Sample Json Schema validation using Postman Test script:

URI: https://gorest.co.in/public/v2/users/

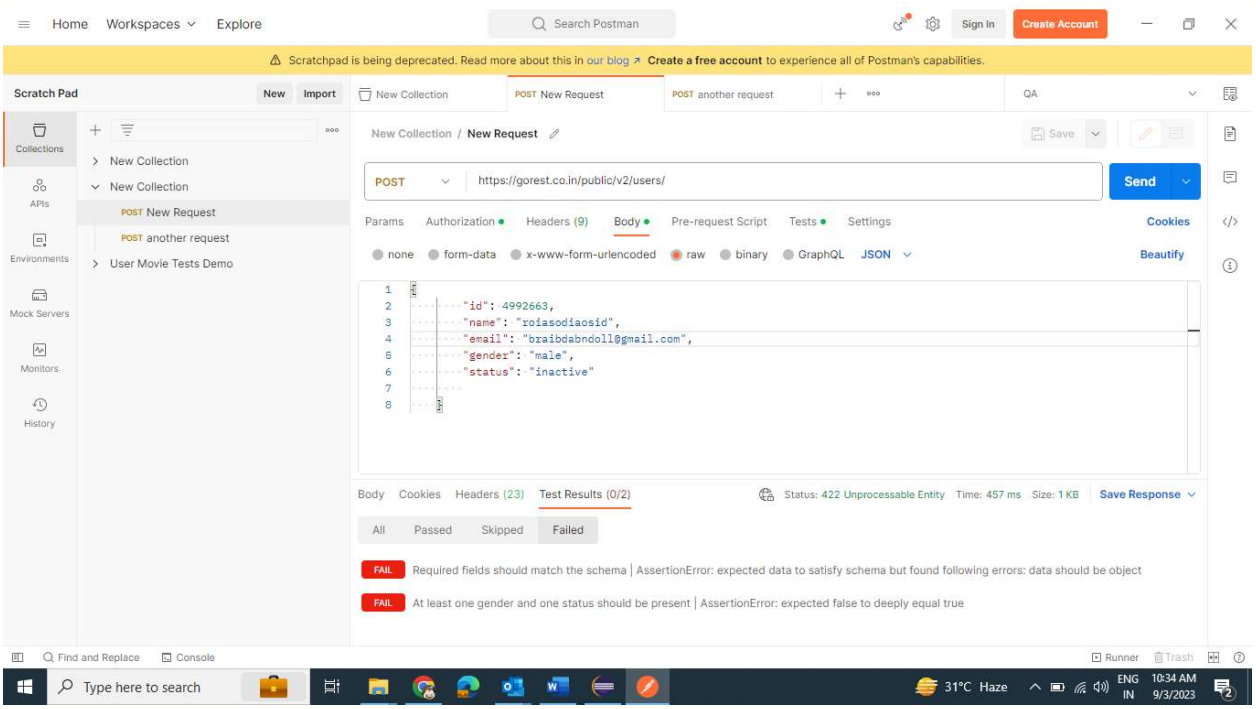
Authorization: only bearer token needed (except GET method). To fetch use this website gorest.co.in

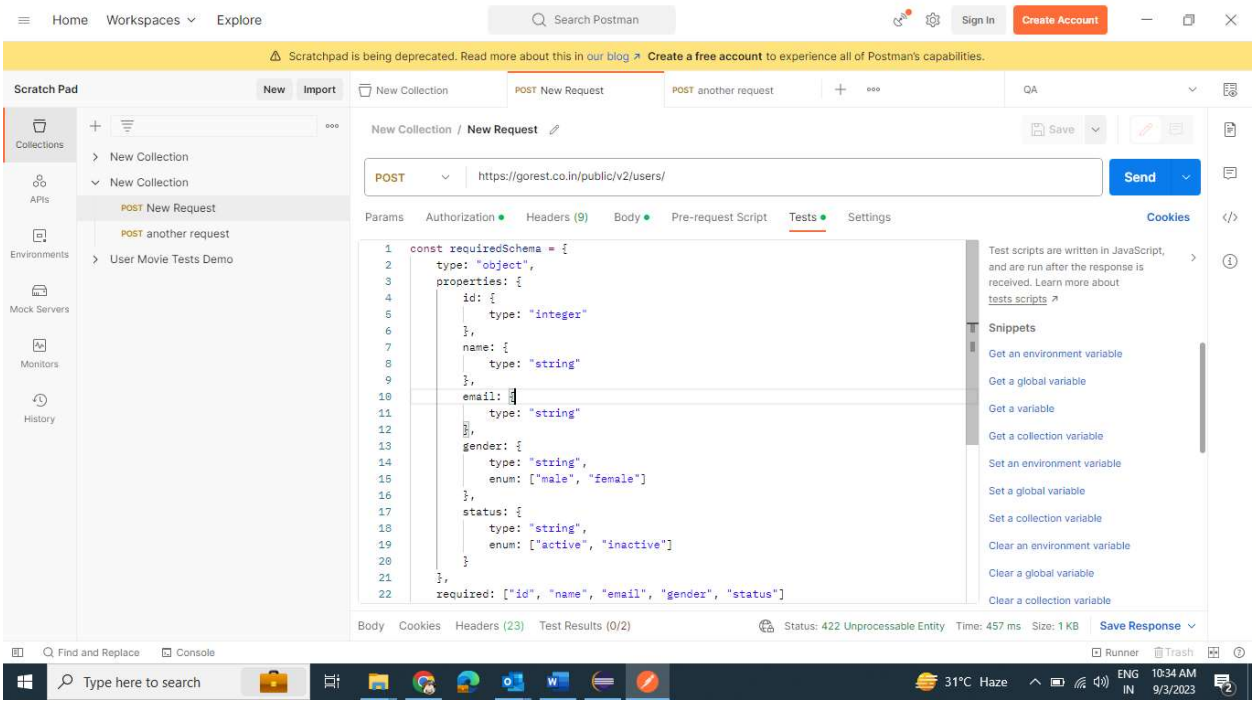
Method : POST

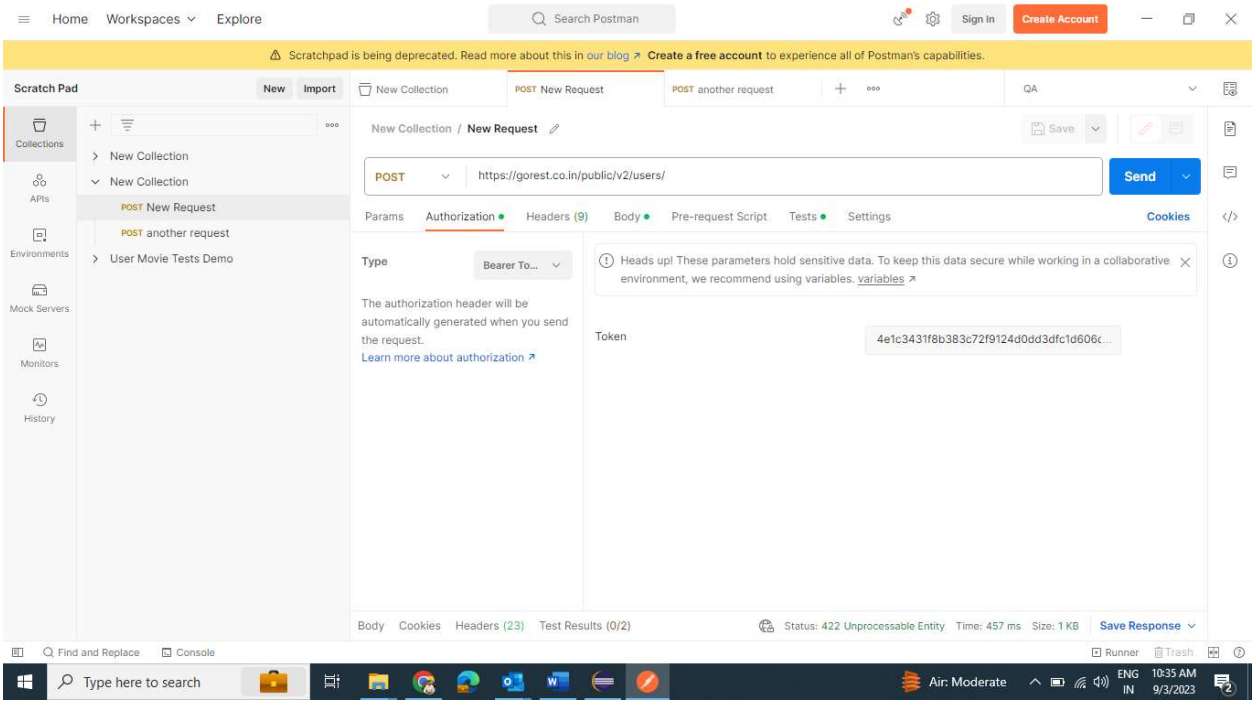
Request Body: *id is incremental (integer) , name should be in string, email should be in string, gender (male or female) can be passed only among two of these. And same for status (active and inactive) , and each time the email should be unique for incremental user.*

```
{
    "id": 4992663,
    "name": "roiasodiaosid",
    "email": "braibdabbbndoll@gmail.com",
    "gender": "male",
    "status": "inctive"

}
```







Test case 1: To verify the datatype of each key’s value

Test Case 2: Verify Default Values

Test Description: Ensure that when retrieving user data through an API, the default values for gender and status are as expected.

Steps:

Send an API request to retrieve user data.

Check the response JSON to verify that the default gender and status values are as expected.

Test Case 3: Update Gender

Test Description: Test the API endpoint that allows updating a user's gender.

Steps:

Send an API request to update the user's gender to a new value (e.g., "female").

Send another API request to retrieve the user's data.

Verify that the updated gender value matches the new value ("female").

Test Case 4: Update Status

Test Description: Test the API endpoint that allows updating a user's status.

Steps:

Send an API request to update the user's status to a new value (e.g., "active").

Send another API request to retrieve the user's data.

Verify that the updated status value matches the new value ("active").

Test Case 5: Simulate Invalid Gender Update

Test Description: Test the API to ensure that invalid values for gender are not accepted.

Steps:

Send an API request to update the user's gender to an invalid value (e.g., "unknown").

Verify that the API responds with an error or does not update the gender field.

Send an API request to retrieve the user's data.

Verify that the gender value remains unchanged.

Test Case 6: Simulate Invalid Status Update

Test Description: Test the API to ensure that invalid values for status are not accepted.

Steps:

Send an API request to update the user's status to an invalid value (e.g., "pending").

Verify that the API responds with an error or does not update the status field.

Send an API request to retrieve the user's data.

Verify that the status value remains unchanged.

Test script :

```
// Define a JSON schema for the expected response structure
const requiredSchema = {
    type: "object",                  // Specifies that the response should be an object.
    properties: {                    // Defines the expected properties within the object.
        id: {
            type: "integer"          // Expects the "id" property to be an integer.
        },
        name: {
            type: "string"           // Expects the "name" property to be a string.
        },
        email: {
            type: "string"           // Expects the "email" property to be a string.
        },
        gender: {
            type: "string",          // Expects the "gender" property to be a string.
            enum: ["male", "female"]  // Specifies that the "gender" property should be either "male" or "female
".
        },
        status: {
            type: "string",          // Expects the "status" property to be a string.
            enum: ["active", "inactive"] // Specifies that the "status" property should be either "active" or "i
nactive".
        }
    },
    required: ["id", "name", "email", "gender", "status"] // Specifies that these properties are required within
 the object.
};

// Validate only the required fields
pm.test("Required fields should match the schema", function () {
    // Use Postman's test assertions to validate the response JSON against the defined schema.
    pm.expect(pm.response.json()).to.have.jsonSchema(requiredSchema);
});

// Validate at least one gender and at least one status value
pm.test("At least one gender and one status should be present", function () {
    // Extract the response JSON from the API response.
    const responseJson = pm.response.json();

    // Check if the "gender" property in the response is "male" or "female."
    const hasMaleOrFemale = responseJson.gender === "male" || responseJson.gender === "female";

    // Check if the "status" property in the response is "active" or "inactive."
    const hasActiveOrInactive = responseJson.status === "active" || responseJson.status === "inactive";

    // Use Postman's test assertions to ensure that at least one gender and one status value are present.
```

```
    pm.expect(hasMaleOrFemale && hasActiveOrInactive).to.eql(true);
});
```