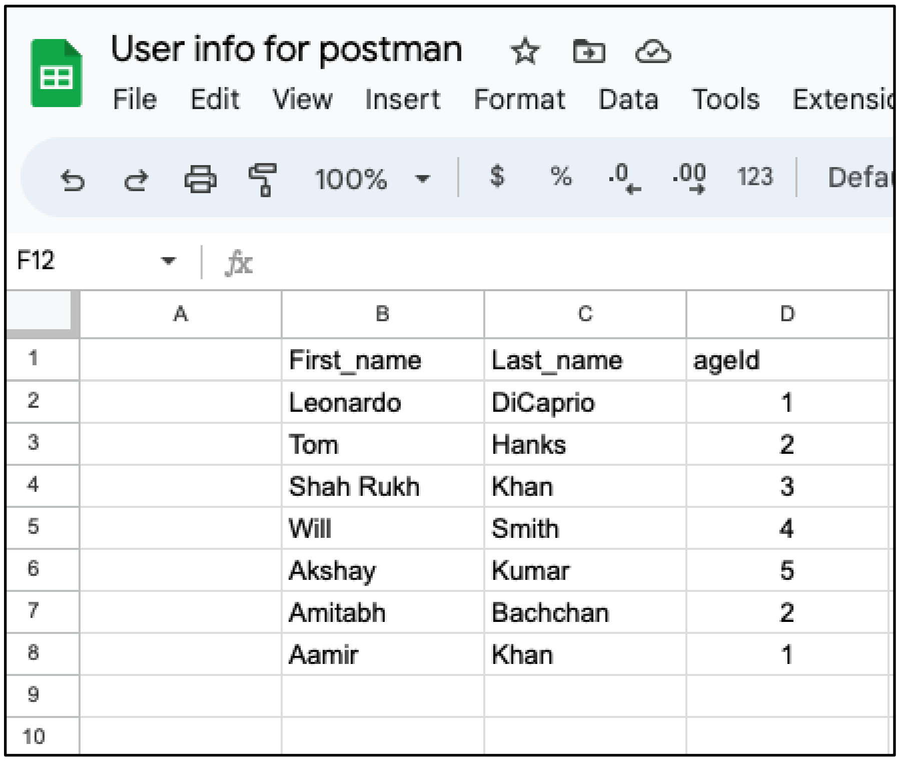
Utilize a CSV File to Insert Multiple Objects value

Apply Process:

Create a **.csv** File
Go to Postman & select the request. Ex- **POST** Request
Insert the proper **URL** (http://localhost:3000/names)
Write the following Script on **Body > Raw > JSON**

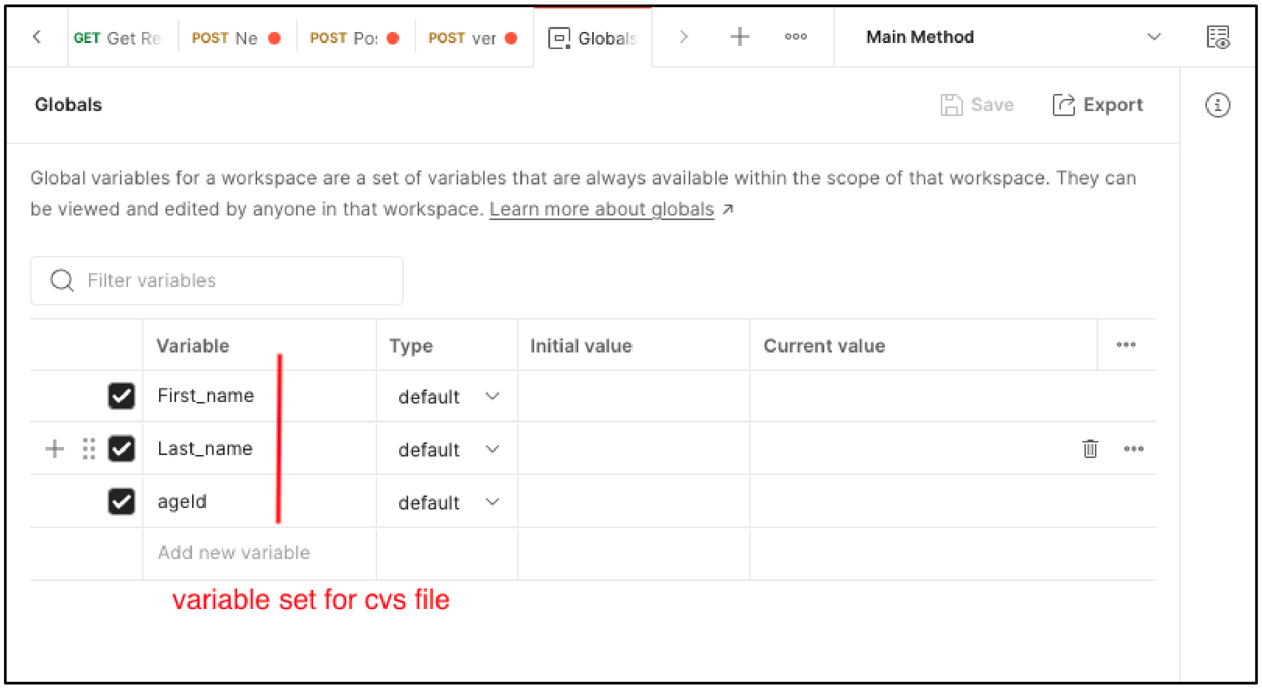
```
{
"First_name": "{{First_name}}",
"Last_name": "{{Last_name}}",
"ageId": {{ageId}}
}
```

Run the Collection and Put a Check Mark on the specific request
Select the CSV file and Check the **Preview**
Click on **Run Collection** Burton
Check the response on the following URL (http://localhost:3000/names) by **Get method.**



	A	B	C	D
1		First_name	Last_name	ageId
2		Leonardo	DiCaprio	1
3		Tom	Hanks	2
4		Shah Rukh	Khan	3
5		Will	Smith	4
6		Akshay	Kumar	5
7		Amitabh	Bachchan	2
8		Aamir	Khan	1
9				
10				

“Postman Documentation”

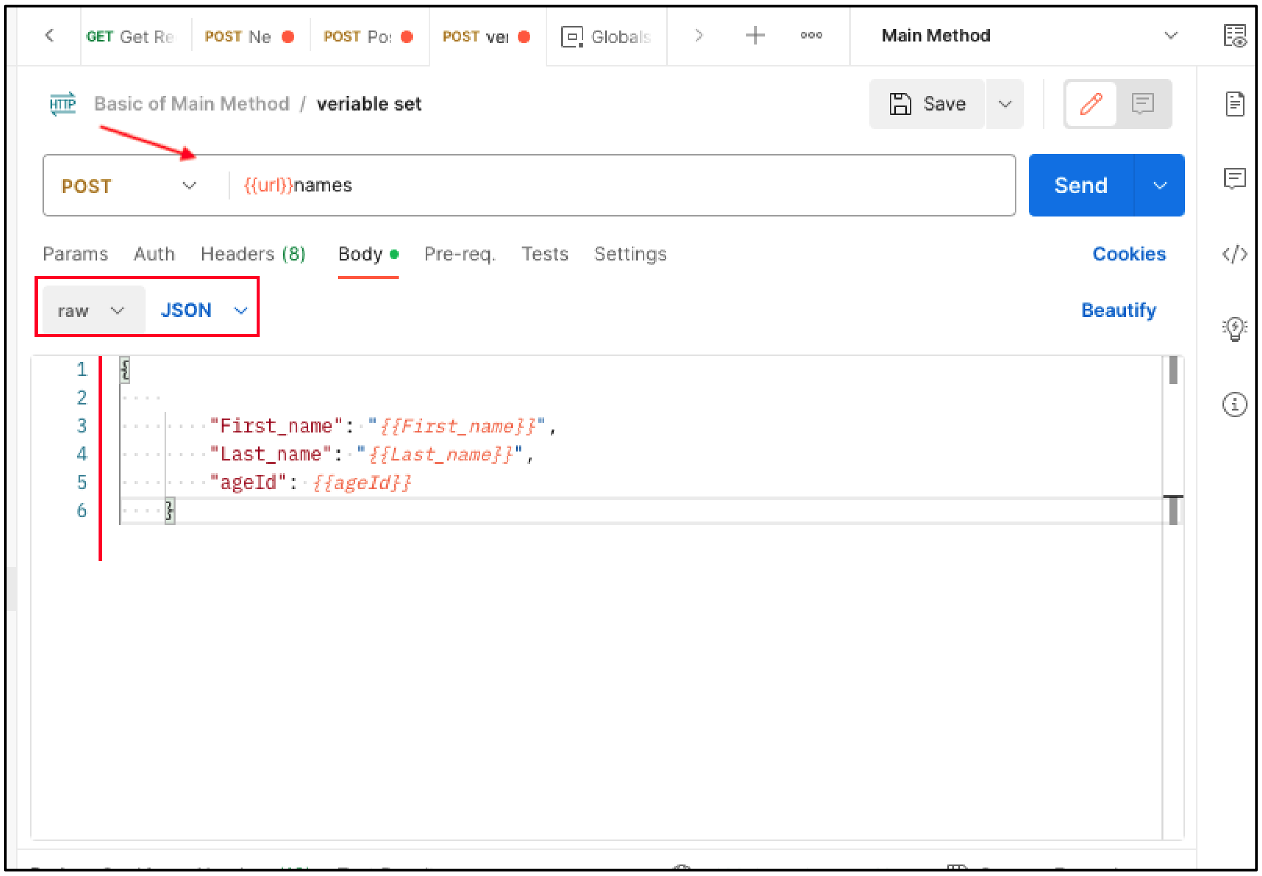


Globals

Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. Learn more about globals ↗

	Variable	Type	Initial value	Current value
✓	First_name	default		
✓	Last_name	default		
✓	ageId	default		
	Add new variable			

variable set for cvs file



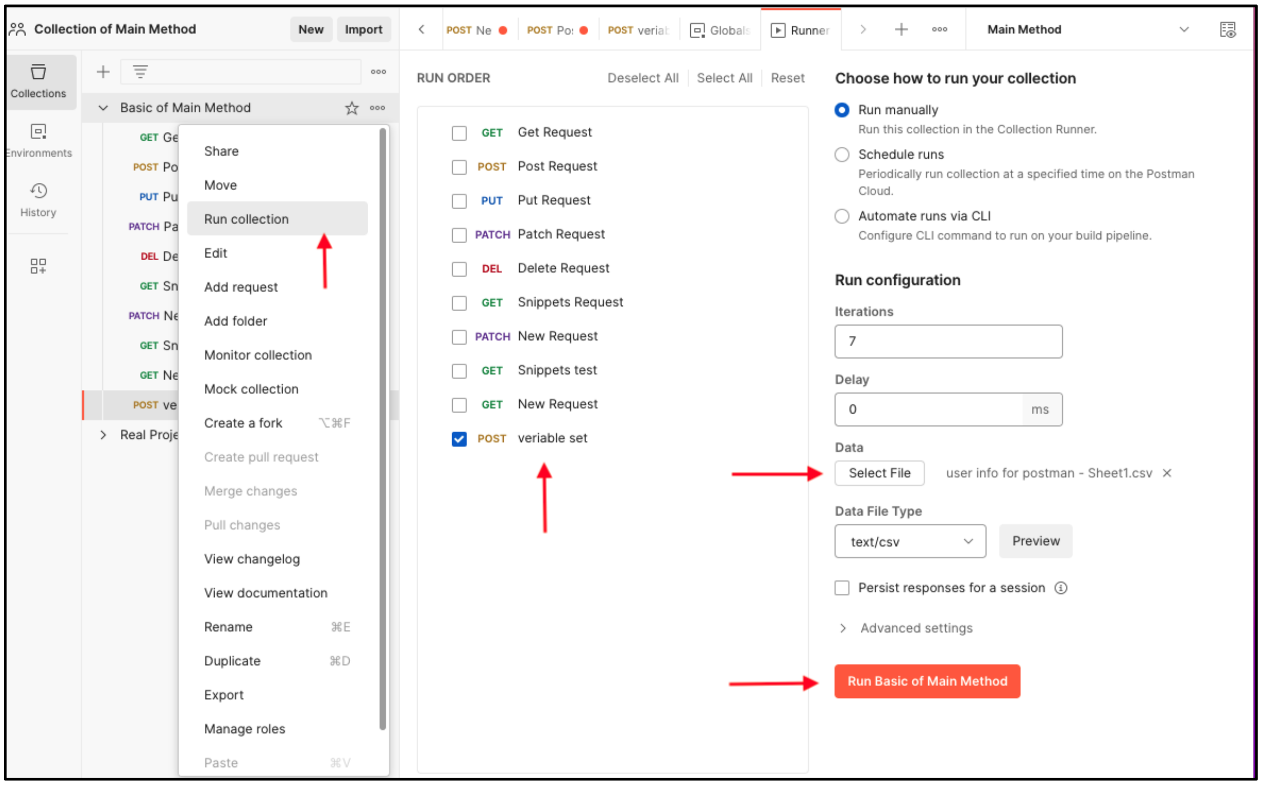
Basic of Main Method / veriable set

POST {{url}}names

Body: raw JSON

```
{
....
....|...."First_name": "{{First_name}}",
....|...."Last_name": "{{Last_name}}",
....|...."ageId": {{ageId}}
....}
```

“Postman Documentation”



- [ ] GET Get Request
- [ ] POST Post Request
- [ ] PUT Put Request
- [ ] PATCH Patch Request
- [ ] DEL Delete Request
- [ ] GET Snippets Request
- [ ] PATCH New Request
- [ ] GET Snippets test
- [ ] GET New Request
- [x] POST veriable set

Choose how to run your collection

[✓ Run manually] — Run this collection in the Collection Runner.

Schedule runs — Periodically run collection at a specified time on the Postman Cloud.

Automate runs via CLI — Configure CLI command to run on your build pipeline.

Run configuration

Iterations: 7

Delay: 0 ms

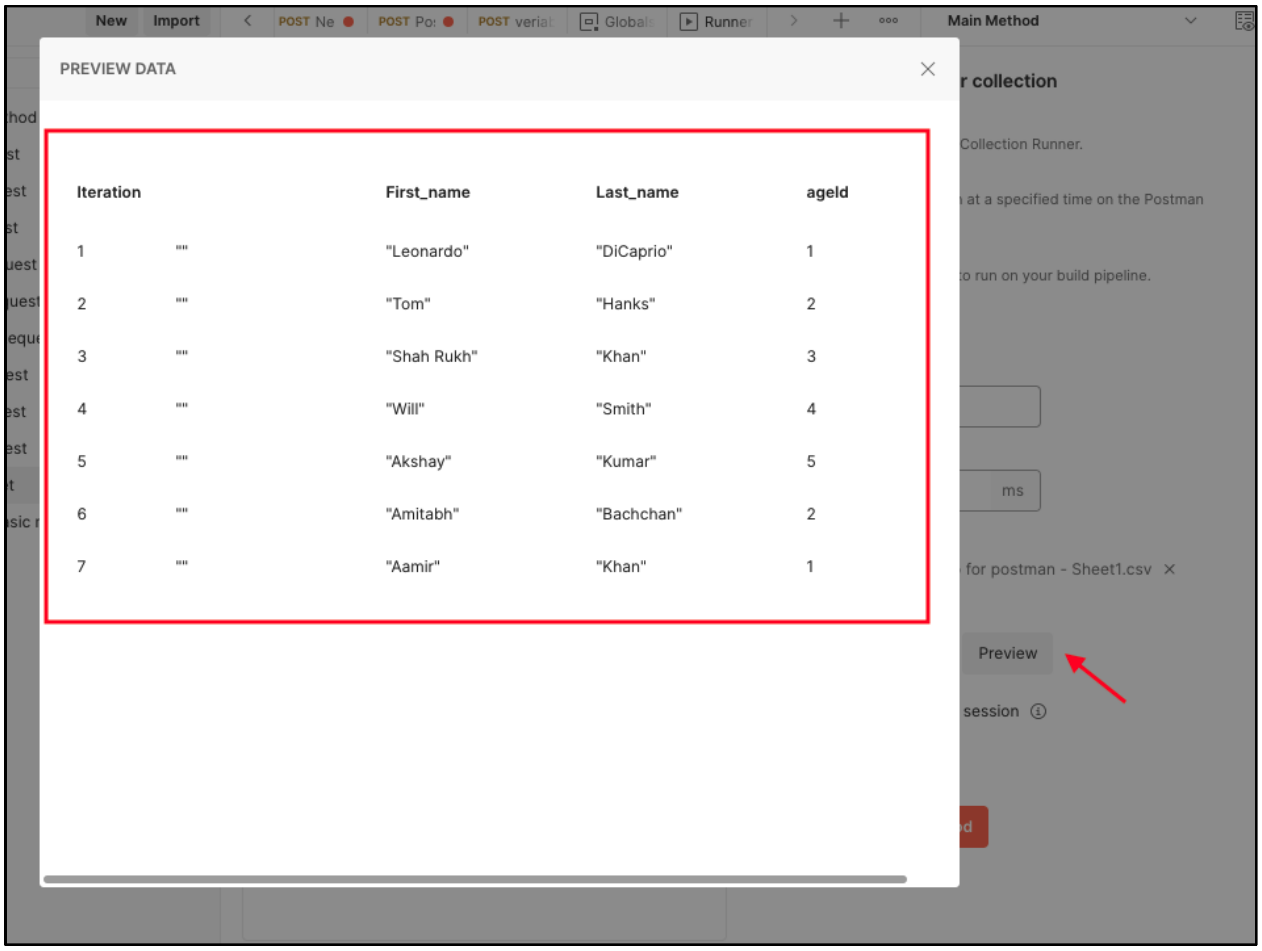
Data: Select File — user info for postman - Sheet1.csv

Data File Type: text/csv — Preview

- [ ] Persist responses for a session

Advanced settings

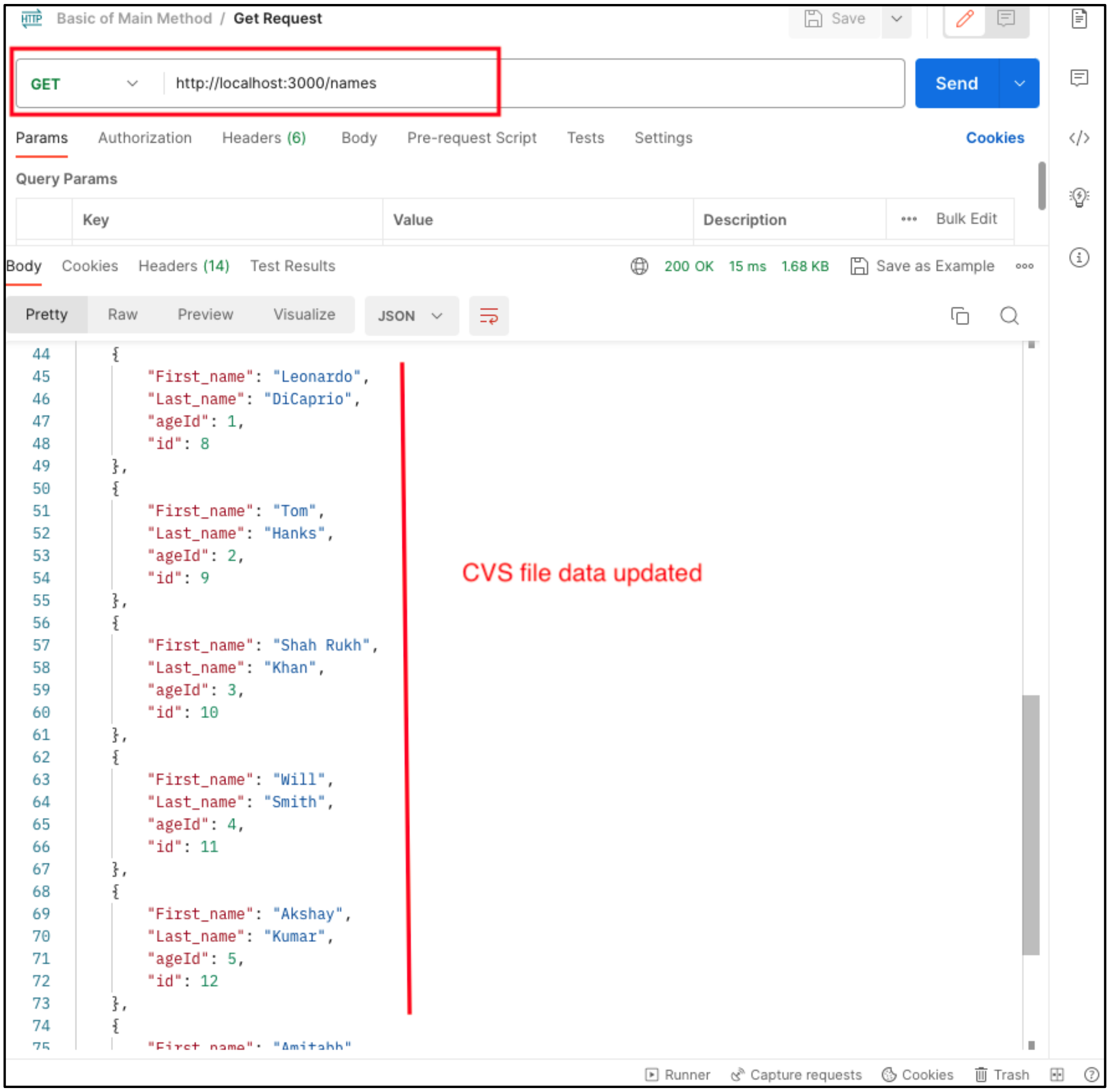
Run Basic of Main Method



PREVIEW DATA

Iteration		First_name	Last_name	ageId
1	""	"Leonardo"	"DiCaprio"	1
2	""	"Tom"	"Hanks"	2
3	""	"Shah Rukh"	"Khan"	3
4	""	"Will"	"Smith"	4
5	""	"Akshay"	"Kumar"	5
6	""	"Amitabh"	"Bachchan"	2
7	""	"Aamir"	"Khan"	1

“Postman Documentation”

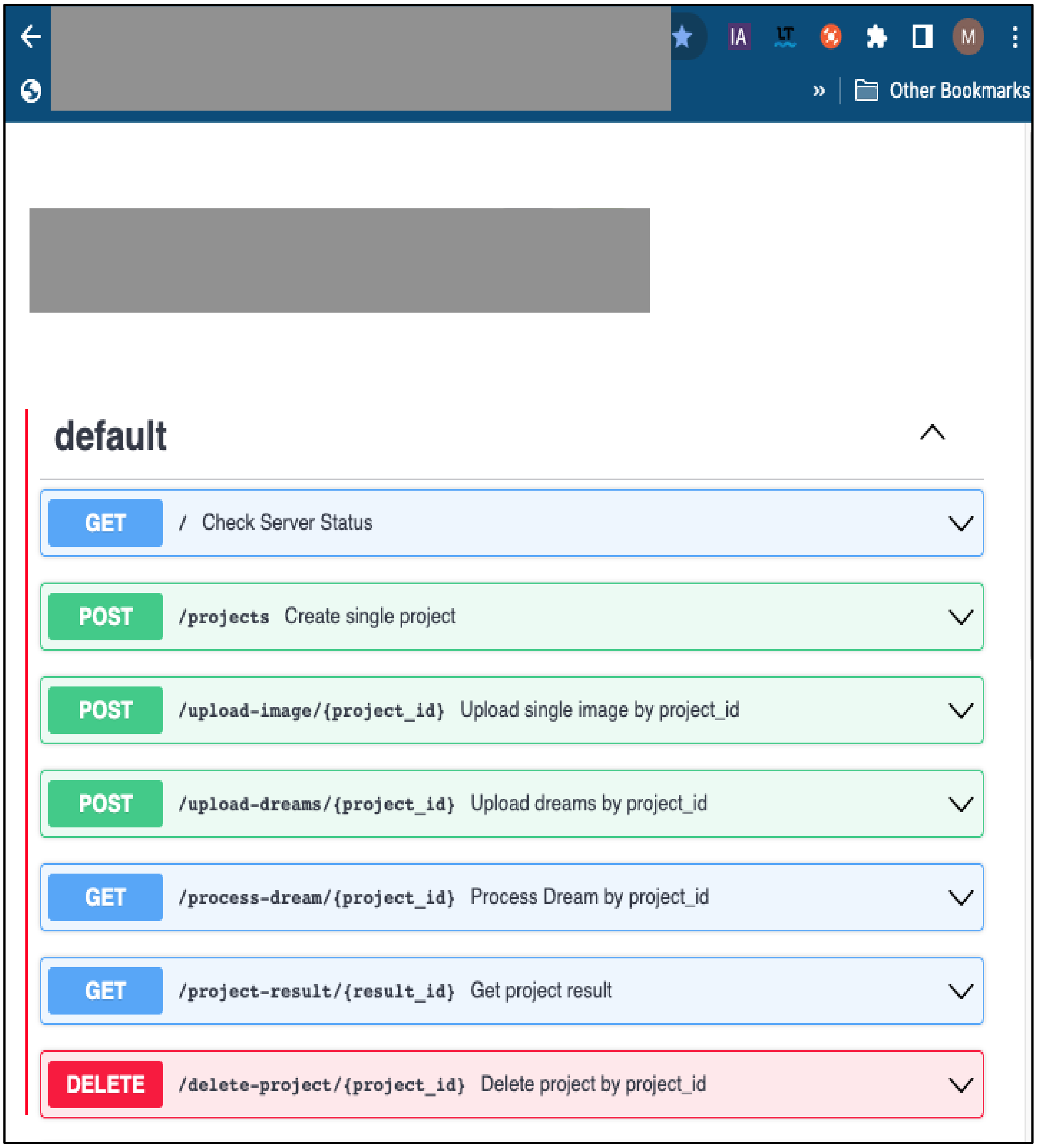


```
    {
        "First_name": "Leonardo",
        "Last_name": "DiCaprio",
        "ageId": 1,
        "id": 8
    },
    {
        "First_name": "Tom",
        "Last_name": "Hanks",
        "ageId": 2,
        "id": 9
    },
    {
        "First_name": "Shah Rukh",
        "Last_name": "Khan",
        "ageId": 3,
        "id": 10
    },
    {
        "First_name": "Will",
        "Last_name": "Smith",
        "ageId": 4,
        "id": 11
    },
    {
        "First_name": "Akshay",
        "Last_name": "Kumar",
        "ageId": 5,
        "id": 12
    },
    {
        "First_name": "Amitabh"
```

“Postman Documentation”

Live Project of Kite Dream Studio:

Project Name [redacted]
Project Link : [redacted]
Video Link : ht[redacted]
Project details by Postman: [redacted]



[redacted]

[redacted]

default

GET / Check Server Status

POST /projects Create single project

POST /upload-image/{project_id} Upload single image by project_id

POST /upload-dreams/{project_id} Upload dreams by project_id

GET /process-dream/{project_id} Process Dream by project_id

GET /project-result/{result_id} Get project result

DELETE /delete-project/{project_id} Delete project by project_id