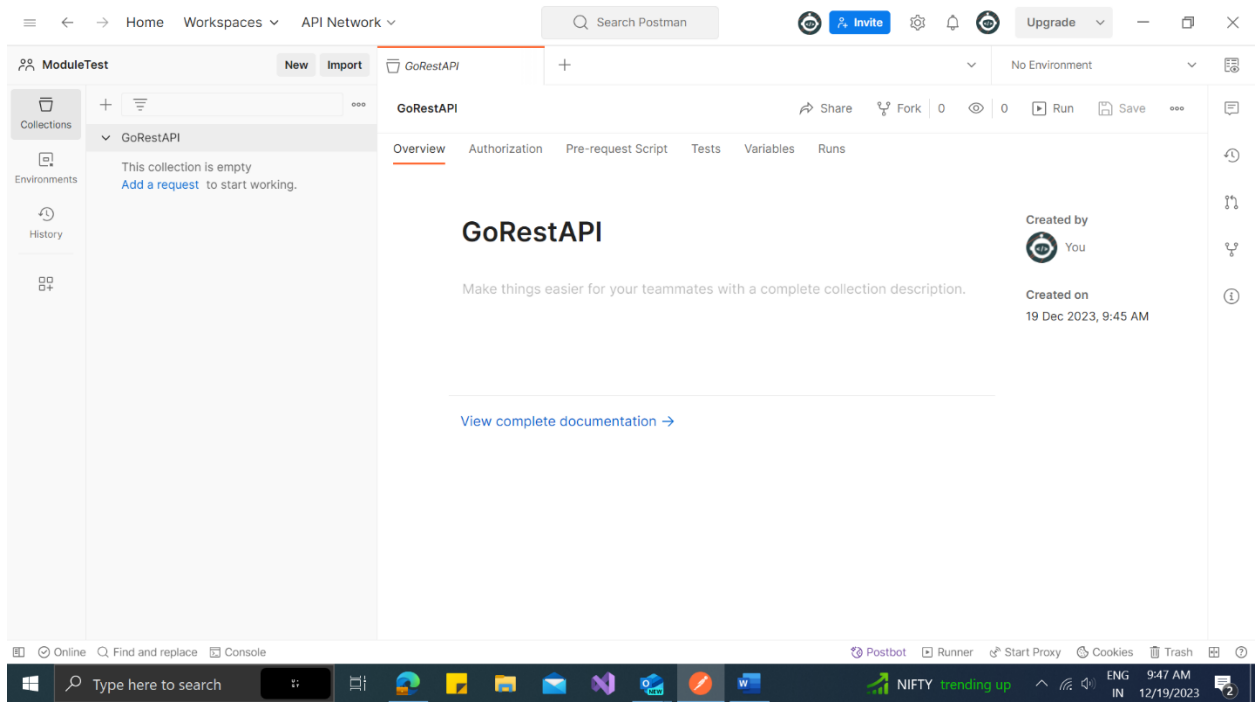
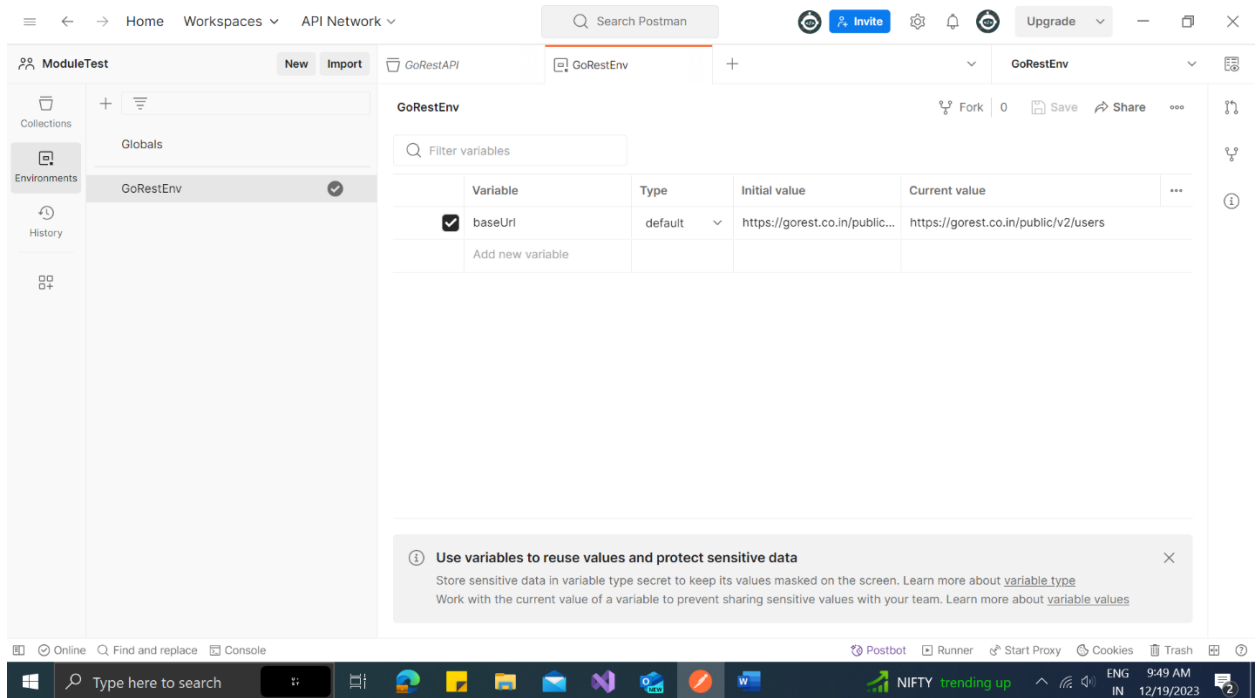


GoRestAPI

1. Creating new collection “GoRestAPI”



2. Creating an Environment called “GoRestEnv” with variable “baseUrl”



3. GET – GetALLUsers response body

The screenshot shows the Postman interface for a GET request to the endpoint `GoRestAPI / GetAllUsers`. The request is configured with the method `GET` and the URL `{{baseUrl}}`. The response status is `200 OK` with a response time of `426 ms` and a size of `2.41 KB`. The response body is displayed in the `Body` tab, showing a JSON array of two user objects. The first user has an `id` of `5839905`, `name` of `"Dr. Puneet Shukla"`, `email` of `"dr_puneet_shukla@hoeger-mueller.example"`, `gender` of `"male"`, and `status` of `"inactive"`. The second user has an `id` of `5839903`, `name` of `"Tanushri Devar"`, `email` of `"tanushri_devar@treutel-mcglynn.example"`, `gender` of `"male"`, and `status` of `"inactive"`.

```
1 {
2   {
3     "id": 5839905,
4     "name": "Dr. Puneet Shukla",
5     "email": "dr_puneet_shukla@hoeger-mueller.example",
6     "gender": "male",
7     "status": "inactive"
8   },
9   {
10    "id": 5839903,
11    "name": "Tanushri Devar",
12    "email": "tanushri_devar@treutel-mcglynn.example",
13    "gender": "male",
14    "status": "inactive"
15  }
16 }
```

4. GET – GetALLUsers Tests

The screenshot shows the Postman interface for the same GET request to the endpoint `GoRestAPI / GetAllUsers`. The `Tests` tab is selected, showing two test scripts. The first test checks if the status code is 200, and the second test checks if the status code name has the string "OK". Both tests are marked as `PASS`. The response status is `200 OK` with a response time of `426 ms` and a size of `2.41 KB`.

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Status code name has string OK", function () {
5   pm.response.to.have.status("OK");
6 });
```

Test Results (2/2)

All	Passed	Skipped	Failed
	PASS		
	PASS		

Status code is 200

Status code name has string OK

5. GET – GetSingleUser response body

The screenshot shows the Postman interface for the `GET /GetSingleUser` endpoint. The URL is `{{baseUrl}}/5838996`. The response body is displayed in the `Body` tab, showing a JSON object with the following fields:

```
1 {
2   "id": 5838996,
3   "name": "Amritambu Nair PhD",
4   "email": "nair_amritambu_phd@hahn-roberts.test",
5   "gender": "female",
6   "status": "active"
7 }
```

The status bar at the bottom indicates a `200 OK` response with a response time of `300 ms` and a body size of `1.09 KB`.

6. GET – GetSingleUser Tests

The screenshot shows the Postman interface for the `GET /GetSingleUser` endpoint with tests defined. The tests are written in JavaScript and are located in the `Tests` tab:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Id value check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.id).to.eql(5838996);
7 });
```

The test results are displayed in the `Test Results (2/2)` tab, showing that both tests passed:

- PASS** Status code is 200
- PASS** Id value check

The status bar at the bottom indicates a `200 OK` response with a response time of `340 ms` and a body size of `1.09 KB`.

7. POST – CreateUser Response body

The screenshot shows the Postman interface for the **POST /CreateUser** endpoint. The request body is a JSON object with the following structure:

```
1 {
2   "name": "Puni",
3   "email": "Puni@gmail.com",
4   "gender": "female",
5   "status": "active"
6 }
```

The response body is also in JSON format, showing the created user details:

```
1 {
2   "id": 5839417,
3   "name": "Puni",
4   "email": "Puni@gmail.com",
5   "gender": "female",
6   "status": "active"
7 }
```

The status bar at the bottom indicates: 201 Created, 785 ms, 1.16 KB.

8. POST – CreateUser Authorization

The screenshot shows the Postman interface for the **POST /CreateUser** endpoint with the **Authorization** tab selected. It provides instructions on how to obtain an Access Token from the Go REST API.

Get your Access Token for Go REST API

1. Go to your [Go REST Access Tokens Page](#)
2. Create a new **Access Token** (or choose an existing one)
3. Copy the **Token**, and paste it here

Access Token
Visit [gorest.co.in/my-account/access-tokens](#) to get your Access Token.

7db0d51d7686fe5d2683a3fa99c859f8e05f34cbbc75 ✓

[Save authorization to collection](#) ⓘ

Type: Bearer ... Token: 7db0d51d7686fe5d2683a3fa99c859f8e05f34cbbc75

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization

The status bar at the bottom indicates: 201 Created, 544 ms, 1.16 KB.

9. POST – CreateUser Tests

The screenshot shows the Postman interface for a POST request to `GoRestAPI /CreateUser`. The request is configured with the following details:

- Method:** POST
- URL:** `{{baseUrl}}`
- Params:** None
- Authorization:** None
- Headers (10):** None
- Body:** None
- Pre-request Script:** None
- Tests:**

```
1 pm.test("Status code is 201", function () {
2   pm.response.to.have.status(201);
3 });
4 pm.test("Name value check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.name).to.eql("Puni");
7 });
```
- Settings:** None
- Environment:** GoRestEnv

The test results show two passed tests:

- PASS:** Status code is 201
- PASS:** Name value check

The response body is `201 Created`, with a response time of `785 ms` and a size of `1.16 KB`. The interface also shows a sidebar with collections, environments, and history, and a bottom status bar with system information.

10. PUT – UpdateUser Response Body

The screenshot shows the Postman interface for a PUT request to `GoRestAPI /UpdateUser`. The request is configured with the following details:

- Method:** PUT
- URL:** `{{baseUrl}}/5838997`
- Params:** None
- Authorization:** None
- Headers (10):** None
- Body:**

```
1 {
2   "name": "ArunRaj",
3   "email": "arunraj@gmail.com",
4   "gender": "male",
5   "status": "active"
6 }
```
- Pre-request Script:** None
- Tests:** None
- Settings:** None
- Environment:** GoRestEnv

The response body is `200 OK`, with a response time of `807 ms` and a size of `1.13 KB`. The response body is displayed in the "Pretty" view:

```
1 {
2   "email": "arunraj@gmail.com",
3   "name": "ArunRaj",
4   "gender": "male",
5   "status": "active",
6   "id": 5838997
7 }
```

The interface also shows a sidebar with collections, environments, and history, and a bottom status bar with system information.

11. PUT – UpdateUser Authorization

The screenshot shows the Postman interface for the 'ModuleTest' workspace. The 'GoRestAPI' collection is expanded, showing the 'PUT UpdateUser' endpoint. The request is configured with the method 'PUT' and the URL '({{baseUrl}})/5838997'. The 'Authorization' tab is selected, showing 'Bearer ...' as the type and a token '7db0d51d7686fe5d2683a3fa99c859f8e051...'. The status bar at the bottom indicates a successful response: '200 OK 807 ms 1.13 KB'.

12. PUT – UpdateUser Tests

The screenshot shows the Postman interface for the 'ModuleTest' workspace, focusing on the 'PUT UpdateUser' endpoint. The 'Tests' tab is selected, displaying the following test script:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Email value check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.email).to.eql("arunraj@gmail.com");
7 });
```

The status bar at the bottom indicates a successful response: '200 OK 706 ms 1.13 KB'. Below the status bar, the 'Test Results (2/2)' section shows two passed tests:

- PASS Status code is 200
- PASS Email value check

13. Patch – PartialUpdateUser Response Body

The screenshot shows the Postman interface for a PATCH request named 'PATCH PartialUpdateUser'. The request is configured with the URL 'GoRestAPI / PartialUpdateUser' and the method 'PATCH'. The body is set to 'JSON' and contains the following JSON payload:

```
1 {
2   "status": "inactive"
3 }
```

The response is displayed in the 'Body' tab, showing a 200 OK status with a response time of 432 ms and a body size of 1.13 KB. The response body is a JSON object:

```
1 {
2   "status": "inactive",
3   "id": 5838997,
4   "name": "ArunRaj",
5   "email": "arunraj@gmail.com",
6   "gender": "male"
7 }
```

14. Patch – PartialUpdateUser Authorization

The screenshot shows the Postman interface for the same PATCH request, but with the 'Authorization' tab selected. The authorization is set to 'Bearer Token'. The token field contains the value '7db0d51d7686fe5d2683a3fa99c859f8e051...'. Below the token field, there is a note: 'The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization'.

15. Patch – PartialUpdateUser Tests

The screenshot shows the Postman interface for a PATCH request named 'PATCH PartialUpdateUser'. The request is sent to the URL 'GoRestAPI / PartialUpdateUser' with the body 'PATCH PartialUpdateUser'. The 'Tests' tab is active, displaying the following test script:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Status value check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.status).to.eql("inactive");
7 });
```

The 'Test Results' tab shows two passed tests:

- PASS Status code is 200
- PASS Status value check

The status bar at the bottom indicates a 200 OK response with a response time of 849 ms and a body size of 114 KB.

16. DELETE – DeleteUser Response Body

The screenshot shows the Postman interface for a DELETE request named 'DEL DeleteUser'. The request is sent to the URL 'GoRestAPI / DeleteUser' with the body 'DEL DeleteUser'. The 'Body' tab is active, displaying the message 'This request does not have a body'. The 'Test Results' tab shows two passed tests:

- PASS Status code is 204
- PASS Status value check

The status bar at the bottom indicates a 204 No Content response with a response time of 413 ms and a body size of 903 B.

17. DELETE – DeleteUser Authorization

The screenshot shows the Postman interface for the **GoRestAPI / DeleteUser** endpoint. The **Authorization** tab is selected, showing a **Bearer ...** token type and a token value: `7db0d51d7686fe5d2683a3fa99c859f8e05l...`. The **Params** tab shows the URL `{{baseUrl}}/5839409`. The **Send** button is visible. The status bar at the bottom indicates a **204 No Content** response with a response time of **413 ms** and a body size of **903 B**.

18. DELETE – DeleteUser Tests

The screenshot shows the Postman interface for the **GoRestAPI / DeleteUser** endpoint. The **Tests** tab is selected, showing the following test script:

```
1 pm.test("Status code is 204", function () {
2   pm.response.to.have.status(204);
3 });
4
5 pm.test("Status code name has string No Content", function () {
6   pm.response.to.have.status("No Content");
7 });
```

The **Test Results (2/2)** section shows two passed tests:

- PASS** Status code is 204
- PASS** Status code name has string No Content

The status bar at the bottom indicates a **204 No Content** response with a response time of **413 ms** and a body size of **903 B**.