API Testing

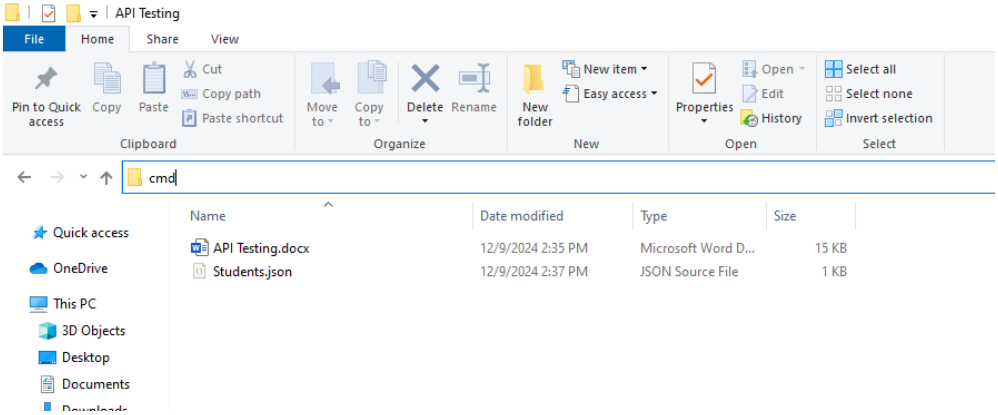
Create an Own/Dummy REST API " Students.Json " file with the following data :


```
{
 "students":[
 {
     "id": 1,
     "name": "John Doe",
     "age": 18,
     "grade": "12th",
     "subjects": [
        "Math",
        "Physics",
        "English"
     ]
   },
   {
     "id": 2,
     "name": "Jane Smith",
     "age": 17,
     "grade": "11th",
     "subjects": [
        "Biology",
        "Chemistry",
        "History"
     ]
   },
   {
     "id": 3,
     "name": "David Johnson",
     "age": 16,
     "grade": "10th",
     "subjects": [
        "Computer Science",
        "Spanish",
        "Art"
     ]
    }
   ]
}
```

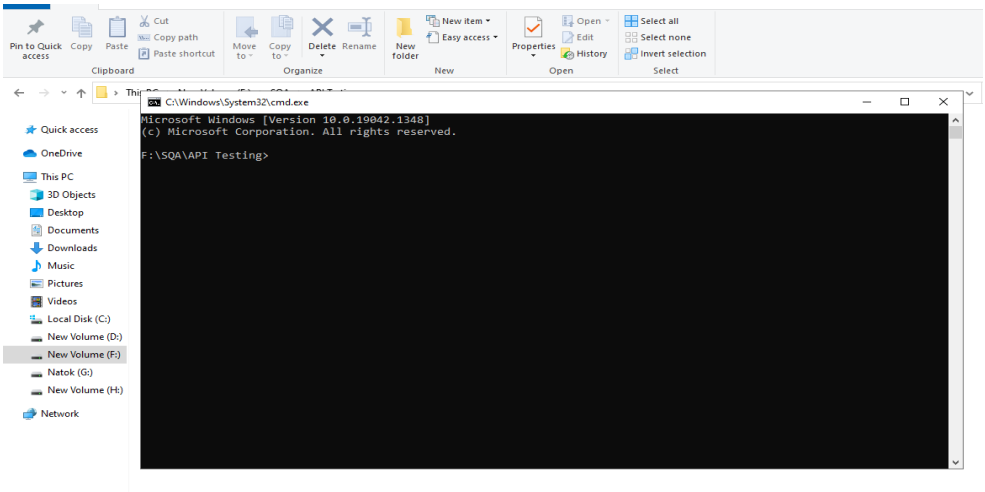
How to create own API ? **Task no-1**

Ans: Goto the location where the " Students.json " File is stored.

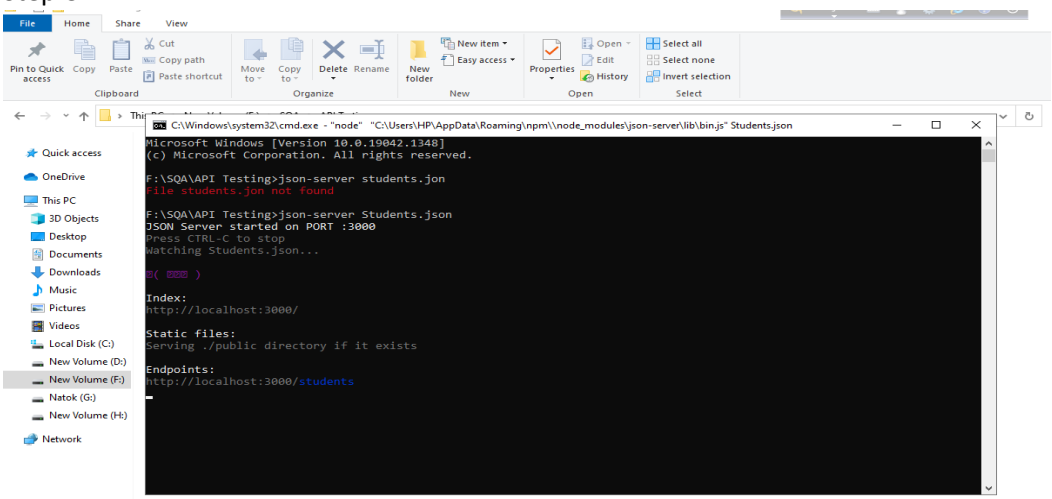
Step-1



Step-2

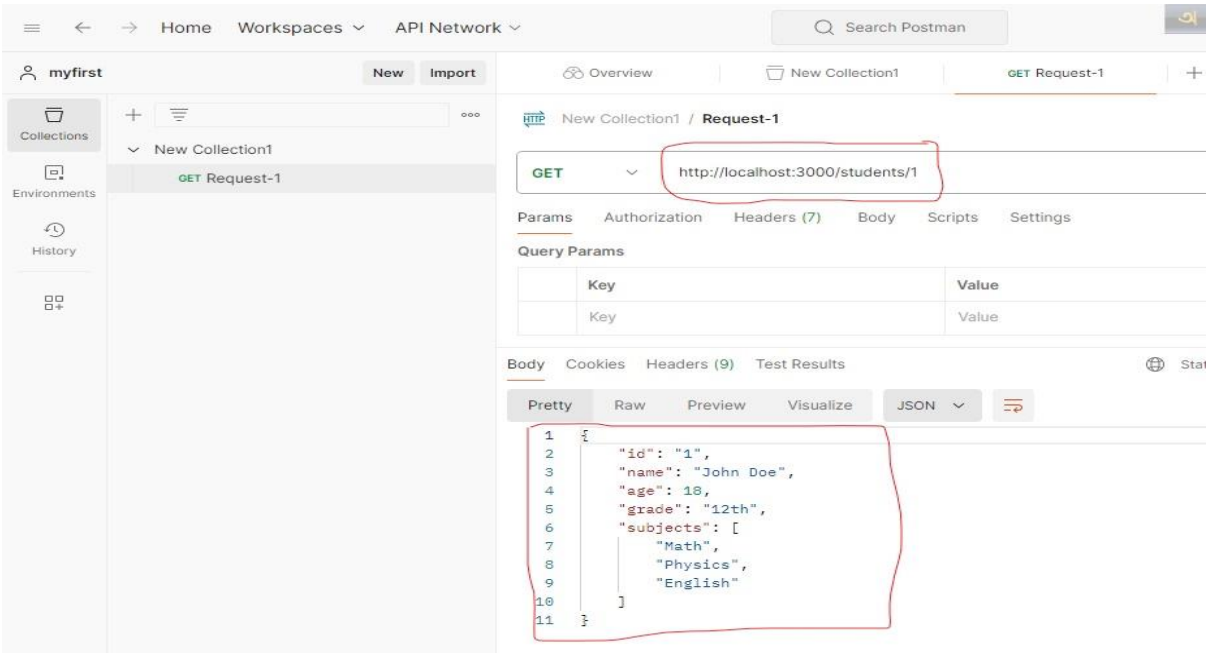


Step-3

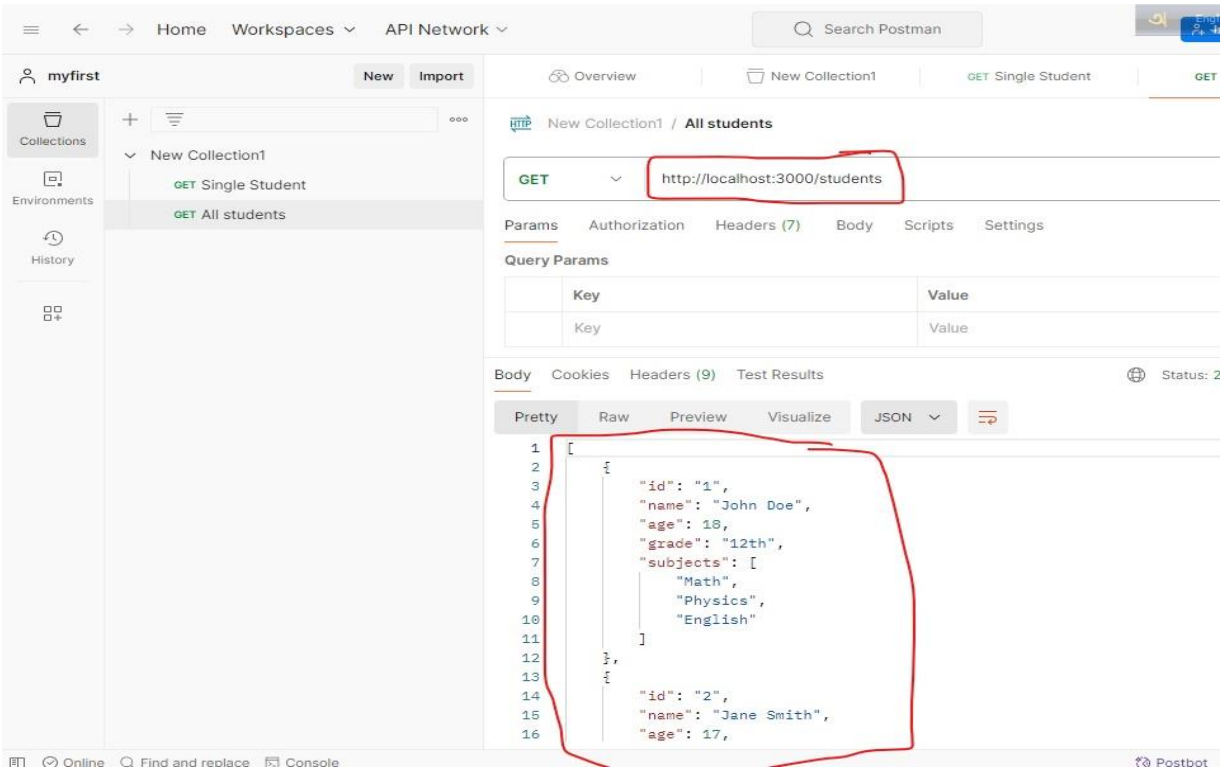


Link: http://localhost:3000/students

How to get single student data from the above URL? **Task No- 2**



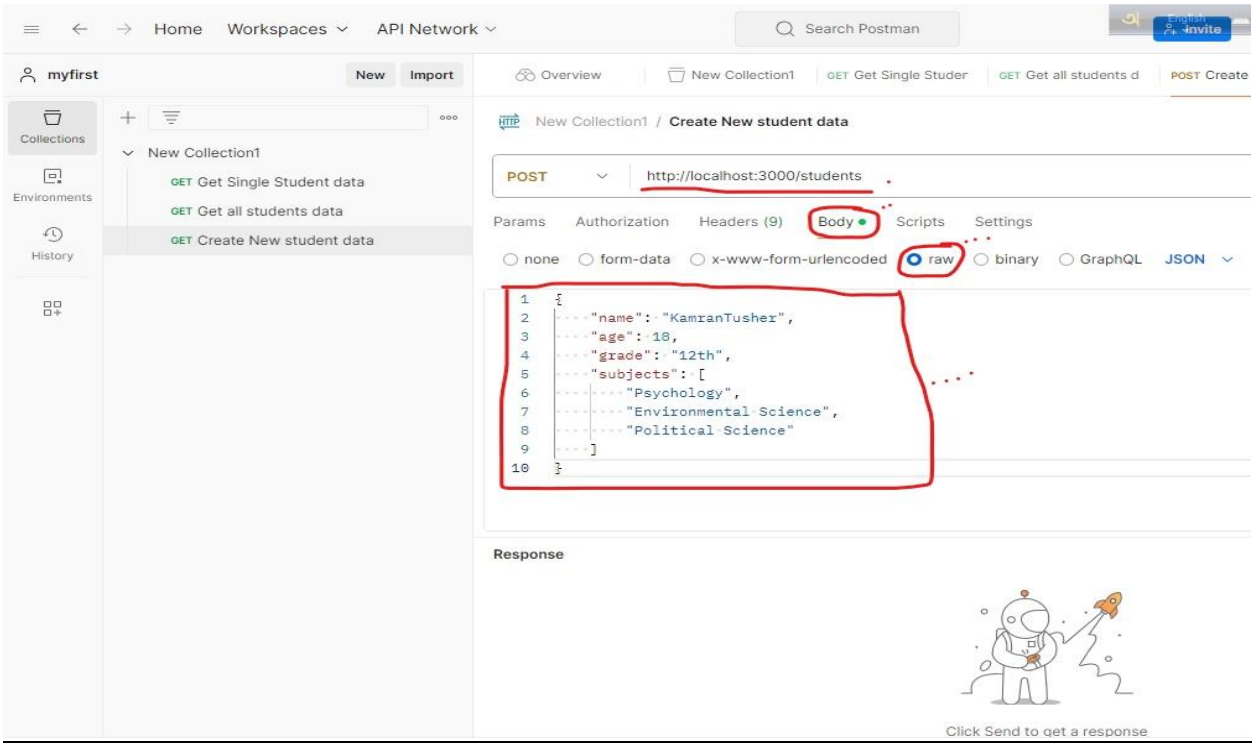
How to get all students data from the above URL? **Task No- 3**



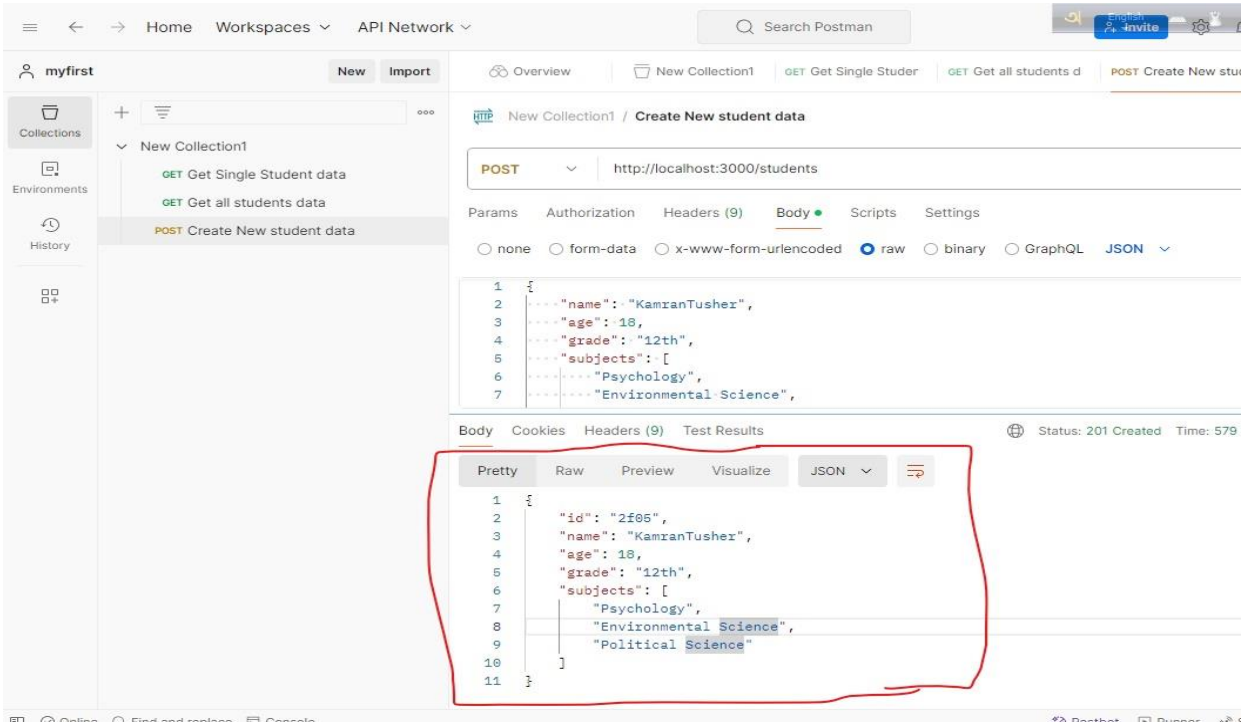
How to create new student data? Create = Post **Task No- 4**

Ans: Add a new collection " Post " then give the url. Then insert a new request in the "Request Payload "

Step-1 (Request Payload)

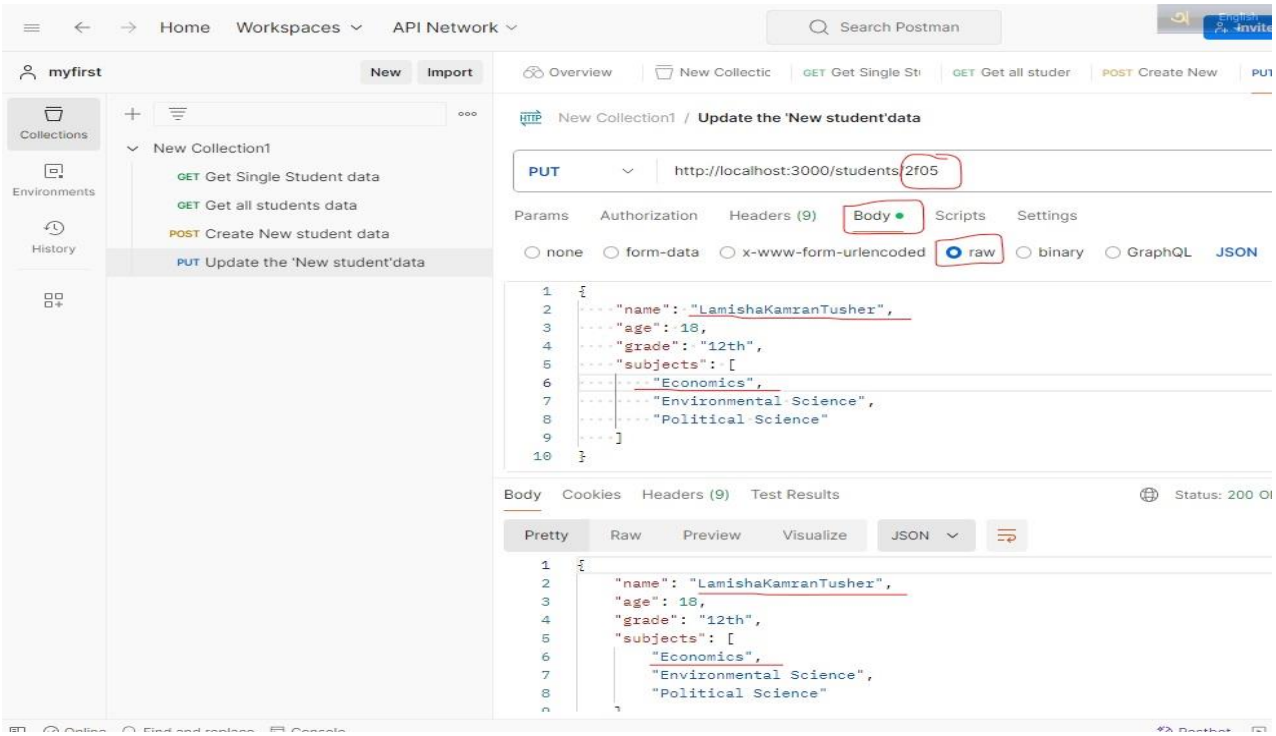


Step-2 (Response Payload)



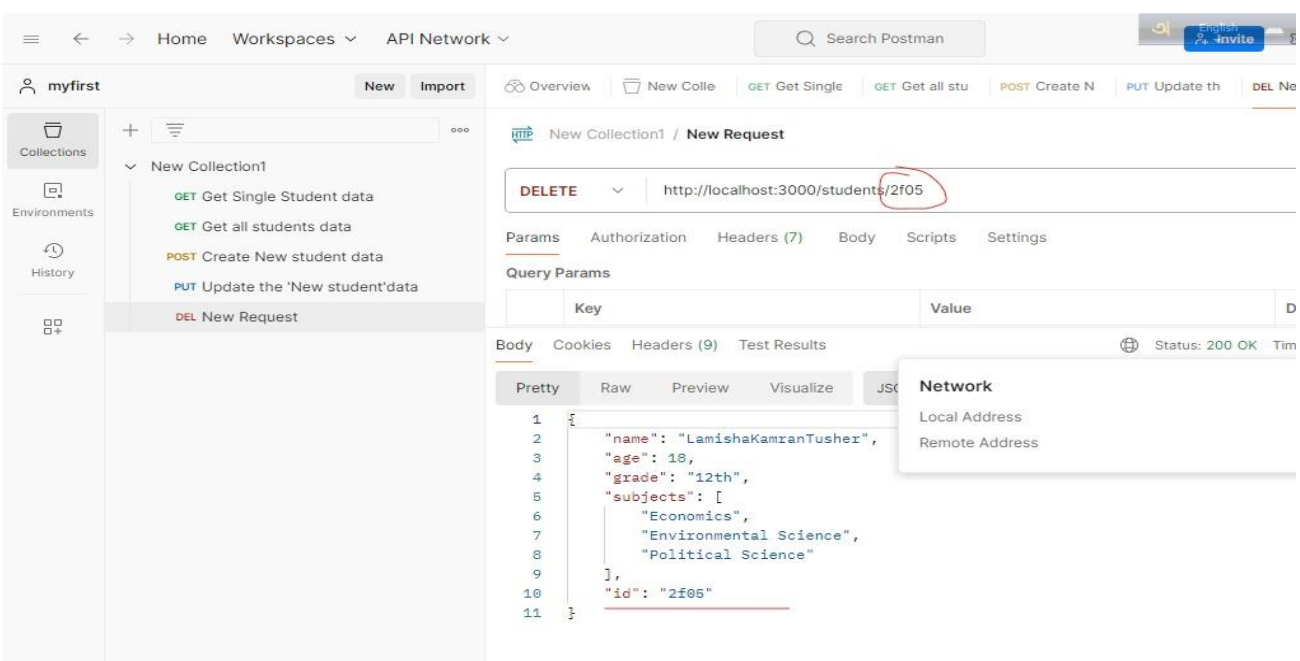
How to update the earlier “New student data” data? Update = Put **Task No- 5**

Ans : Make sure to set the “Id Number” beside the link_(http://localhost:3000/students/2f05)



How to delete any data from the request ? **Task No- 6**

Ans : I want to delete ID = “ 2f05 “



Validation :

1. Response Body
2. Headers
3. Cookies
4. Status Code
5. Time

JSON ---Javascript object Notation

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**

JSON Data Types

- Number
- String
- Boolean
- Null
- Object
- Array

Note: In JSON format, we have to represent the data in the form of KEY: Value Pair

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" }
}
```

String

Example

{

"name": "John",

}

Note: KEY is always included in " " double quotation

"Key": Value { "name": " John " }

Here { "name" : " John " }------name is included in double quotation But John included in double quotation here because John is string.

When we input multiple inputs in one variable then we use [ ] this is called JSON Array.

Example:

{

"name": "John",

"age": 30,

"phone": [12345,6789]

}

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.

```
{ "middlename":null }
```

Example :

```
{
"Firstname": "John",
"Lastname": Null,
"age": 30,
"phone": [12345,6789],
"Status": true
}
```

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": "Nick",
       "lastname": "Thameson"
    }
  ]
}
```

JSON Object object—Which Contains Multiple KEY : Value Pairs
Student Data (Student Contains SID, SName, Grade) HERE 4 Student/Object.

```
{

"Student":[

        {
        "SID":101,
        "SName":"Kamran Tusher",
        "Grade":A
        },

         {
        "SID":102,
        "SName":"Lamisha Rahman",
        "Grade":A+
        },

        {
        "SID":103,
        "SName":"Zenith Chowdhury",
        "Grade":A
        },

        {
        "SID":104,
        "SName":"Liyana Lio",
        "Grade":A
        }
       ]
}
```


Explanation

```
{                                                        Object Starts
    "Title": "The Cuckoo's Calling"
    "Author": "Robert Galbraith",
    "Genre": "classic crime novel",
    "Detail": {                                          Object Starts
        "Publisher": "Little Brown"                      Value string
        "Publication_Year": 2013,                        Value number
        "ISBN-13": 9781408704004,
        "Language": "English",
        "Pages": 494
    }                                                    Object ends
    "Price": [                                           Array starts
        {                                                Object Starts
            "type": "Hardcover",
            "price": 16.65,
        }                                                Object ends
        {                                                Object Starts
            "type": "Kindle Edition",
            "price": 7.03,
        }                                                Object ends
    ]                                                    Array ends
}                                                        Object ends
```

JSON vs XML

JSON	XML
JSON is simple to read and write.	XML is less simple 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.

Validate JSON Path : **Task No- 7**

```
{
"Student":[
        {
        "SID":101,
        "SName":"Kamran Tusher",
        "Grade":A
        },

        {
        "SID":102,
        "SName":"Lamisha Rahman",
        "Grade":A+
        },

        {
        "SID":103,
        "SName":"Zenith Chowdhury",
        "Grade":A
        },

        {
        "SID":104,
        "SName":"Liyana Lio",
        "Grade":A
        }
        ]
}
```

How to extract any data from the above JSON file by using JSON path ?

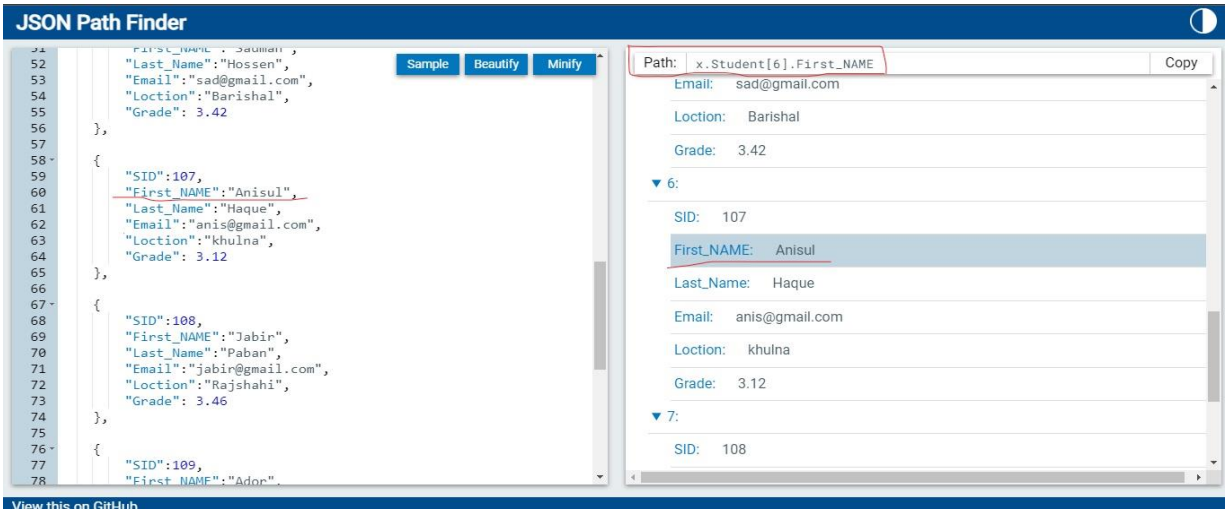
Ans: I want to extract the red-marked data from the file.

Extract Procedure— JSON Path : Student[1].SName-----Lamisha Rahman

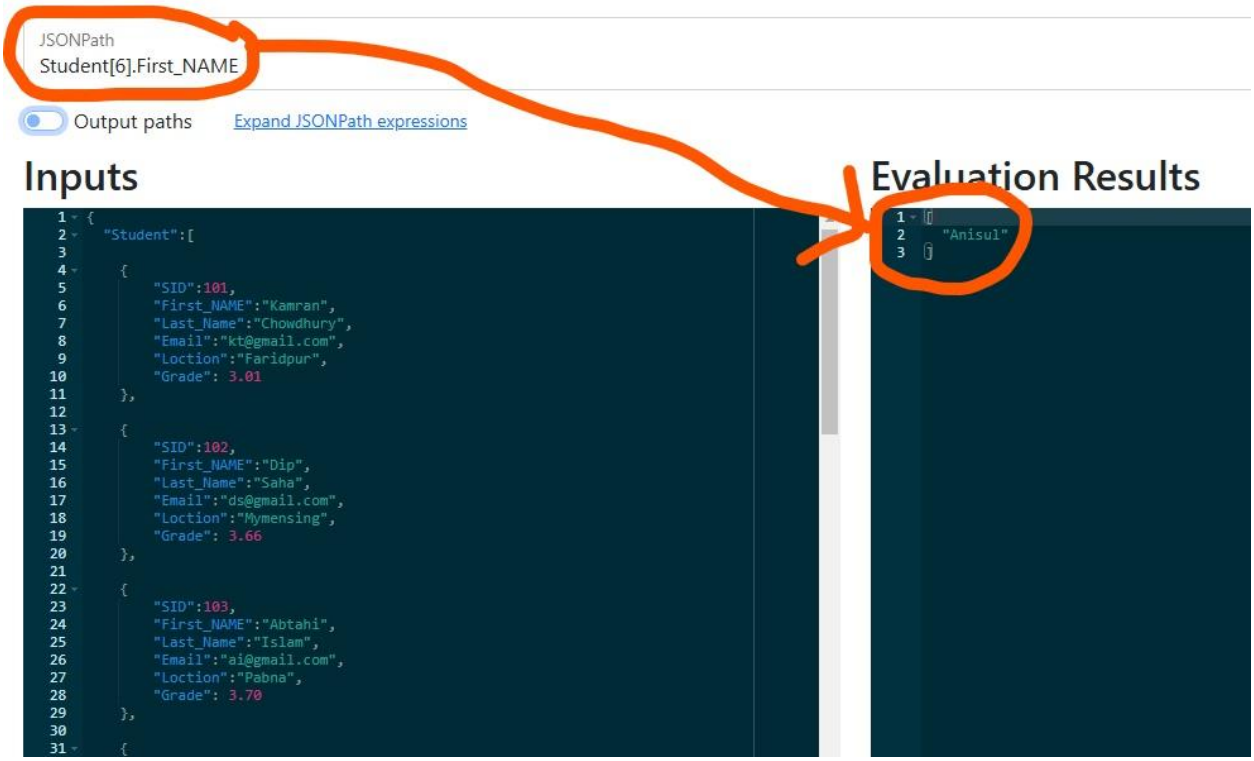
Note: For complex JSON file we need to use tools to extract the data

Tools: 1)JSON Pathfinder. 2) JSON .com

Example: I want get the 2nd object's first name data JSON path Site Link : Link



JSON path Validation (For path validation Site link: Link)

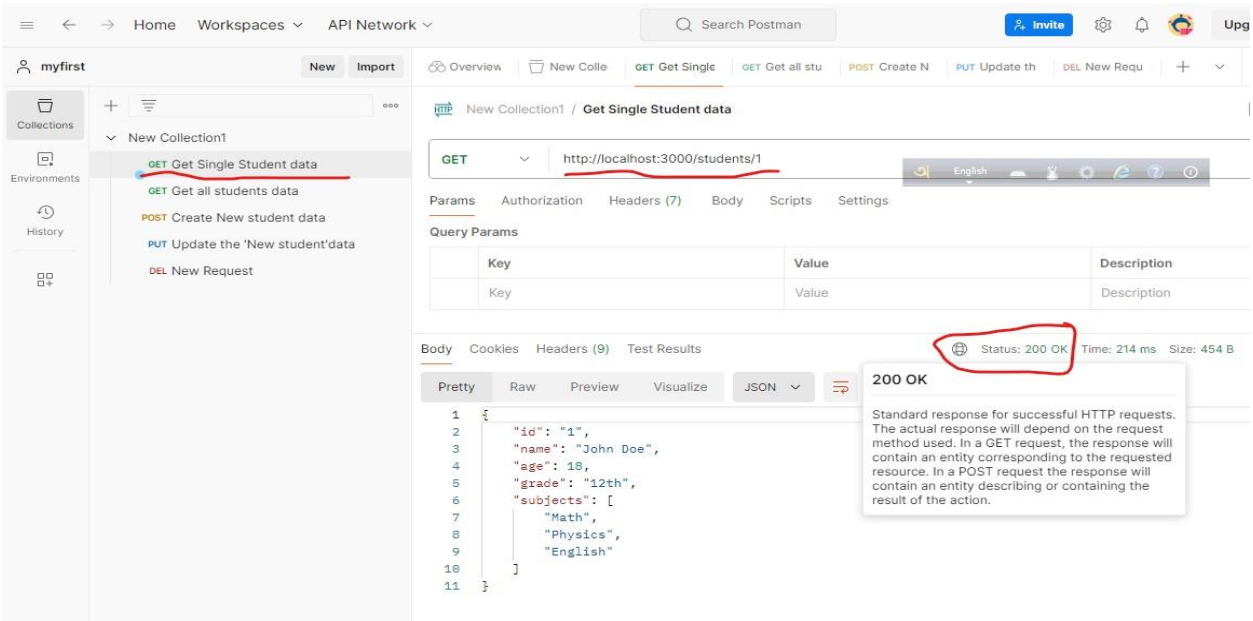


Response Validation

Task No--8

1. Status Code:

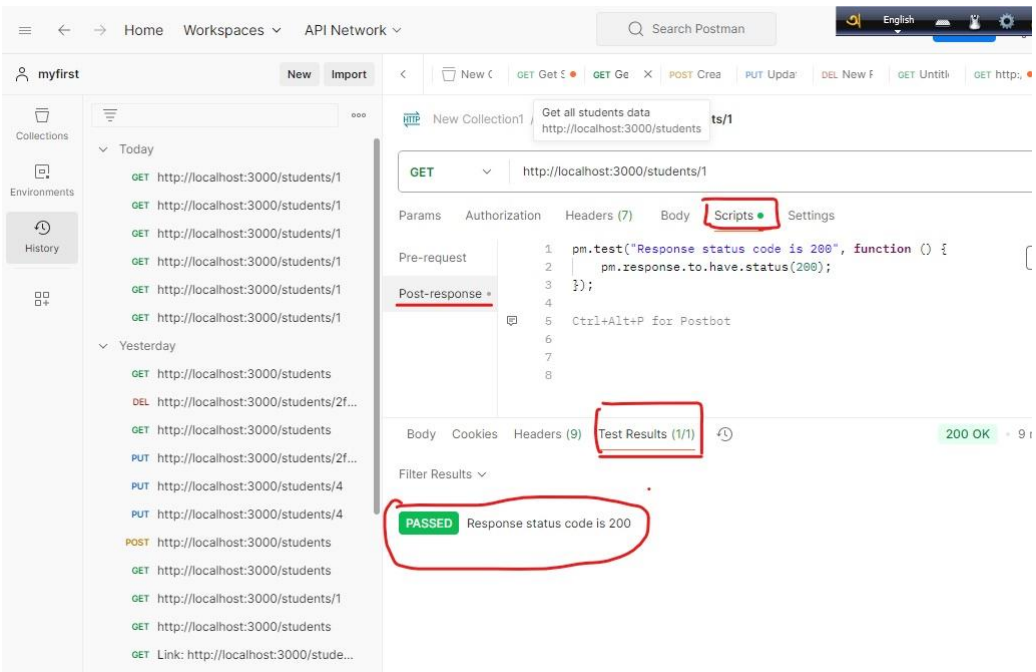
Step-1



Step-2

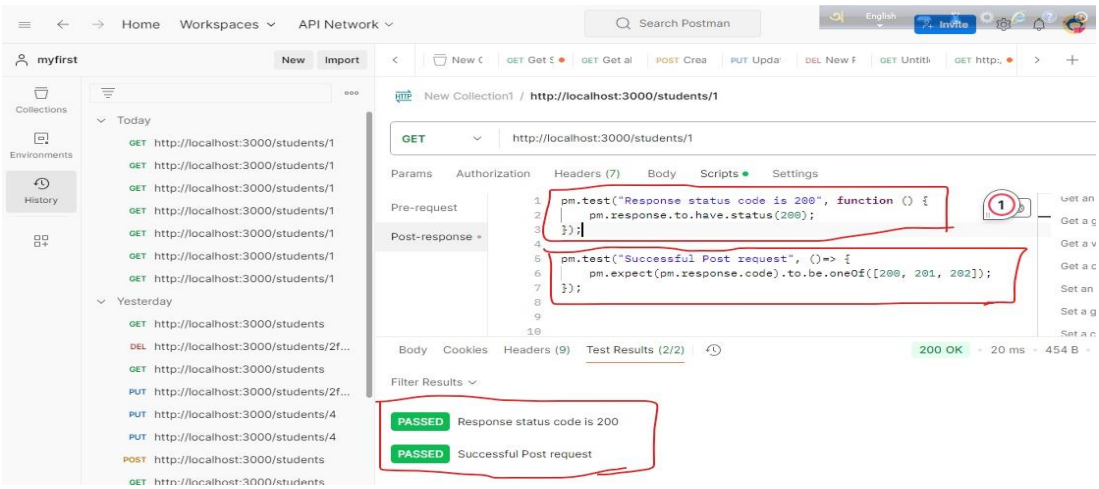
<u>For Single Status code</u>

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



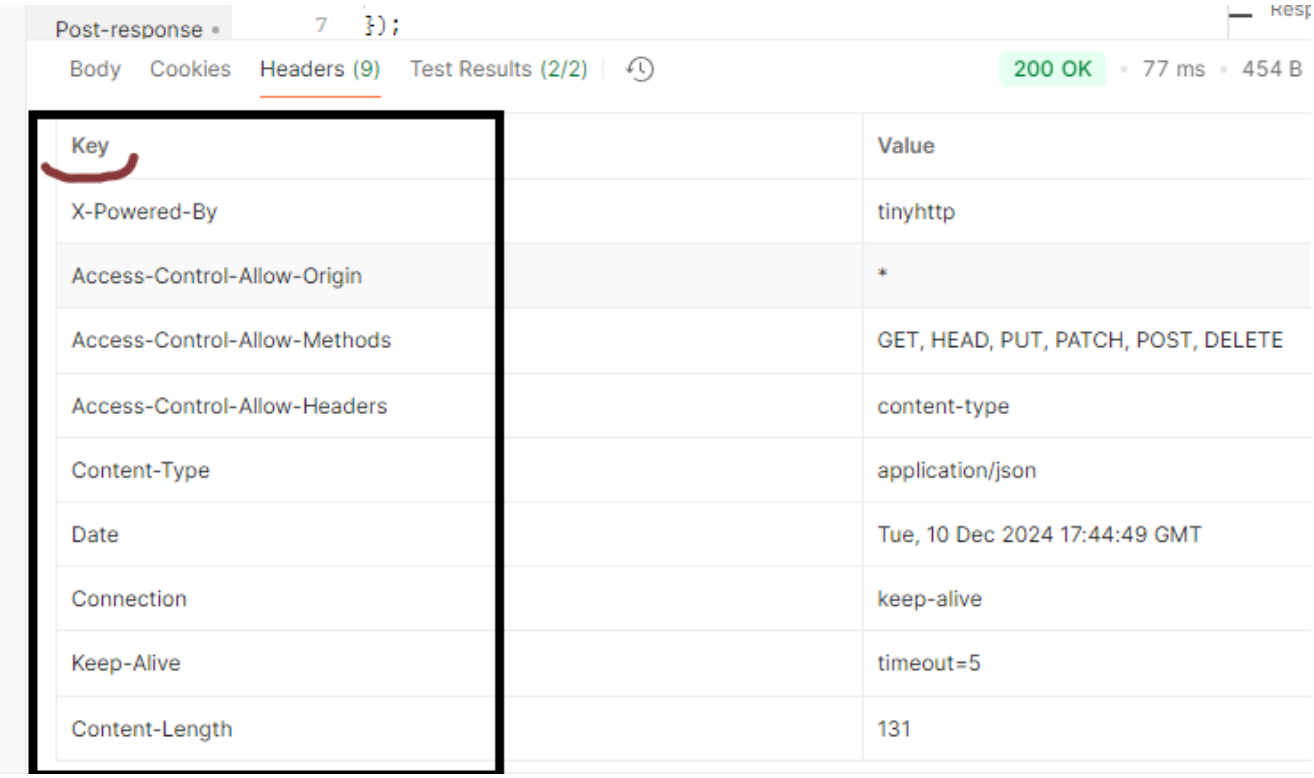
<u>For Multiple Status code</u>

```
pm.test("Successful Post request", ()=> {
   pm.expect(pm.response.code).to.be.oneOf([200, 201, 202]);
});
```



1. **Headers Validation** **Task No—9**

We will Validate “Key” of the header “ given below

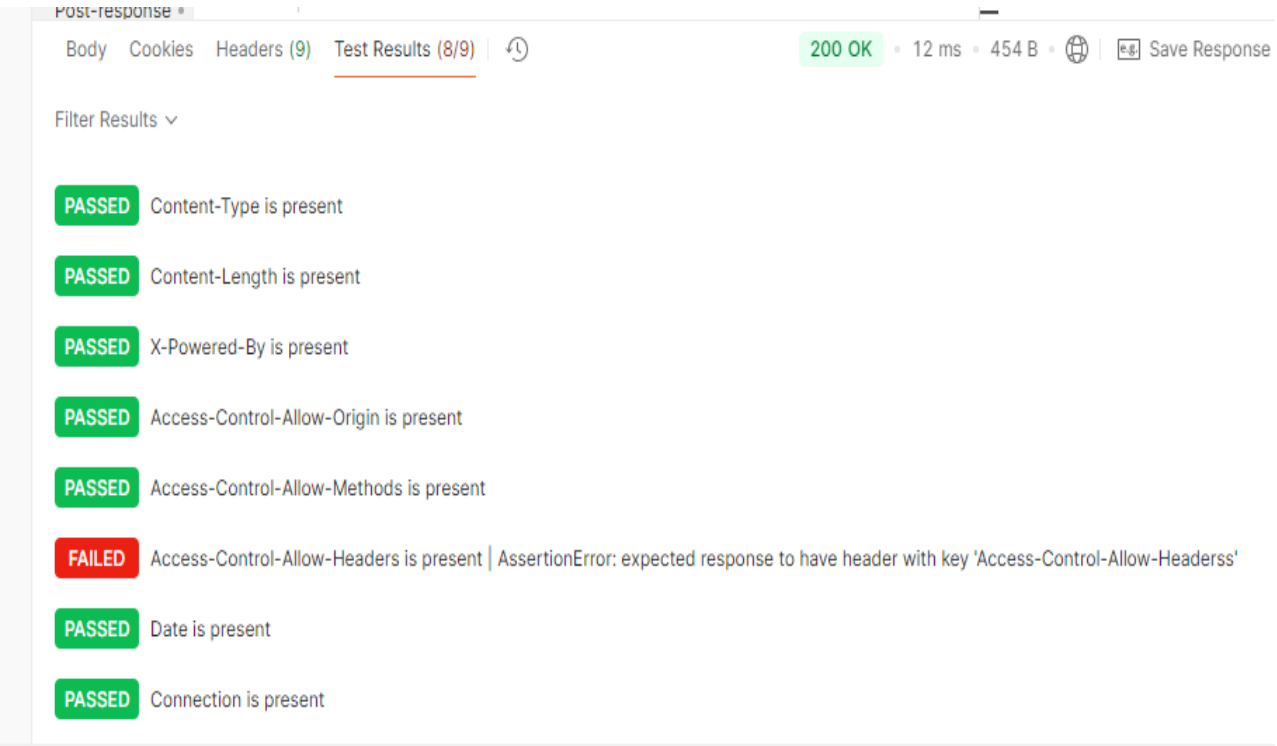


<u>How to validate All the “KEY” response header is present :</u>

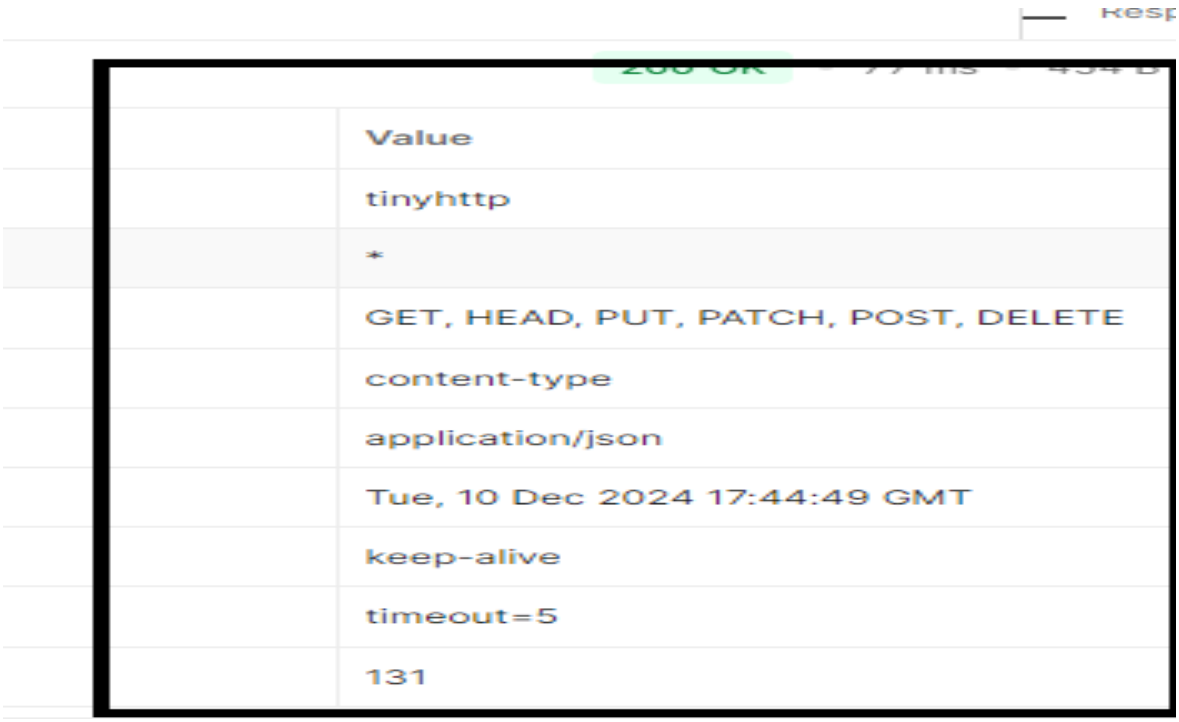
```
pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Content-Type");
});
pm.test("Content-Length is present", function () {
    pm.response.to.have.header("Content-Length");
});
pm.test("X-Powered-By is present", function () {
    pm.response.to.have.header("X-Powered-By");
});
pm.test("Access-Control-Allow-Origin is present", function () {
    pm.response.to.have.header("Access-Control-Allow-Origin");
});
```

```
pm.test("Access-Control-Allow-Methods is present", function () {
   pm.response.to.have.header("Access-Control-Allow-Methods");
});
pm.test("Access-Control-Allow-Headers is present", function () {
   pm.response.to.have.header("Access-Control-Allow-Headerss");
});
pm.test("Date is present", function () {
   pm.response.to.have.header("Date");
});
pm.test("Connection is present", function () {
   pm.response.to.have.header("Connection");
});
pm.test("Keep-Alive is present", function () {
   pm.response.to.have.header("Keep-Alive");
});
```

Here Is the result Images:



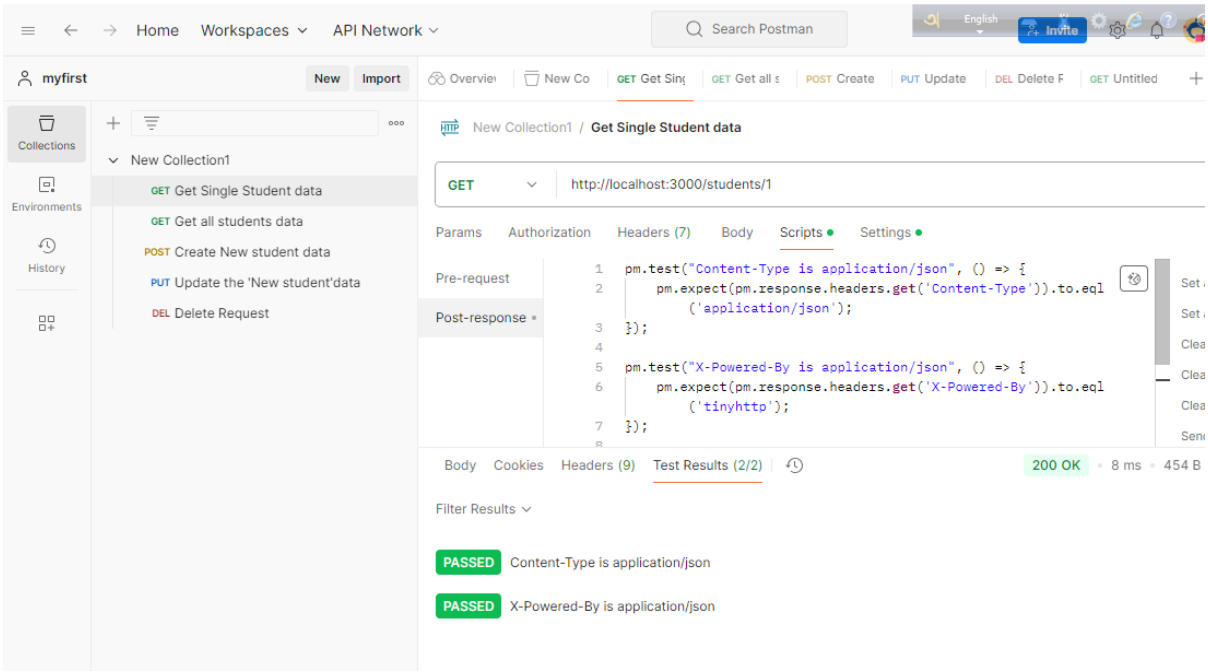
How to Validate the " value " of the header ? Task No-10



Value
tinyhttp
*
GET, HEAD, PUT, PATCH, POST, DELETE
content-type
application/json
Tue, 10 Dec 2024 17:44:49 GMT
keep-alive
timeout=5
131

Code is given below

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



All the “ Value “ validated together using an Array

```
const headers = [
  { key: "X-Powered-By", value: "tinyhttp" },
  { key: "Access-Control-Allow-Origin", value: "*" },
  { key: "Access-Control-Allow-Methods", value: "GET, HEAD, PUT, PATCH, POST, DELETE" },
  { key: "Access-Control-Allow-Headers", value: "content-type" },
  { key: "Content-Type", value: "application/json" },
  //{ key: "Date", value: "Tue, 10 Dec 2024 20:04:59 GMT" },
  { key: "Connection", value: "keep-alive" },
  { key: "Keep-Alive", value: "timeout=5" },
  { key: "Content-Length", value: "131" }
];
// Loop through headers and validate each using a basic for loop
for (let i = 0; i < headers.length; i++) {
  const header = headers[i];
  pm.test(`${header.key} is ${header.value}`, () => {
    pm.expect(pm.response.headers.get(header.key)).to.eql(header.value);
  });
}
```

2. **Cookies Validation :**

We need to verify the cookie name and the value

How to check the cookies are present in the response?

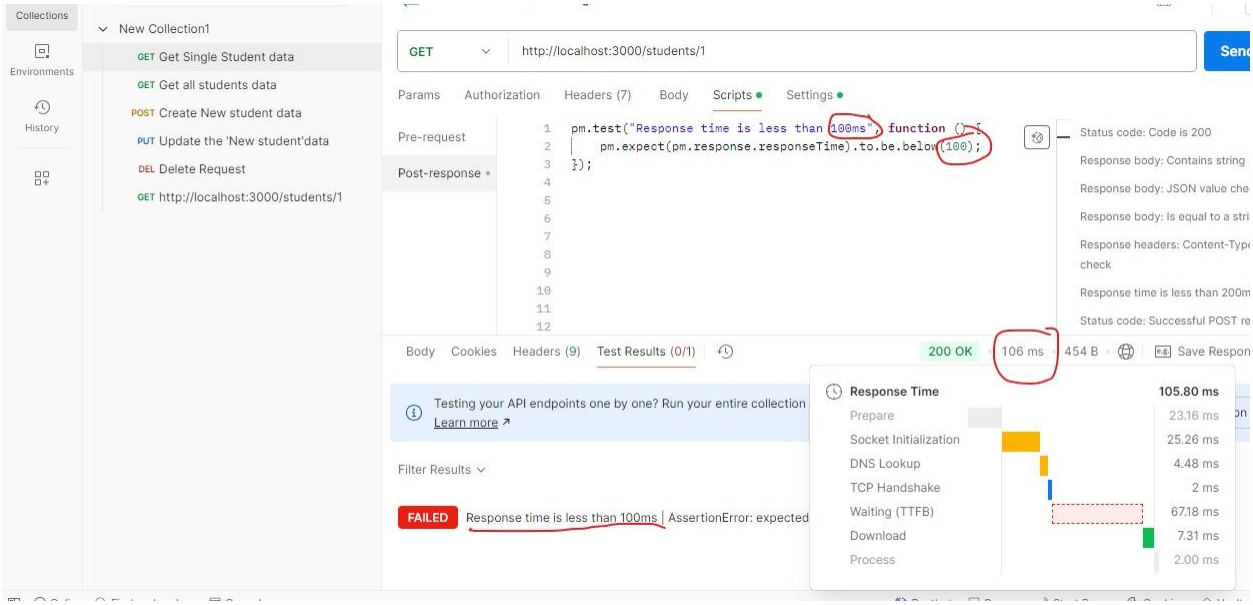
Ans:

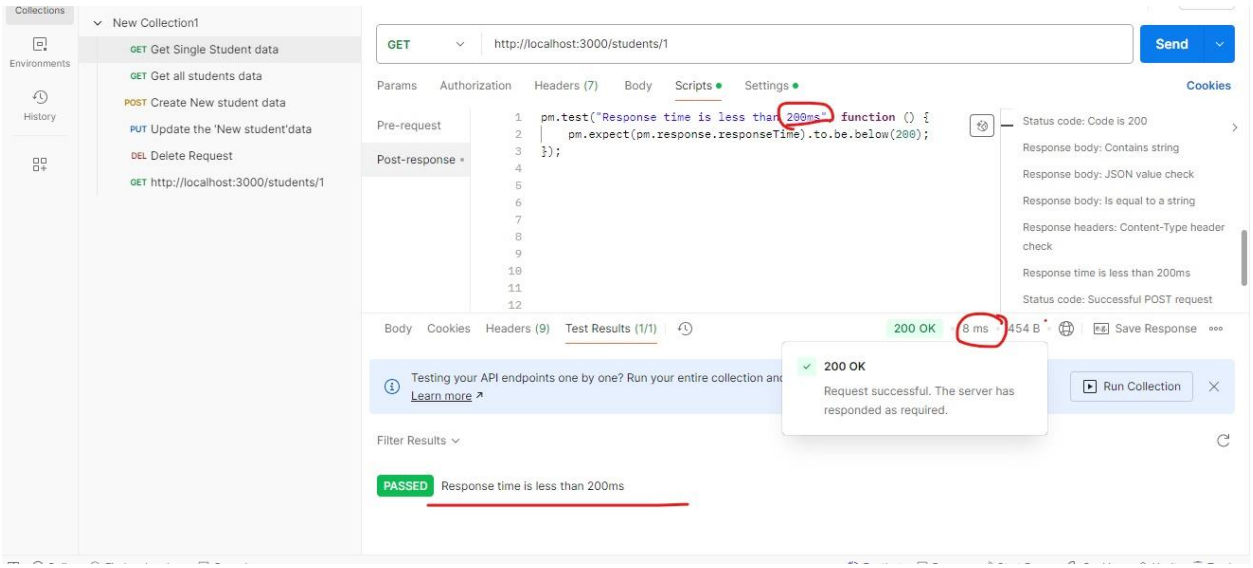
3. **Response Time validation :**

How to check the response time ?

Ans:
```
pm.test("Response time is less than 200ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(200);
});
```

Actually, the response time keep changing during the execution.





Response Body: Task No-

Validate the "Type" of the value: Validate all the "data type" of these data from every assertion in this Response body.

How to validate the 'Data type ' of the value from these assertions in the response body?

Here is the response body: (For single object in the response body)

```
{
   "id": "1",                    //verify the data type of id
   "name": "John Doe",   //verify the data type of name
   "age": 18,                //verify the data type of id
   "grade": "12th",       //verify the data type of id
   "subjects": [            //verify the data type of subject
      "Math",
      "Physics",
      "English"
   ]
}
```

CODE:

```
const jsonData = pm.response.json();
pm.test("Test the data type of the response",() =>{
pm.expect(jsonData).to.be.an("object");
pm.expect(jsonData.name).to.be.an("String");
pm.expect(jsonData.id).to.be.an("String");
pm.expect(jsonData.age).to.be.an("number");
pm.expect(jsonData.subjects).to.be.an("Array");

});
```

myfirst

New Import

Collections

Environments

History

New Collection1

- GET Get Single Student data
- GET Get all students data
- POST Create New student data
- PUT Update the 'New student'data
- DEL Delete Request
- GET http://localhost:3000/students/1

Overview | GET http://loca | GET Get Single | GET Get all stu | Postman \ | Runner | New

New Collection1 / **Get Single Student data**

GET http://localhost:3000/students/1

Params | Authorization | Headers (7) | Body | Scripts • | Settings •

Pre-request

Post-response •

```
const jsonData = pm.response.json();
pm.test("Test the data type of the response",() =>{
pm.expect(jsonData).to.be.an("object");
pm.expect(jsonData.name).to.be.an("String");
pm.expect(jsonData.id).to.be.an("String");
pm.expect(jsonData.age).to.be.an("number");
pm.expect(jsonData.subjects).to.be.an("Array");

});
```

Body | Cookies | Headers (9) | Test Results (1/1) | 200 OK

Pretty | Raw | Preview | Visualize | JSON

```
{
    "id": "1",
    "name": "John Doe",
    "age": 18,
    "grade": "12th",
    "subjects": [
        "Math",
        "Physics",
        "English"
    ]
}
```

Home | Workspaces | API Network | Search Postman | English | Invite

myfirst

New Import

Collections

Environments

History

New Collection1

- GET Get Single Student data
- GET Get all students data
- POST Create New student data
- PUT Update the 'New student'data
- DEL Delete Request
- GET http://localhost:3000/students/1

Overview | GET http://loca | GET Get Single | GET Get all stu | Postman \ | Runner

New Collection1 / **Get Single Student data**

GET http://localhost:3000/students/1

Params | Authorization | Headers (7) | Body | Scripts • | Settings •

Pre-request

Post-response •

```
const jsonData = pm.response.json();
pm.test("Test the data type of the response",() =>{
pm.expect(jsonData).to.be.an("object");
pm.expect(jsonData.name).to.be.an("String");
pm.expect(jsonData.id).to.be.an("String");
pm.expect(jsonData.age).to.be.an("number");
pm.expect(jsonData.subjects).to.be.an("Array");

});
```

Body | Cookies | Headers (9) | Test Results (1/1) | 200 OK

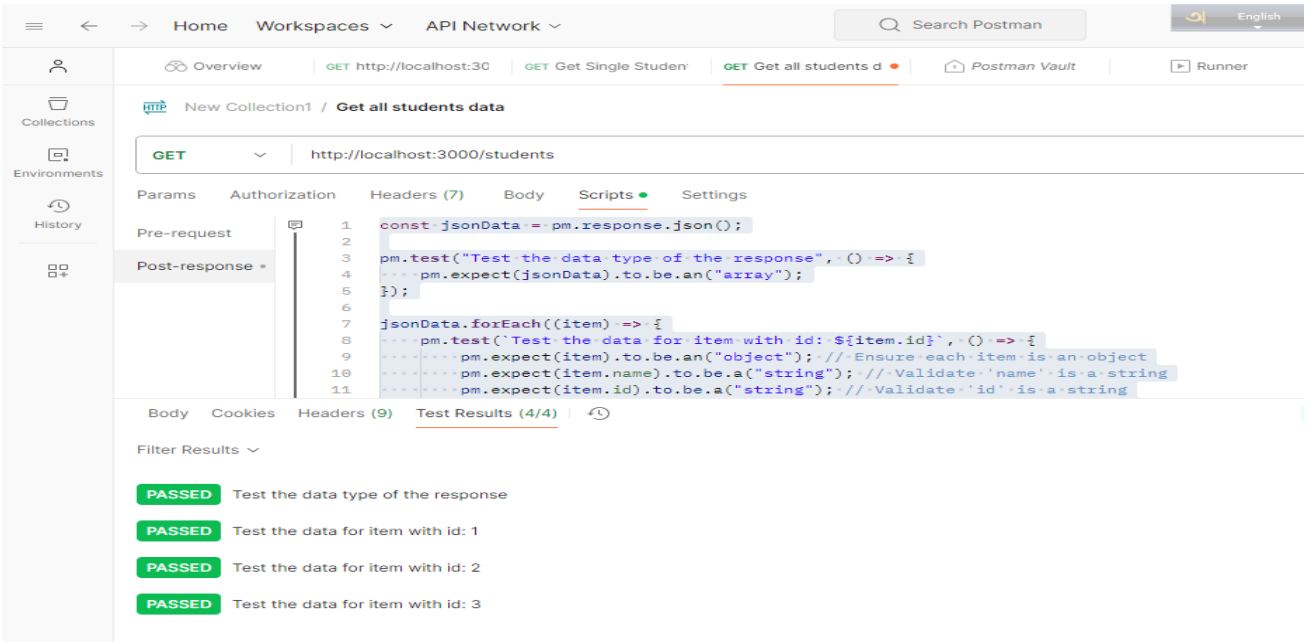
Filter Results

PASSED Test the data type of the response

(For multiple object in the response body):;

CODE:

```
const jsonData = pm.response.json();
pm.test("Test the data type of the response", () => {
   pm.expect(jsonData).to.be.an("array");
});
jsonData.forEach((item) => {
   pm.test(`Test the data for item with id: ${item.id}`, () => {
      pm.expect(item).to.be.an("object"); // Ensure each item is an object
      pm.expect(item.name).to.be.a("string"); // Validate 'name' is a string
      pm.expect(item.id).to.be.a("string"); // Validate 'id' is a string
      pm.expect(item.age).to.be.a("number"); // Validate 'age' is a number
      pm.expect(item.subjects).to.be.an("array"); // Validate 'subjects' is an array
   });
});
```



Array properties in the response body :

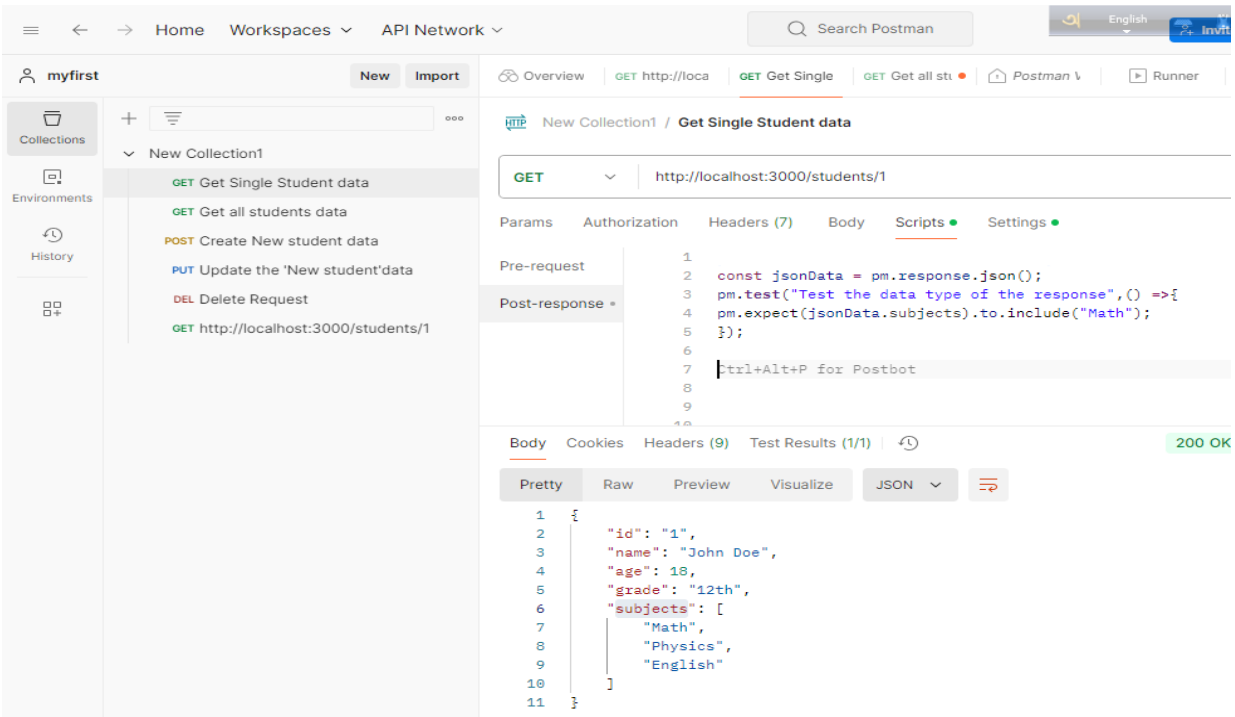
How to validate the Array properties in the response body ?

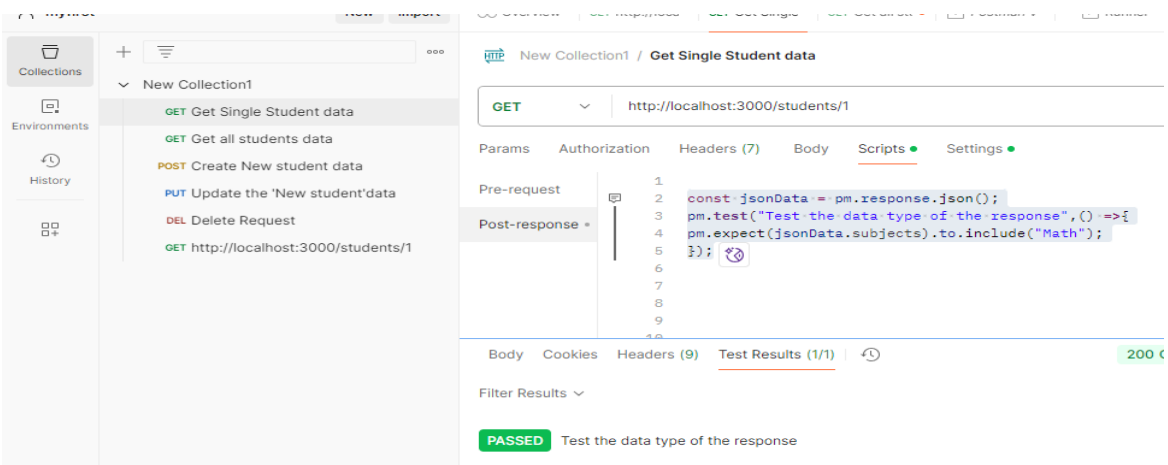
-(Validate a single property in the Array from the response body)

CODE:

```
const jsonData = pm.response.json();

pm.test("Test the data type of the response",() =>{

pm.expect(jsonData.subjects).to.include("Math");

});
```

Input:



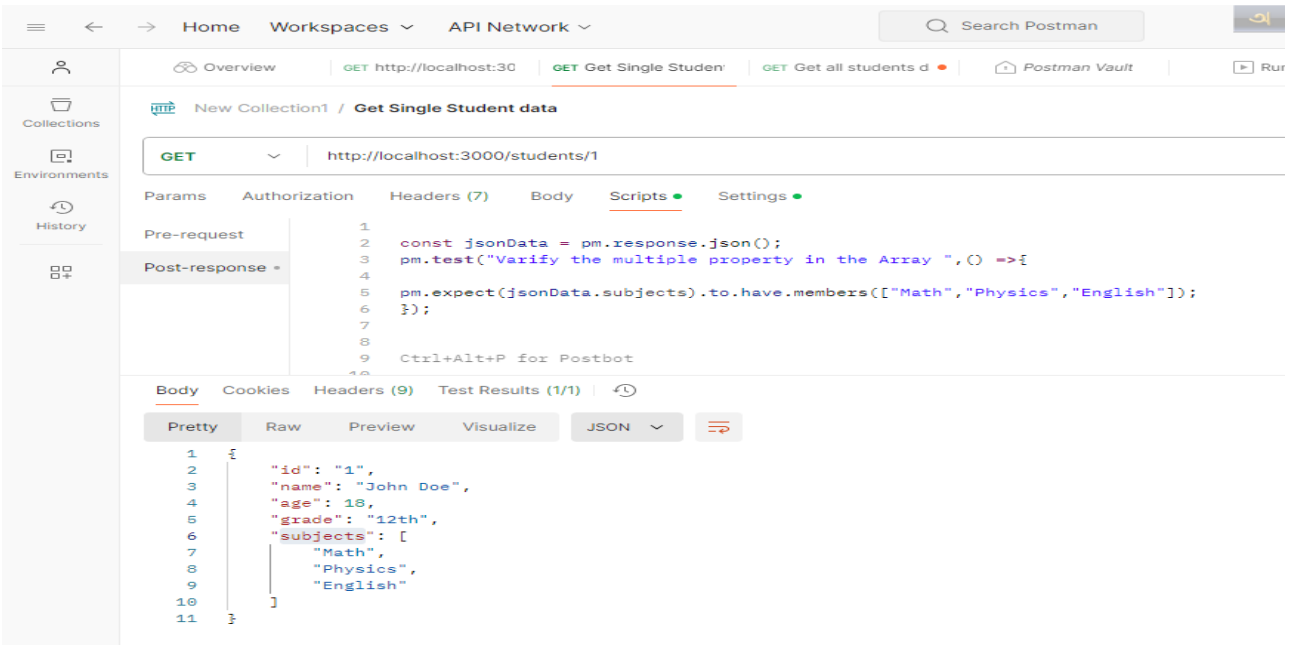


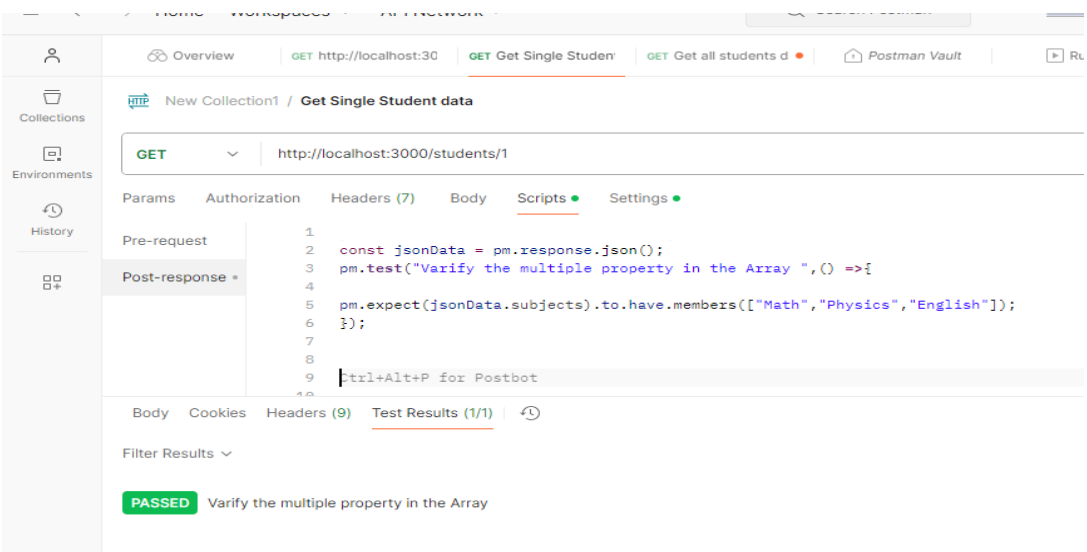
-(Validate a single property in the Array from the response body)

CODE:

```
const jsonData = pm.response.json();
pm.test("Varify the multiple property in the Array ",() =>{
pm.expect(jsonData.subjects).to.have.members(["Math","Physics","English"]);
});
```

Input:





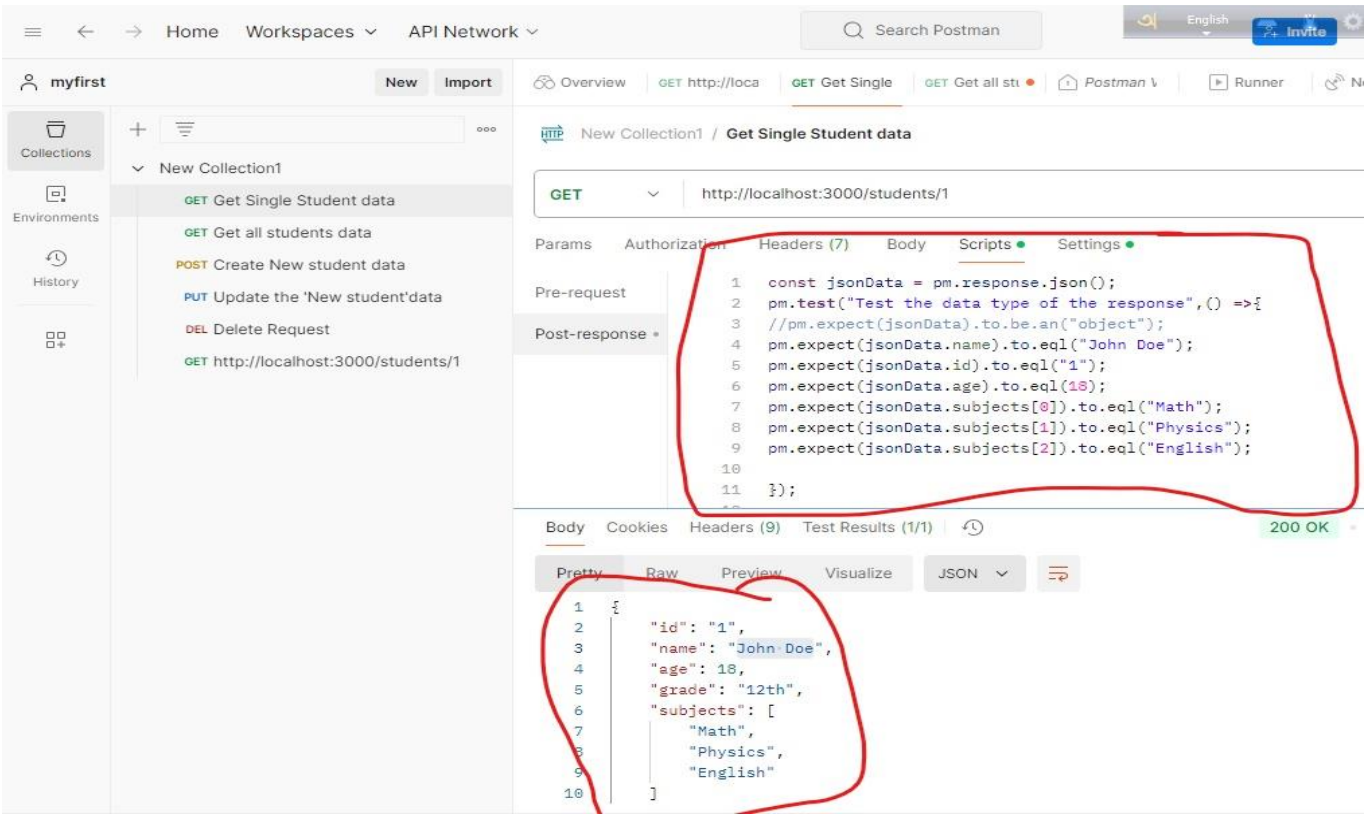
Validate the “value “ Of the response body

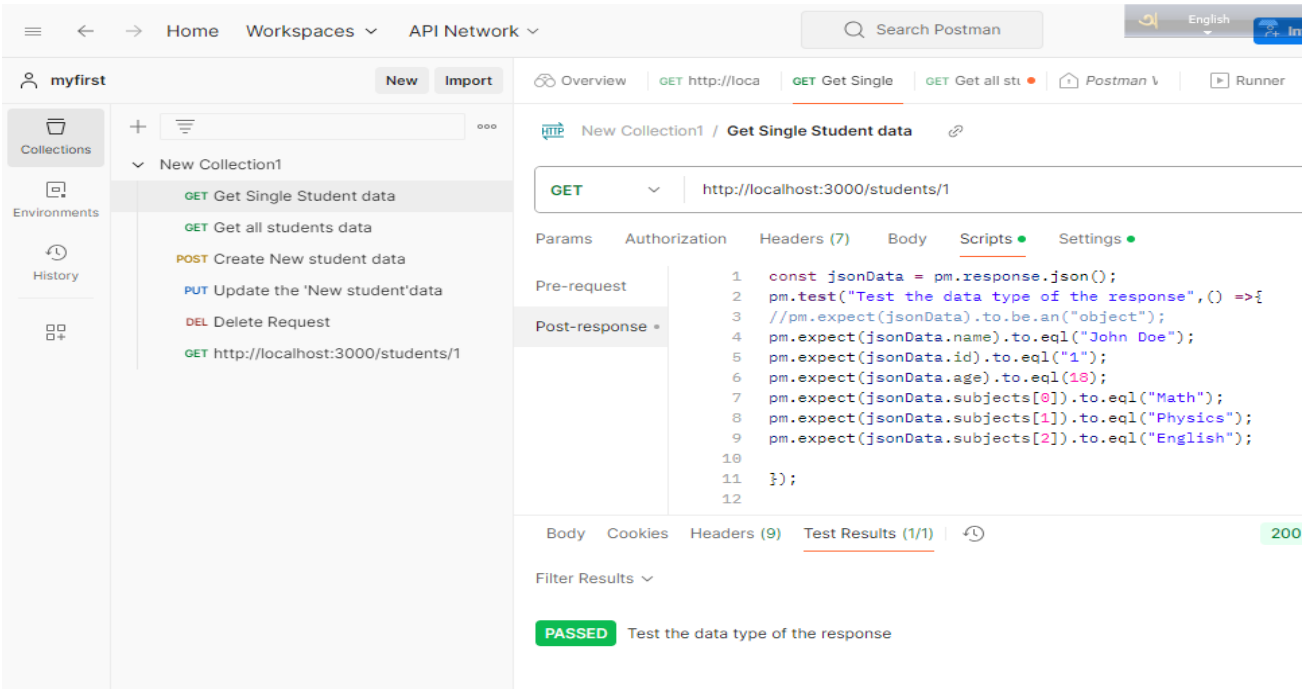
How to validate the value of of every field/assertion is match or not from the response body?

Ans:

CODE:

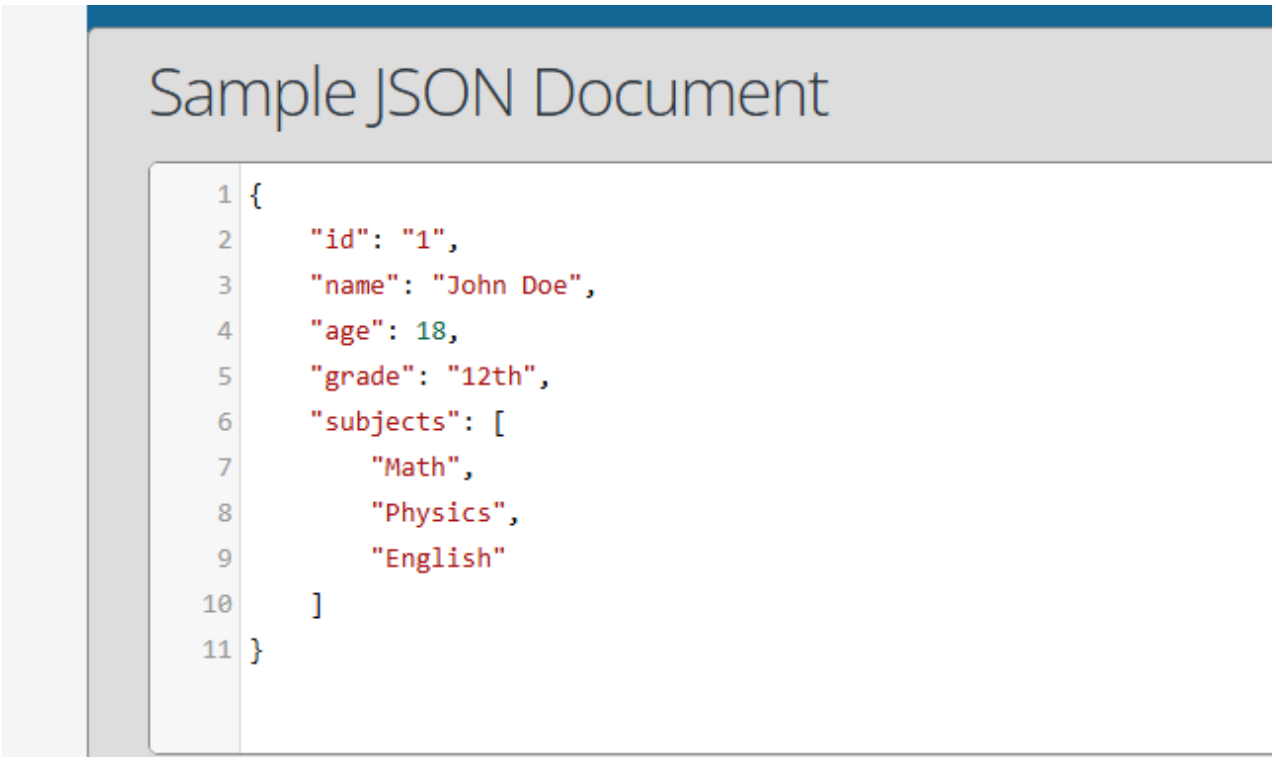
```
const jsonData = pm.response.json();
pm.test("Test the data type of the response",() =>{
//pm.expect(jsonData).to.be.an("object");
pm.expect(jsonData.name).to.eql("John Doe");
pm.expect(jsonData.id).to.eql("1");
pm.expect(jsonData.age).to.eql(18);
pm.expect(jsonData.subjects[0]).to.eql("Math");
pm.expect(jsonData.subjects[1]).to.eql("Physics");
pm.expect(jsonData.subjects[2]).to.eql("English");
});
```





Validate the Json Schema :

Convert Json to JSON schema link : [Link]



```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "id": {
      "type": "string"
    },
    "name": {
      "type": "string"
    },
    "age": {
      "type": "integer"
    },
    "grade": {
      "type": "string"
    },
    "subjects": {
      "type": "array",
      "items": [
        {
          "type": "string"
        },
        {
          "type": "string"
        },
        {
          "type": "string"
        }
      ]
    }
  },
  "required": [
    "id",
    "name",
    "age",
    "grade",
    "subjects"
  ]
}
```

PostMan variables

What is a variable?

Ans: Variable is something which contains some data.

Why is variable need in postman?

Ans: Variable Is used to avoid the duplicate value

Where is use variable in Postman?

Ans : Variable is used in multiple level like Collection, and Environment. Request level.

Scope?

Ans : where we can create the variables

Scope to set up the variables:

1. Global variable ---Set the variable in global level

2. Collection variable---- Set the variable in collection level

3. Request variable---- Set the variable in request level

4. Environment variable---- Set the variable in environment level

5. Data variable---------- Set the variable in data level.

1. Global variable:

How do we create a global variable?

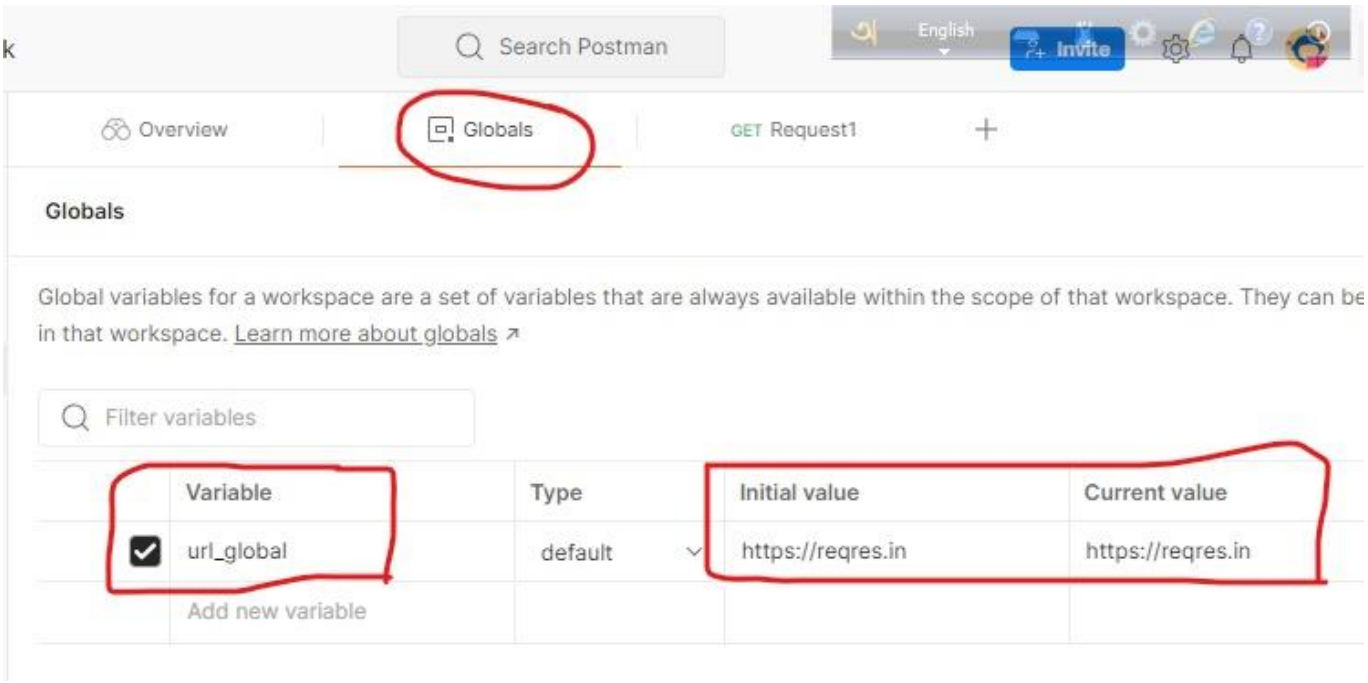
Ans: workspace--→Collection---→Request/Folder.

A global variable is accessible throughout the every workspace.

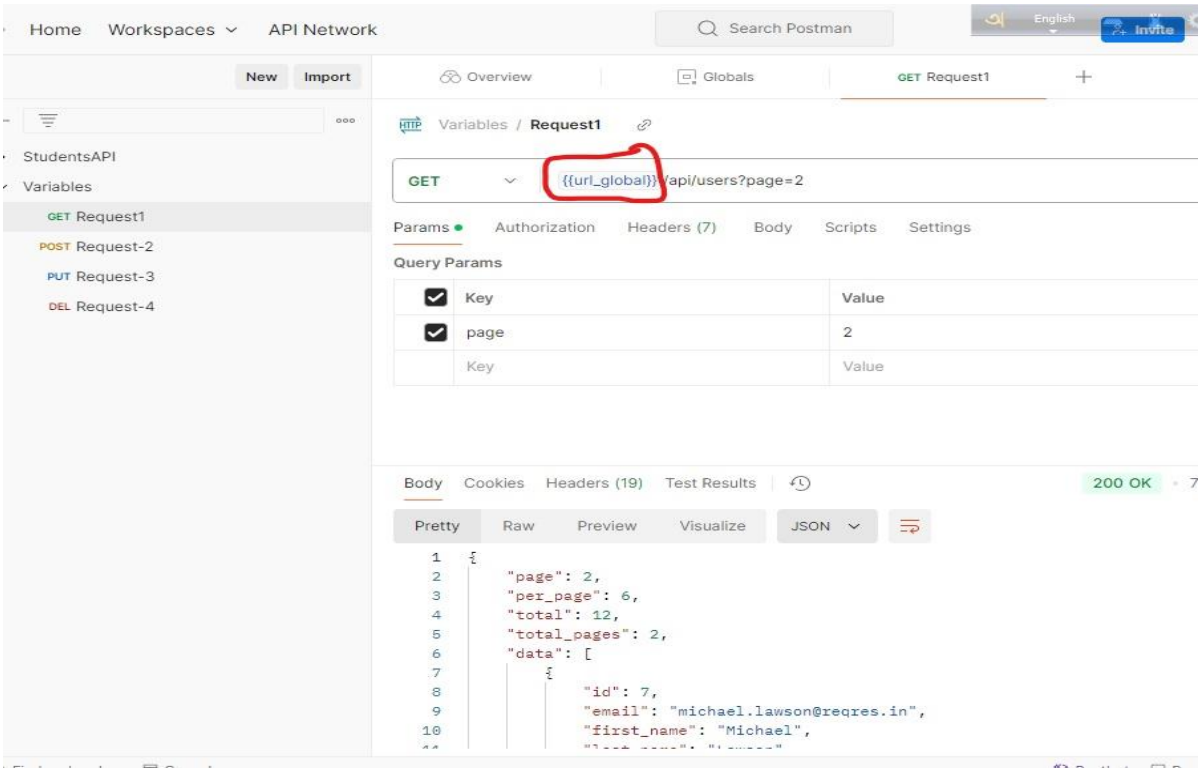
Step-1

Create a Global variable on the collection and save.

Step-2

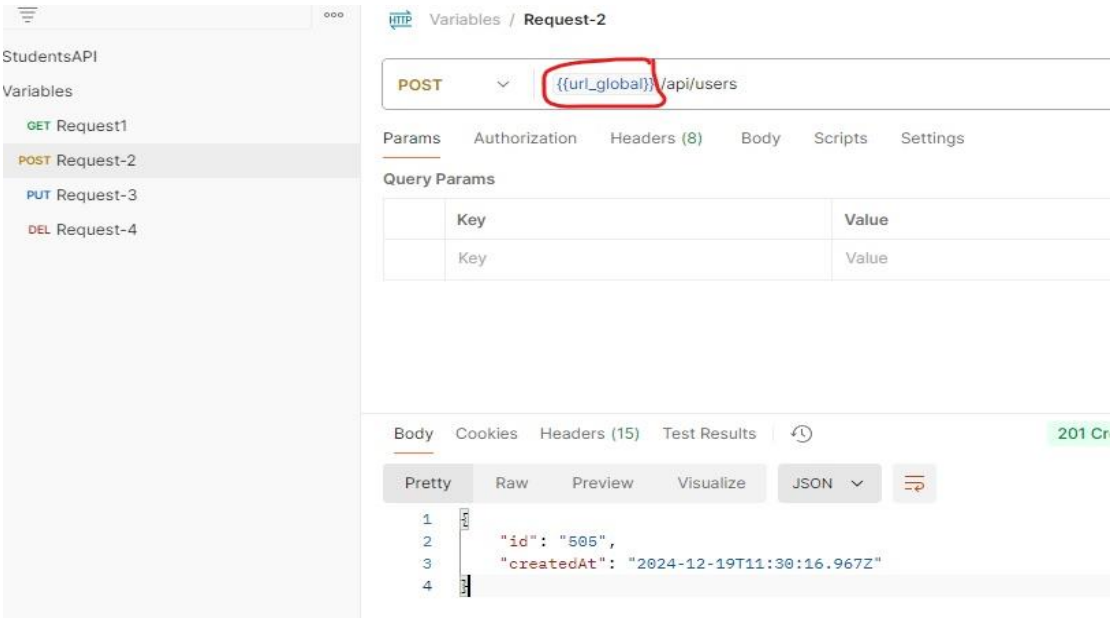


Step-3 ----(Get Request)

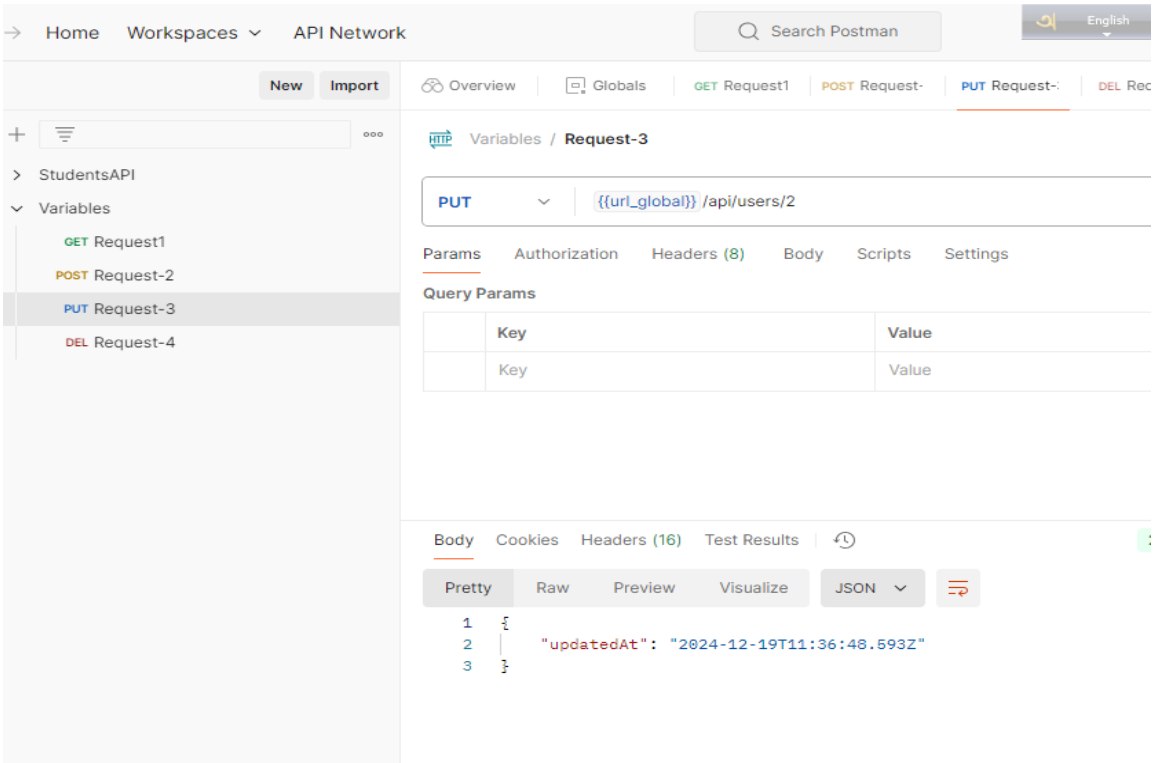


Step-5 ----(Post Request)

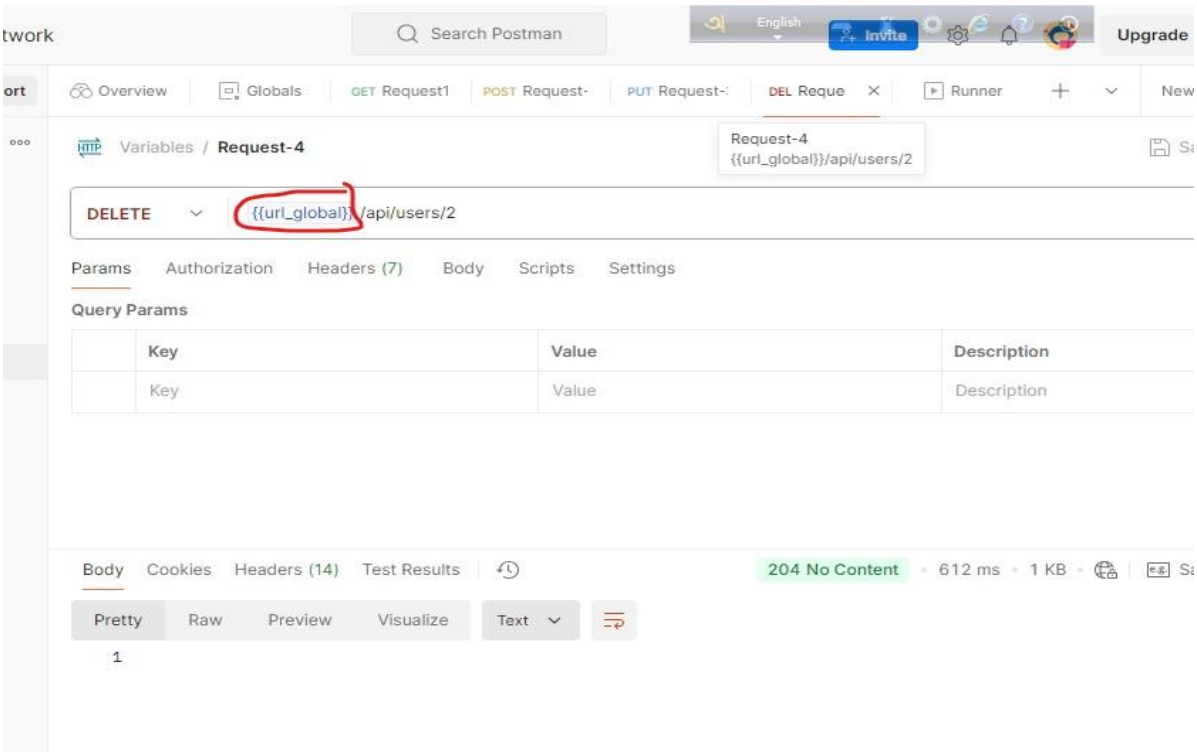
Put this " {{url_global}} " variable in every collection accordingly to change the value



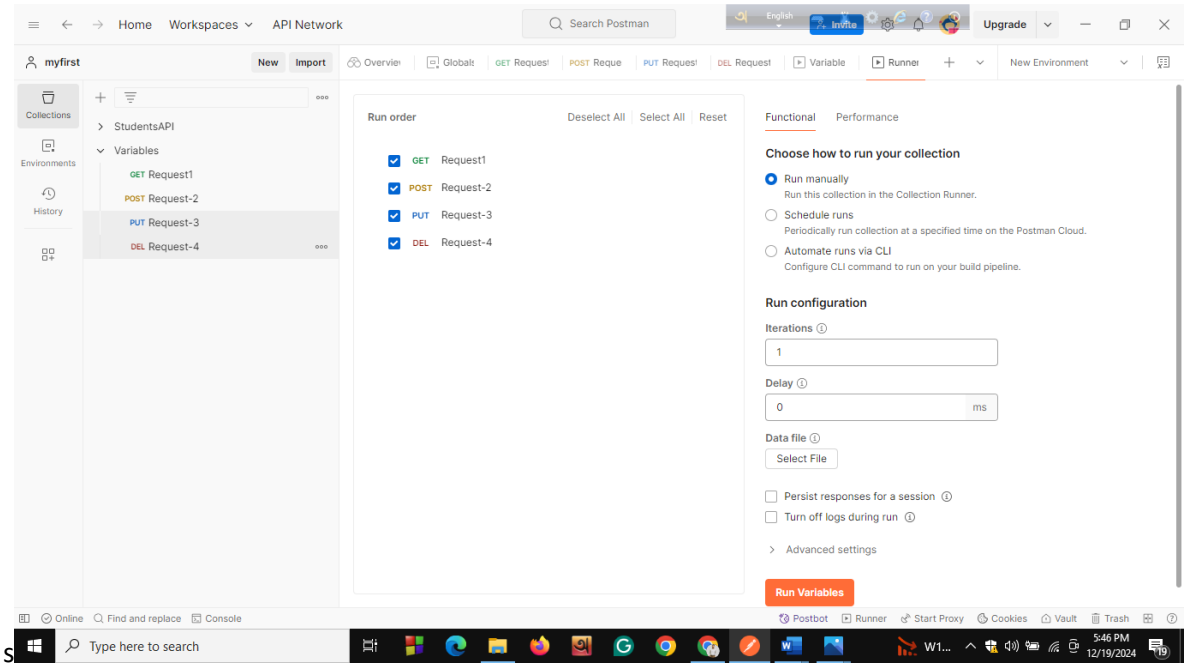
Step-6 (Put Request)



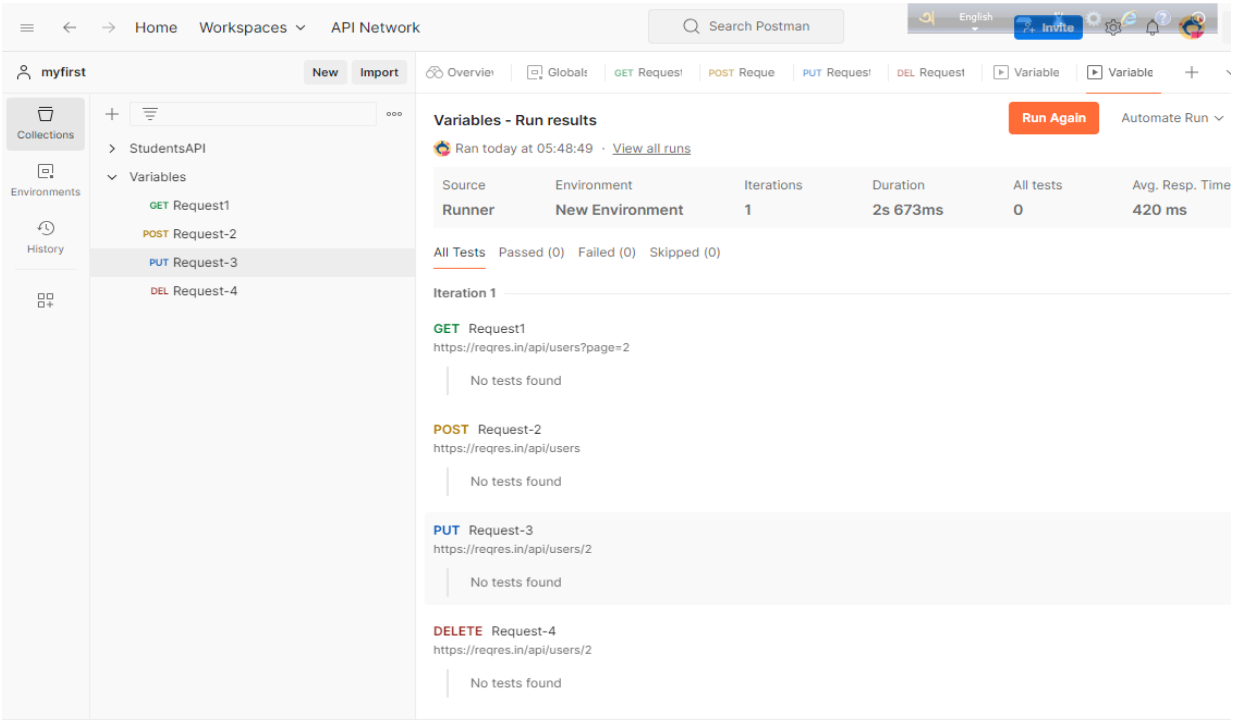
Step-7 ----(Delete Request)



Step-8

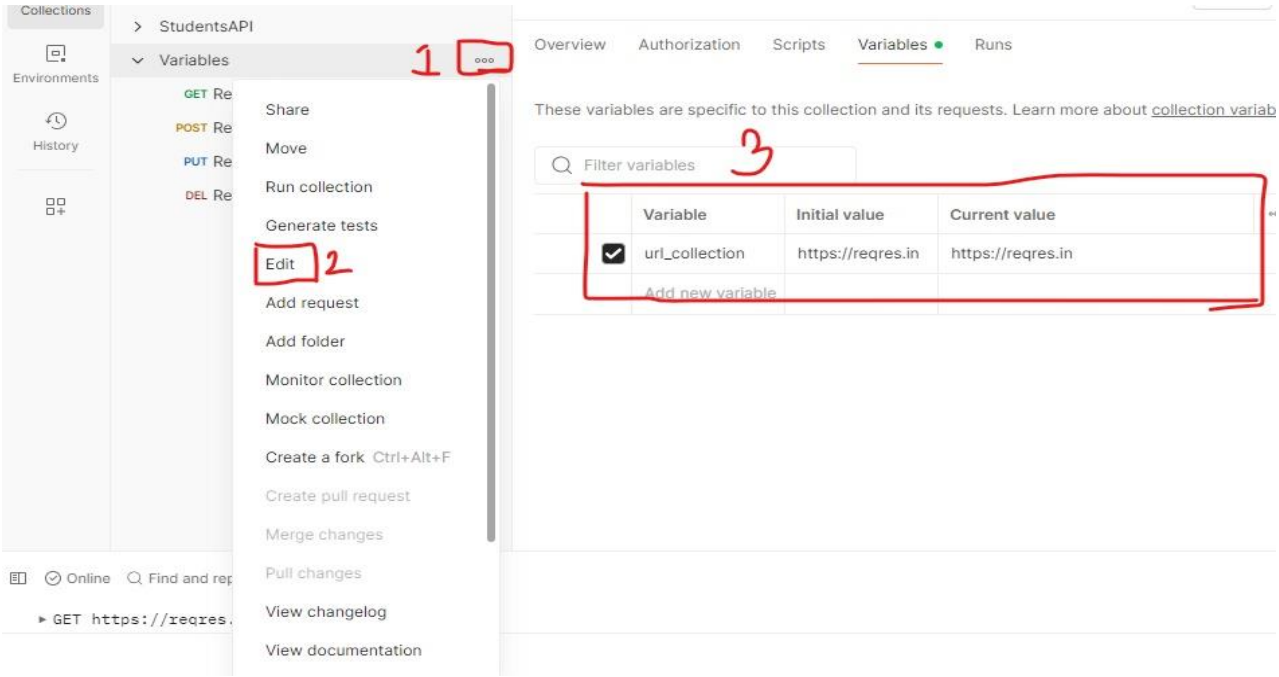

Home
Workspaces
API Network
Search Postman
Upgrade
myfirst
New
Import
Collections
Environments
History
StudentsAPI
Variables
GET Request1
POST Request-2
PUT Request-3
DEL Request-4
Run order
Deselect All
Select All
Reset
Functional
Performance
Choose how to run your collection
Run manually
Run this collection 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
1
Delay
0
ms
Data file
Select File
Persist responses for a session
Turn off logs during run
Advanced settings
Run Variables
Type here to search

Step-9

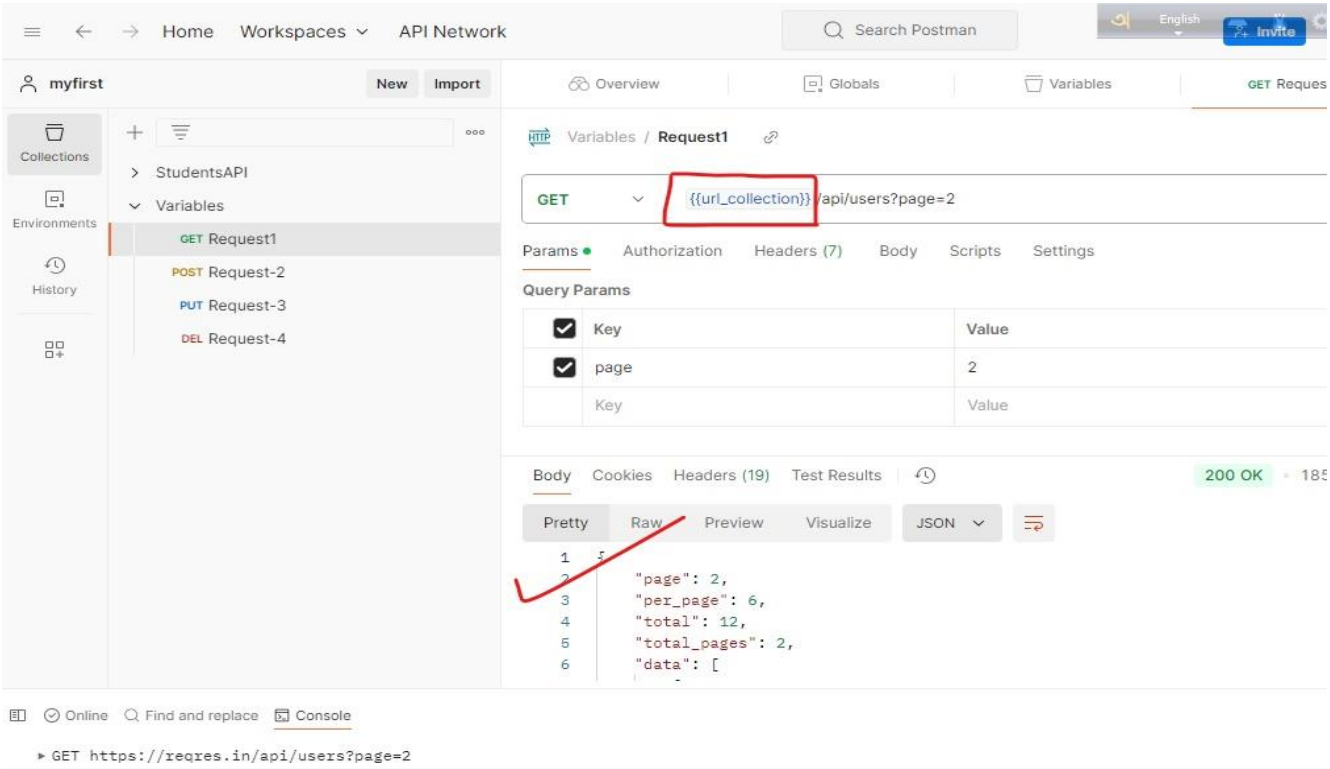

Home
Workspaces
API Network
Search Postman
myfirst
New
Import
Collections
Environments
History
StudentsAPI
Variables
GET Request1
POST Request-2
PUT Request-3
DEL Request-4
Variables - Run results
Run Again
Automate Run
Ran today at 05:48:49 · View all runs
Source
Runner
Environment
New Environment
Iterations
1
Duration
2s 673ms
All tests
0
Avg. Resp. Time
420 ms
All Tests
Passed (0)
Failed (0)
Skipped (0)
Iteration 1
GET Request1
https://reqres.in/api/users?page=2
No tests found
POST Request-2
https://reqres.in/api/users
No tests found
PUT Request-3
https://reqres.in/api/users/2
No tests found
DELETE Request-4
https://reqres.in/api/users/2
No tests found

<u>Collection variable</u> : Collection variable is accessible within the Collection among multiple requests.

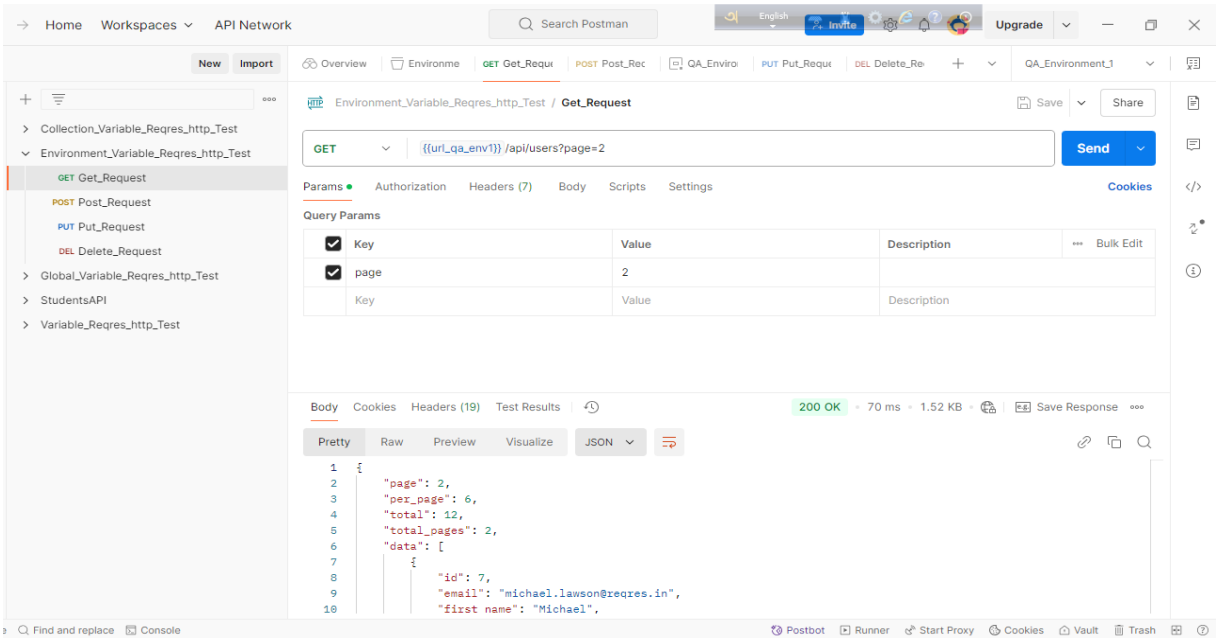
Step-1



Step-2

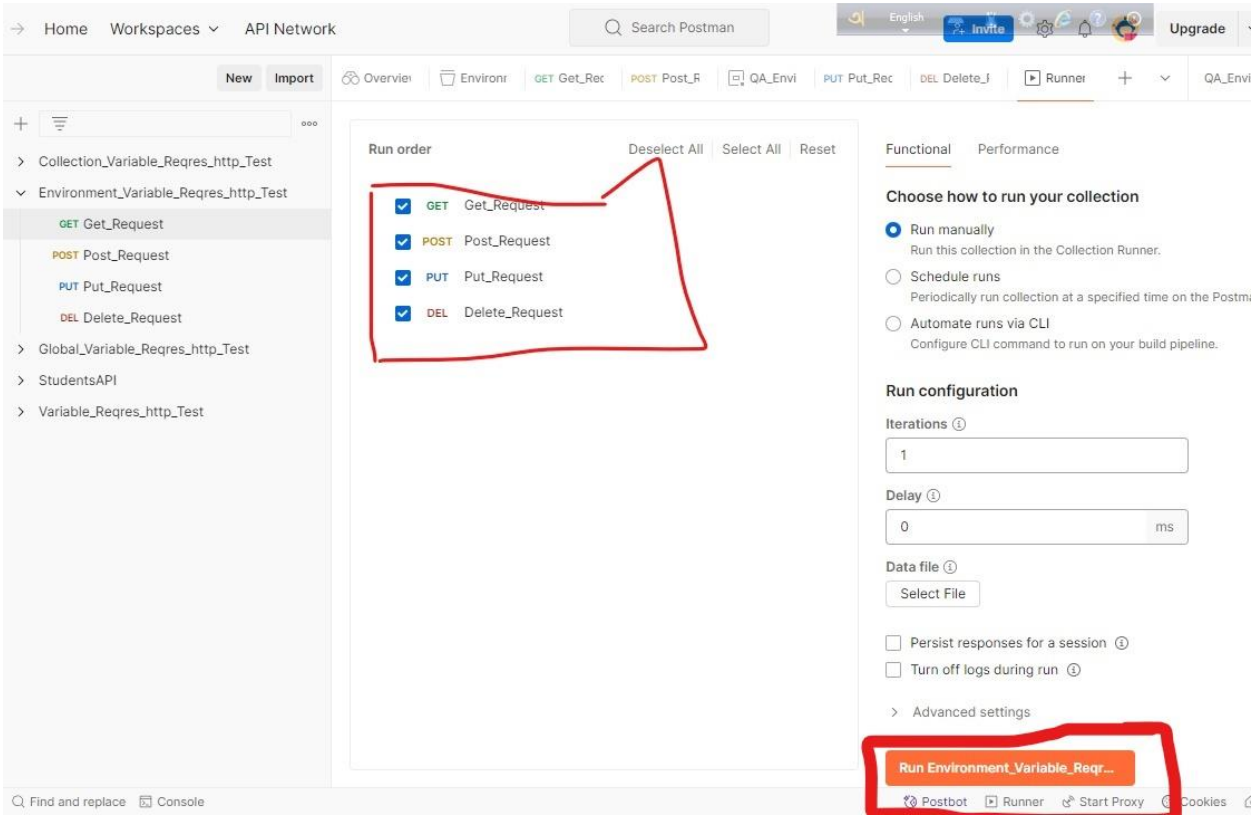


Environment Variable: Accessible in all collections but we use it for a specific environment

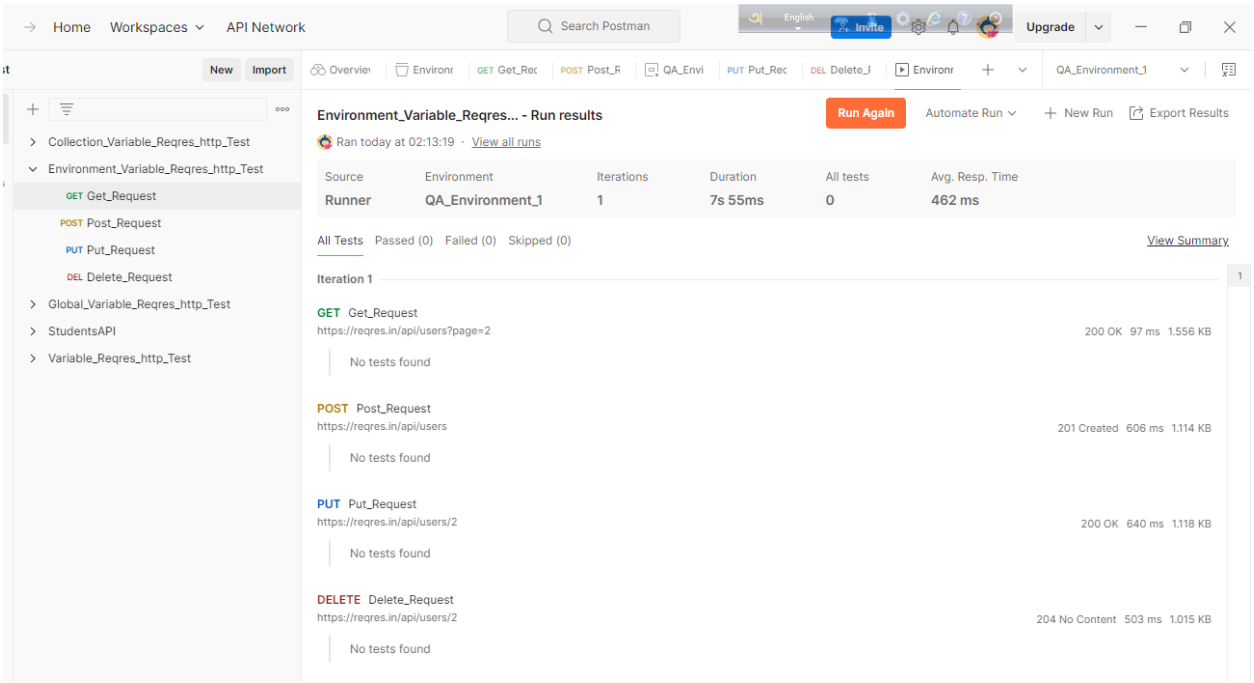


Run the whole collection of Environment Variable:

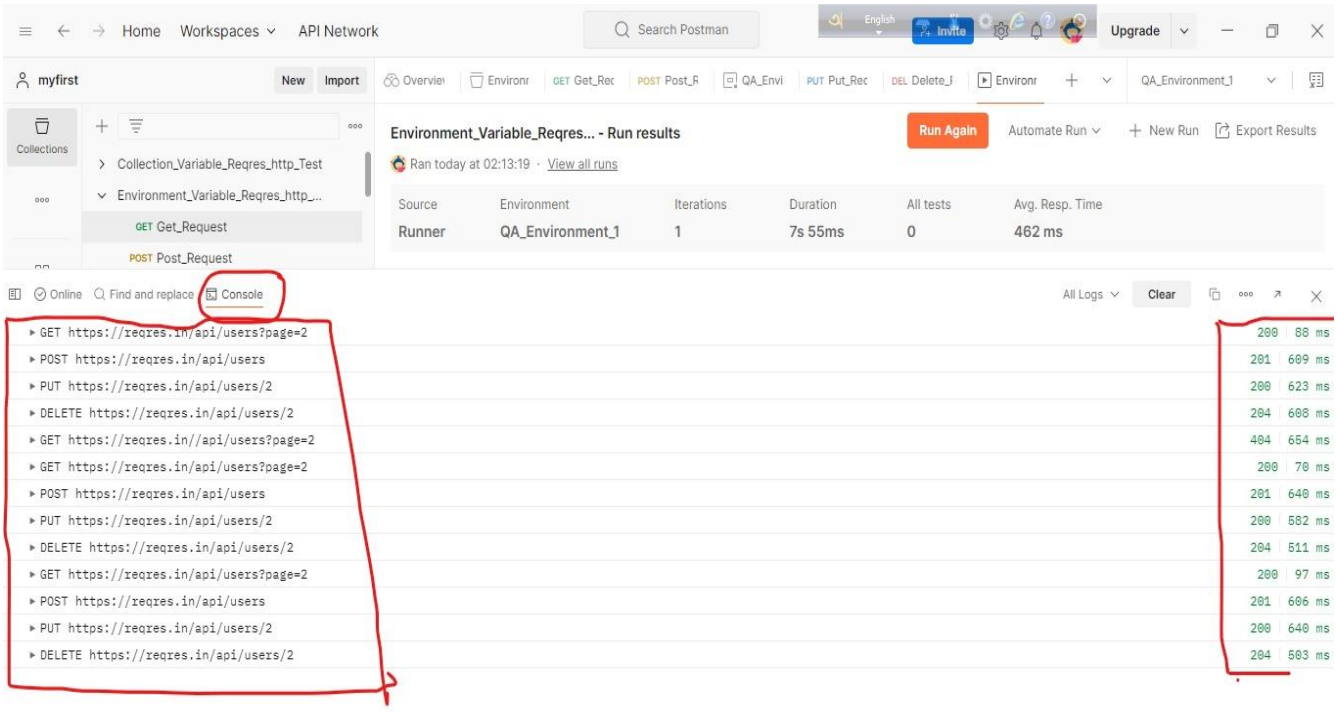
Step-1



Step-2



Console result

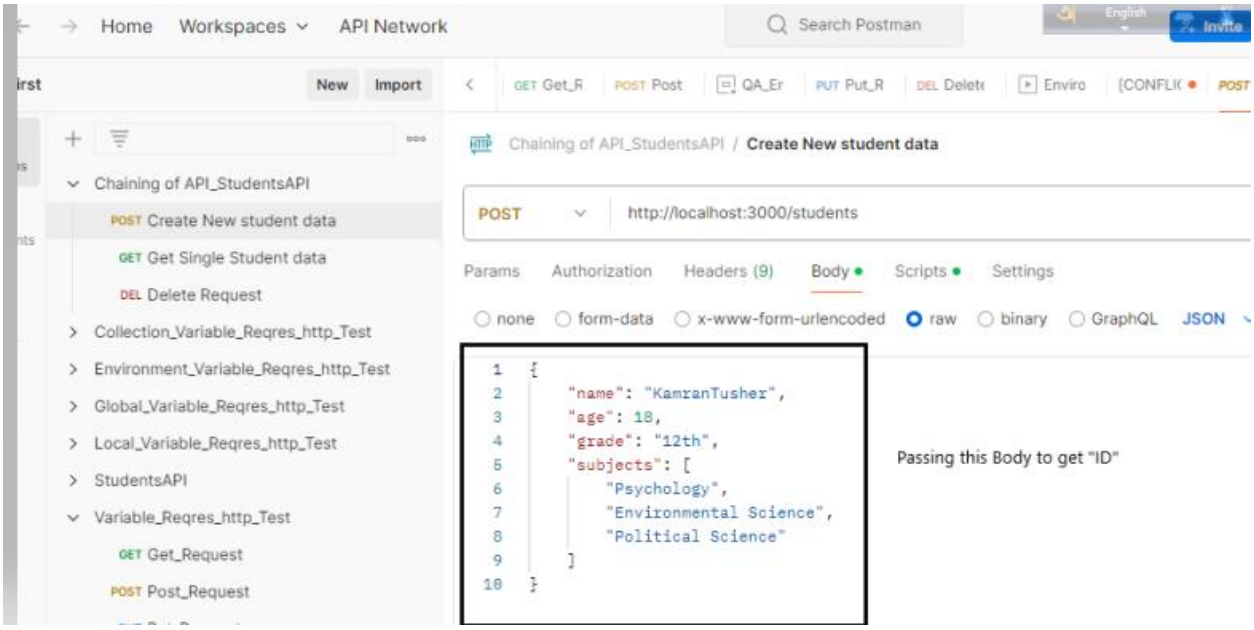


<u>Chaining of API</u> : The response of one API becomes the request of another API is called Chaing of API.

Task:

When I pass a body it will create an ‘ID’ in response to request from the body..Then this ‘ ID ‘ will stored in a variable.

<u>Step-1) Pass this body</u>



<u>Step-2</u>

Write a test script in the “ post-response “ body.

<u>Script</u>

var JsonData= JSON.**parse**(ResponseBody);

pm.environment("id"."JsonData.id");

Here this particular script will set an environment and give a id for the particular body which is given on the top.