



API Testing Project

API Testing for a Public REST Service: End-to-End Scenarios
Using Rest Assured (Java) for automated API testing
And Postman, and Newman for manual API testing

Prepared by: Souad EL-Sayed

Mentor: Dr. Amany Shosha

Content

- Introduction
- Tools & technologies
- Project Features
- Manual Testing
- Automation Testing



Introduction



Go Rest

- **GoRest** (<https://gorest.co.in>) is a publicly available RESTful web service
- designed for learning, practicing, and testing API automation
- It provides endpoints for typical CRUD operations on resources like users, posts, comments, and todos.



Project Overview

- This project focuses on automating end-to-end **API scenarios** against GoRest's (Users , Posts and Todos) endpoints.
- covering creation, retrieval, update, and deletion of these resources
- validated through both **manual** testing (**Postman**) and **automated** testing (**Rest Assured** in Java).



Tools & technologies



Tools & technologies

- Postman
- Java script
- Newman
- IDE: IntelliJ IDEA
- Maven
- TestNG
- Rest Assured
- Java
- Allure



Project Features



Project Features

Testing Scope:

- User **authentication** for protected operations.
- **CRUD** operations for users, posts, and todos endpoints.
- **Positive** and **negative** test scenarios.
- **Validation** of response status codes, and Response structure.
- **Data-driven** execution using **JSON** files.



Project Features

Testing Approach:

- **Manual** verification via **Postman** collections.
- **Automated** execution via **Rest Assured**.
- **Reporting** and analysis through **Allure and Newman**.



Manual Testing



Manual Testing Using Postman



Manual Testing

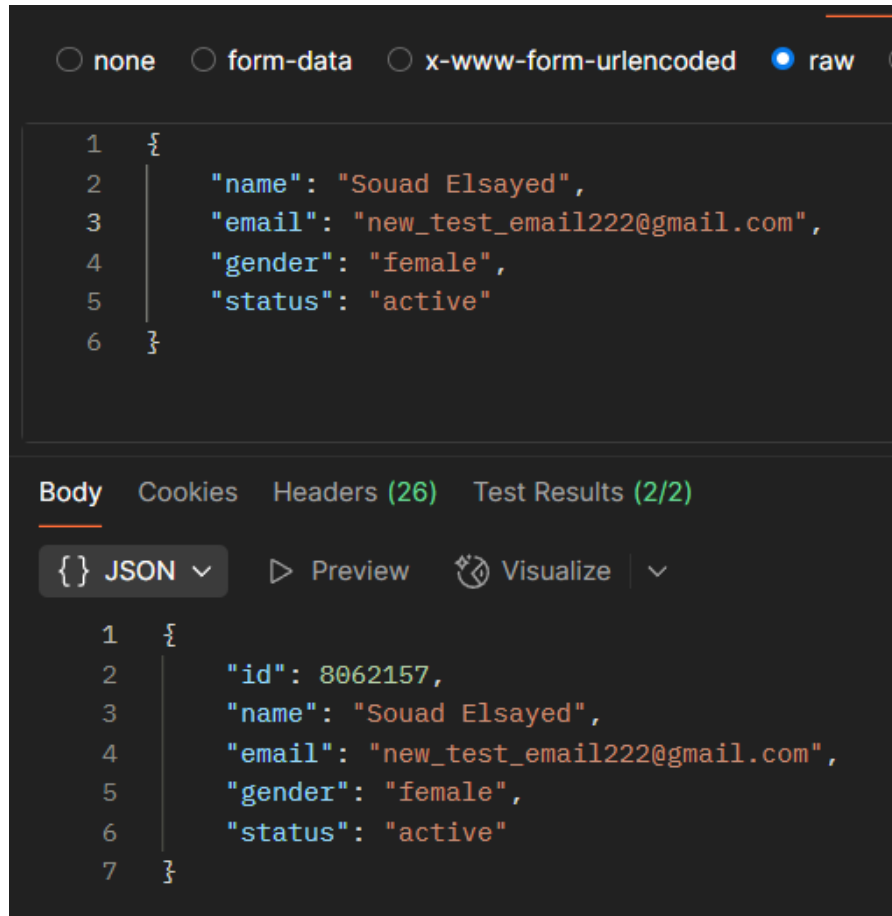
Key Capabilities of Postman for API Testing:

- Use **environment/global** variables for **dynamic** values and easy environment switching without manual URL edits.
- Execute collections with **CSV** File to test varied scenarios.
- Run Postman collections from the command line with **Newman** and make HTML Report



Manual Testing

create user with valid data :



The screenshot shows a REST client interface with the following details:

- Request:**
 - Method: `POST`
 - URL: `http://localhost:3000/users`
 - Body Type: `raw`
 - Body Content:

```
1 {
2   "name": "Souad Elsayed",
3   "email": "new_test_email222@gmail.com",
4   "gender": "female",
5   "status": "active"
6 }
```
- Response:**
 - Status: `201 Created`
 - Duration: `373 ms`
 - Body Type: `JSON`
 - Body Content:

```
1 {
2   "id": 8062157,
3   "name": "Souad Elsayed",
4   "email": "new_test_email222@gmail.com",
5   "gender": "female",
6   "status": "active"
7 }
```

201 Created • 373 ms • 1

PASSED Status code is 201

PASSED Response contains expected fields and values



Manual Testing

create user with Exist mail:

```
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○

1  {
2    "name": "Ahmed Elsayed",
3    "email": "new_unique3_email@gmail.com",
4    "gender": "male",
5    "status": "active"
6  }
```

Body Cookies Headers (24) Test Results (2/2)

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

```
1  [
2    {
3      "field": "email",
4      "message": "has already been taken"
5    }
6  ]
```

422 Unprocessable Entity

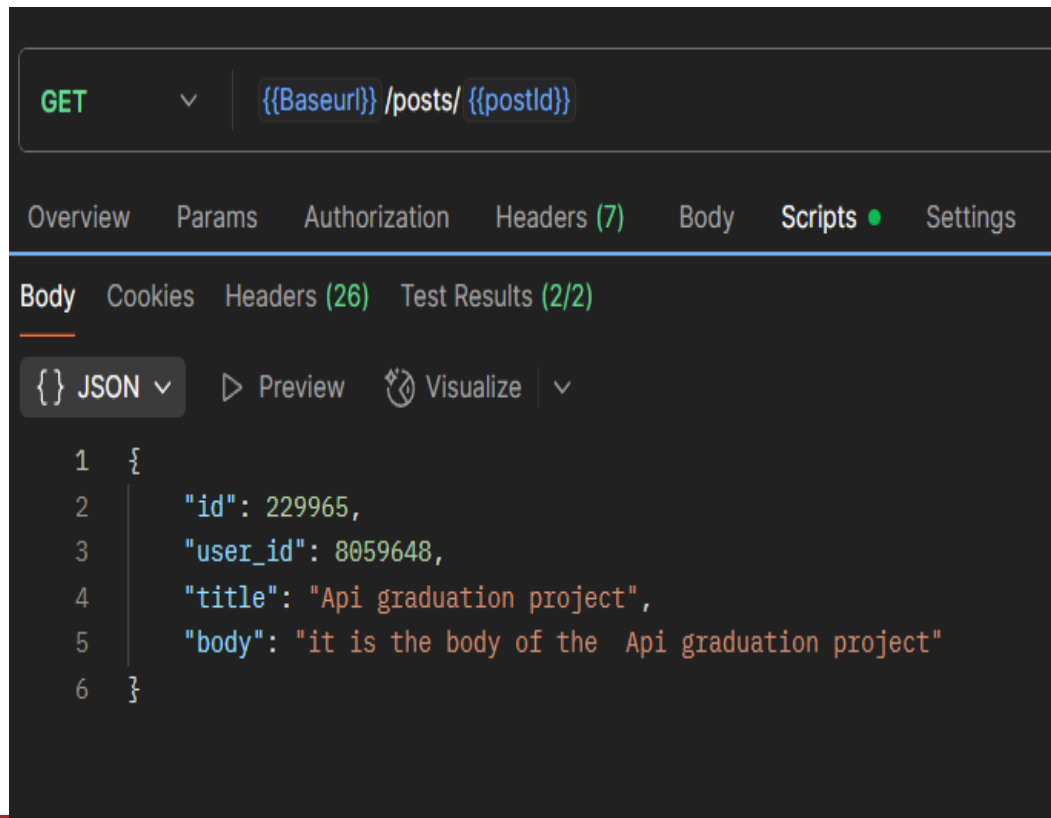
PASSED Status code is 422

PASSED Email already taken error is returned



Manual Testing

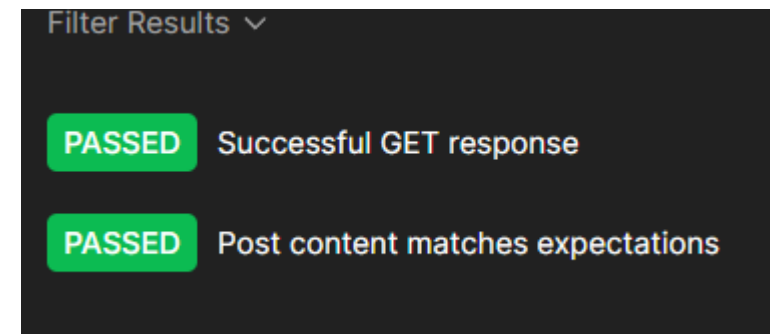
Get Post By ID:



The screenshot shows a REST client interface with a GET request to `{{Baseurl}} /posts/ {{postId}}`. The response is a JSON object with the following structure:

```
1 {  
2   "id": 229965,  
3   "user_id": 8059648,  
4   "title": "Api graduation project",  
5   "body": "it is the body of the  Api graduation project"  
6 }
```

200 OK • 320 ms • 1.



Filter Results ▾

- PASSED** Successful GET response
- PASSED** Post content matches expectations



Manual Testing

Update all data user by id:

```
1  {
2    "name": "Ahmed Elsayed",
3    "email": "UPDATE1_email@gmail.com",
4    "gender": "male",
5    "status": "active"
6  }
```

Body Cookies Headers (26) Test Results (2/2)

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

```
1  {
2    "email": "UPDATE1_email@gmail.com",
3    "name": "Ahmed Elsayed",
4    "gender": "male",
5    "status": "active",
6    "id": 8059648
7  }
```

```
PUT {{Baseurl}} /users/ {{userId}}
```

```
200 OK • 320 ms • 1.
```

PASSED Status code is 200

PASSED User updated successfully

Manual Testing

Update all data user by id:

```
1  {
2
3    "email": "ahmed@gmail.com"
4  }
```

Body Cookies Headers (26) Test Results

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

```
1  {
2    "email": "ahmed@gmail.com",
3    "id": 8059648,
4    "name": "Ahmed Elsayed",
5    "gender": "male",
6    "status": "active"
7  }
```

```
PATCH ▾ {{Baseurl}} /users/ {{userId}}
```

200 OK • 320 ms • 1.

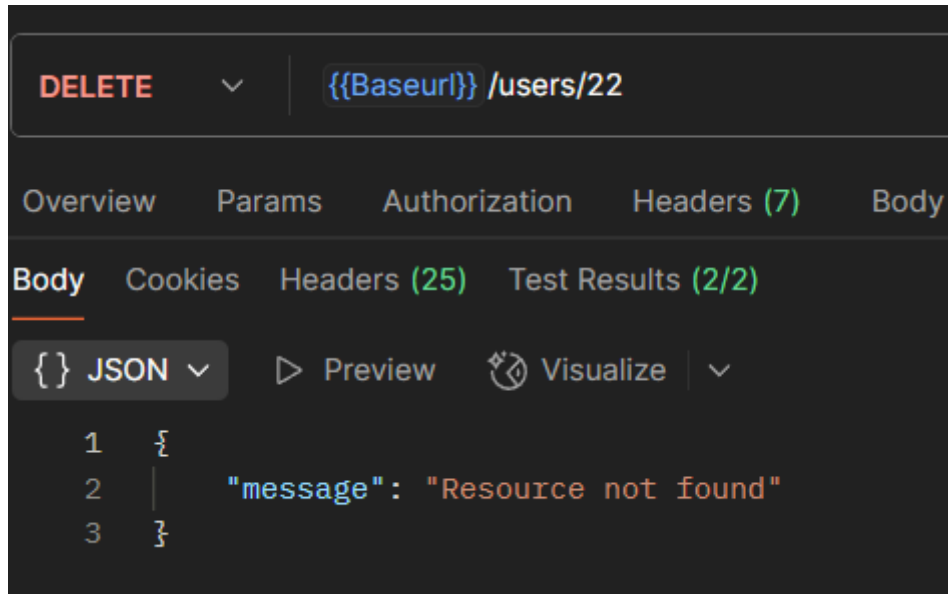
PASSED Status code is 200

PASSED User updated successfully



Manual Testing

delete by valid and invalid id:



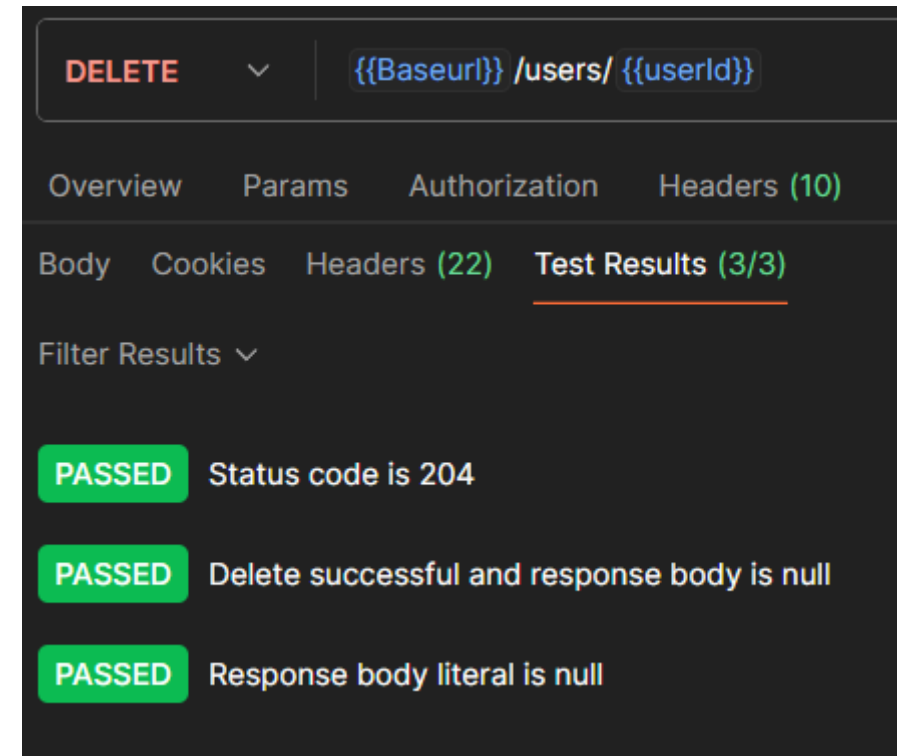
DELETE `{{Baseurl}}/users/22`

Overview Params Authorization Headers (7) Body

Body Cookies Headers (25) Test Results (2/2)

{ } JSON Preview Visualize

```
1 {
2   "message": "Resource not found"
3 }
```



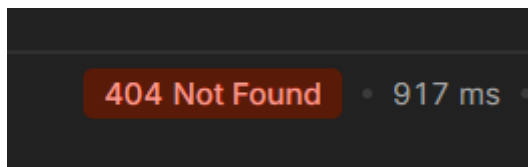
DELETE `{{Baseurl}}/users/{{userId}}`

Overview Params Authorization Headers (10)

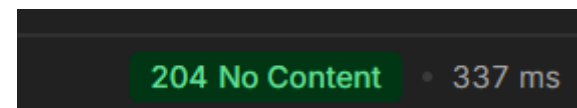
Body Cookies Headers (22) Test Results (3/3)

Filter Results

- PASSED Status code is 204
- PASSED Delete successful and response body is null
- PASSED Response body literal is null



404 Not Found • 917 ms •



204 No Content • 337 ms •

Manual Testing

Data-Driven Testing

user	Userr_id	Title_valid	body	invalid_id	invalid_title	invalid_body	update_title
Souad Elsayed	8059648	The Programmer's Best Friend	Every great coder knows that a cup of coffee fuels productivity. Hereâ€	1			update post title
Ahmed Elsayed	8060980	The Programmer's Best Friend	Every great coder knows that a cup of coffee fuels productivity. Hereâ€	2			update post title
Amany Elsayed	8060982	The Programmer's Best Friend	Every great coder knows that a cup of coffee fuels productivity. Hereâ€	3			update post title
salma Elsayed	8060984	The Programmer's Best Friend	Every great coder knows that a cup of coffee fuels productivity. Hereâ€	4			update post title
ali ahmed	8060986	The Programmer's Best Friend	Every great coder knows that a cup of coffee fuels productivity. Hereâ€	5			update post title

Run configuration

Iterations ⓘ

5



Manual Testing

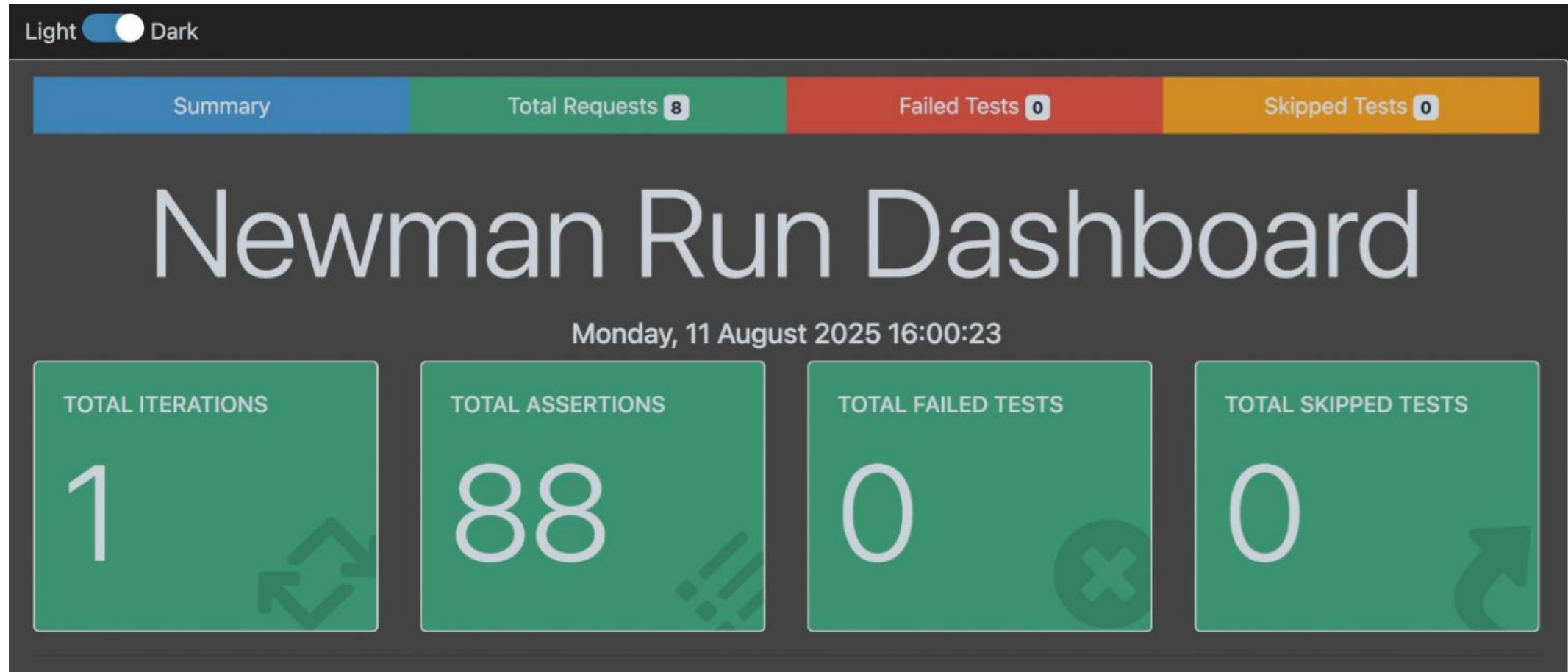
newman Run:

	executed	failed
iterations	1	0
requests	8	0
test-scripts	16	0
prerequest-scripts	11	0
assertions	88	0
total run duration: 1898ms		
total data received: 14.25kB (approx)		
average response time: 206ms [min: 141ms, max: 655ms, s.d.: 169ms]		



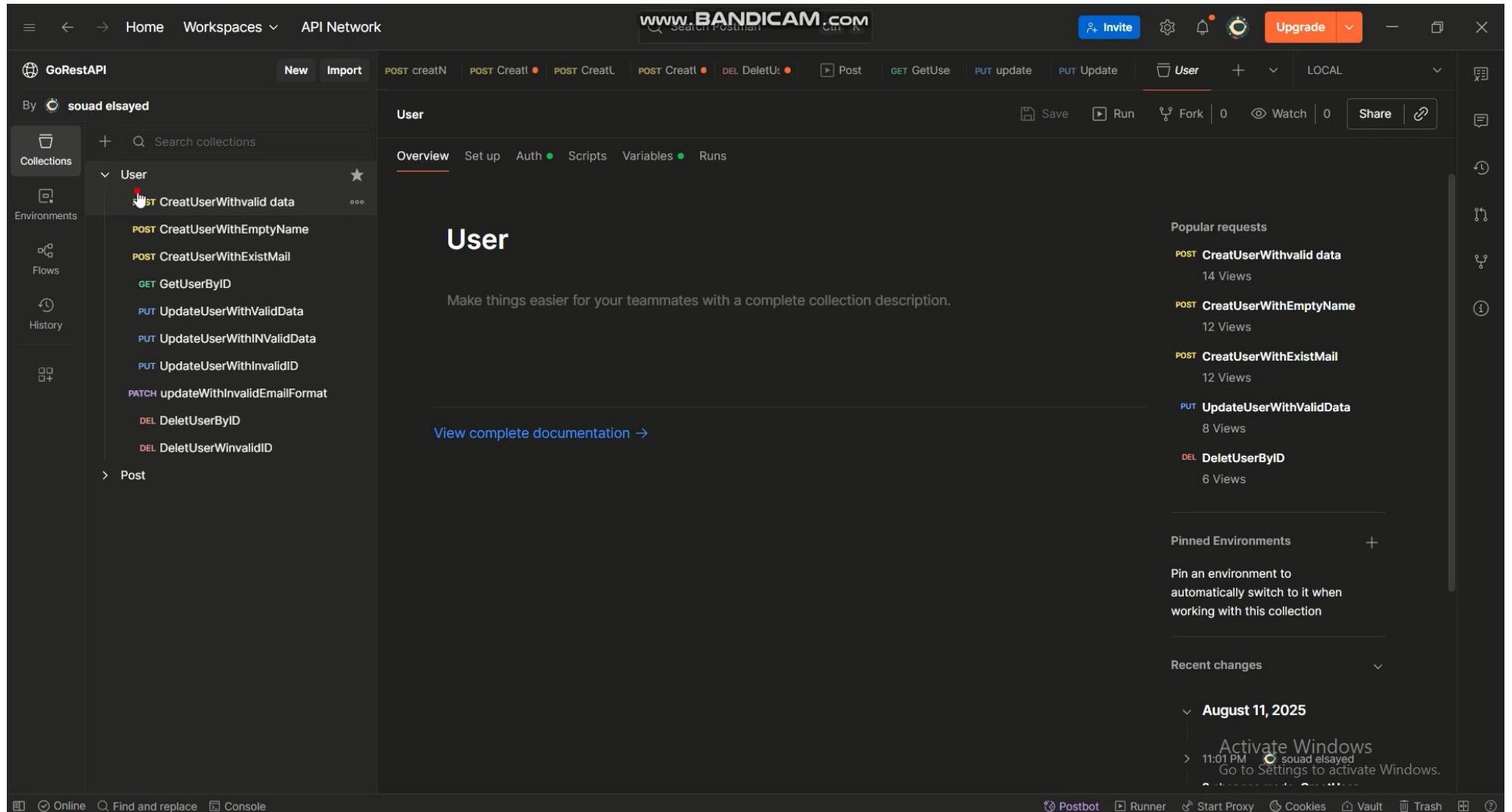
Manual Testing

Reporting:



Manual Testing

Demo:



Automation Testing Using RestAssured

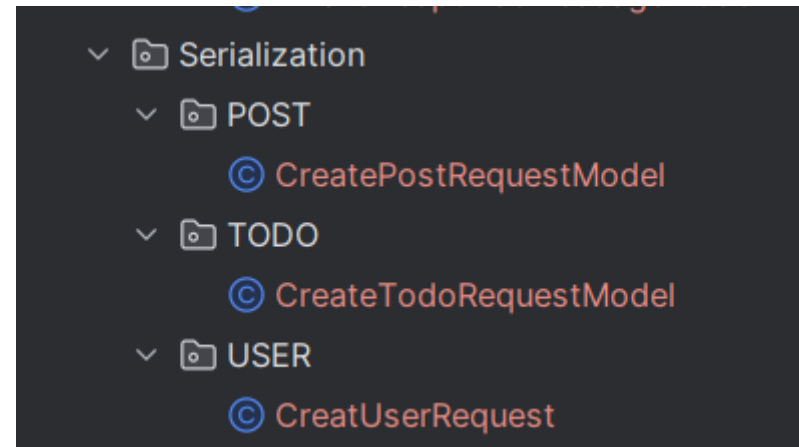
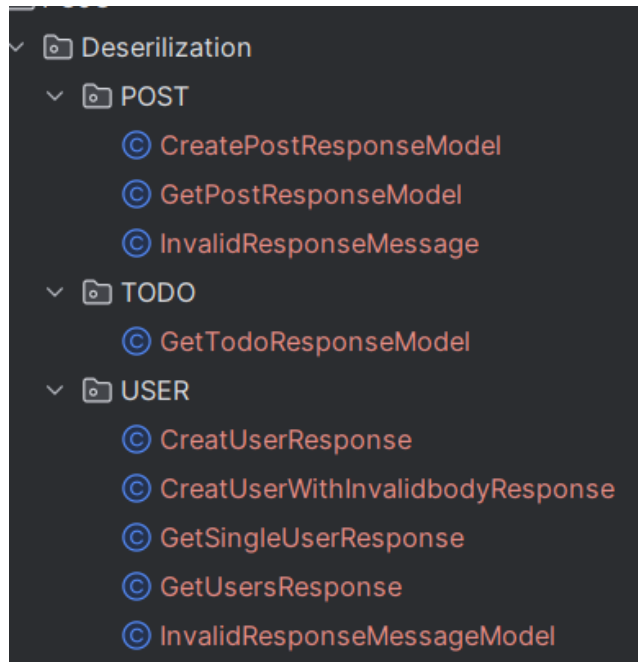


Automation Testing

1. Serialization & Deserialization

Used POJO classes to map API request and response bodies.

- Serialization: Converted Java objects into JSON for API requests.
- Deserialization: Converted API JSON responses back into Java objects for easy validation.



Automation Testing

1. Design Pattern & Code Structure

- Applied POM-inspired structure for API testing.
- Created builder and request classes for cleaner, reusable code.
- Reduced duplication and improved maintainability.



✓  Utils
© Builder



✓  Requests
© PostRequest
© TodoRequests
© UserReq



Automation Testing

Builder Class

```
// API Configuration
public static final String base_url = "https://gorest.co.in/public/v2";
public static final String userEndPoint = "/users/"; 18 usages
public static final String POST_ENDPOINT = "/posts/"; 10 usages
public static final String TODO_ENDPOINT = "/todos/"; 4 usages
```

```
// Test Data Constants
public static final String DEFAULT_USERNAME = "souad"; 5 usages
public static final String DEFAULT_EMAIL = faker.internet().emailAddress();
public static final String DEFAULT_EMAIL_de = "User2@email.com"; no usages
```

```
// Default Headers
public static Map<String, String> getHeaders() { 25 usages
    Map<String, String> headers = new HashMap<>();
    String token = System.getenv(name: "API_TOKEN"); //set API_TOKEN=your_token_here

    if(token == null || token.isEmpty()) {
        token = "e94c4e333d0d73d8e52e5243552c76a5ff603539231d4139def68bef2f65eaca";
    }

    headers.put("Authorization", "Bearer " + token.trim());
    headers.put("Content-Type", CONTENT_TYPE_JSON);
    headers.put("Accept", CONTENT_TYPE_JSON);
    return headers;
}
```

```
public static CreateUserRequest buildDefaultUserRequest() {
    CreateUserRequest request = new CreateUserRequest();
    request.setName(faker.name().fullName());
    request.setEmail(faker.internet().emailAddress());
    request.setGender(GENDER_FEMALE);
    request.setStatus(STATUS_ACTIVE);
    return request;
}
```



Automation Testing

Requests Class

```
public static GetPostResponseModel[] GetAllPosts(){ 1 usage
    return RestAssured.given().log().all()
        .headers(Builder.getHeaders())
        .contentType(ContentType.JSON)
        .get(s: Builder.base_url + Builder.POST_ENDPOINT).as(GetPostResponseModel[].class);
}
```

```
public static InvalidResponseMessage InvalidIDUpdatePost(CreatePostRequestModel updatePost, String id, Integer StatusCode) {
    return RestAssured.given().log().all() RequestSpecification
        .headers(Builder.getHeaders())
        .contentType(ContentType.JSON)
        .body(updatePost)
        .patch(s: Builder.base_url + Builder.POST_ENDPOINT+ id) Response
        .then().statusCode(StatusCode) ValidatableResponse
        .extract().as(InvalidResponseMessage.class);
}
```



Automation Testing

Requests Class

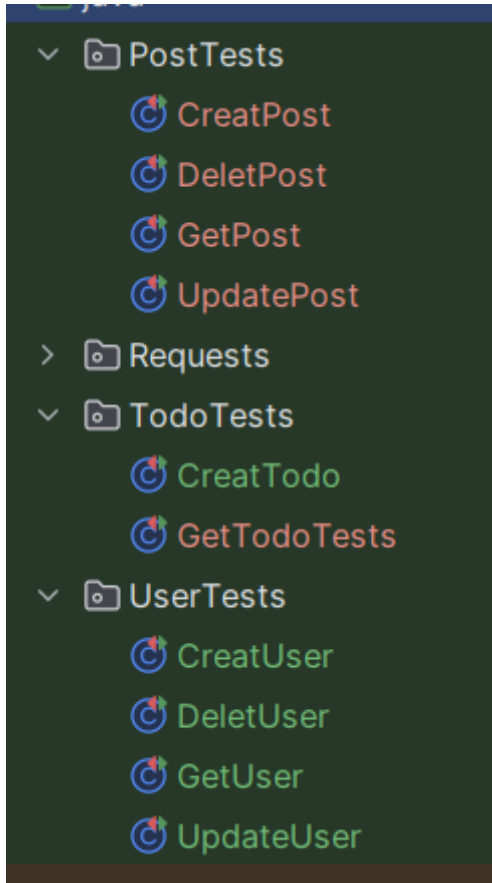
```
public static InvalidResponseMessageModel CreateUserWithoutAuth(CreateUserRequest createUserRequestModel, int statusCode) {  
    return given().log().all() RequestSpecification  
        .contentType(ContentType.JSON)  
        .body(createUserRequestModel)  
        .post(s: Builder.base_url + Builder.userEndPoint) Response  
        .then().statusCode(statusCode) ValidatableResponse  
        .extract().as(InvalidResponseMessageModel.class);  
}
```

```
public static GetTodoResponseModel[] GetAllTodos(Integer statusCode){ 1 usage  
    return RestAssured.given().log().all() RequestSpecification  
        .headers(Builder.getHeaders())  
        .contentType(ContentType.JSON)  
        .get(s: Builder.base_url + Builder.TODO_ENDPOINT) Response  
        .then().statusCode(statusCode).extract() ExtractableResponse<Response>  
        .as(GetTodoResponseModel[].class);  
}
```



Automation Testing

Test cases Class



```
@Test(priority = 1)
public static void ValidateKeysInPosts() {
    GetPostResponseModel[] getPostResponseModel = PostRequest.GetAllPosts();
    for (GetPostResponseModel post : getPostResponseModel) {
        Assert.assertNotNull(post.getId(), message: "Missing field: id");
        Assert.assertNotNull(post.getUserId(), message: "Missing field: user_id");
        Assert.assertNotNull(post.getTitle(), message: "Missing field: title");
        Assert.assertNotNull(post.getBody(), message: "Missing field: body");
    }
}
```



Automation Testing

Test cases Class

```
@Test(priority = 3)
public void update_post_with_invalidId() {
    PostRequest.deletePostWithValidId(Post_id);
    CreatePostRequestModel updatePost = new CreatePostRequestModel();
    updatePost.setTitle("New Title: Why Learning a New Skill is Easier Than You Think");
    updatePost.setBody("New Body : With so many online resources available, learning a " +
        "new skill is more accessible than ever. Pick something you're passionate about and dedicate just 30 minutes a day - you'll be surprised");
    InvalidResponseMessage getPostResponseModel = PostRequest.InvalidIDUpdatePost(updatePost, Post_id, StatusCode: 404);
    Assert.assertEquals(getPostResponseModel.getMessage(), expected: "Resource not found");
}
```

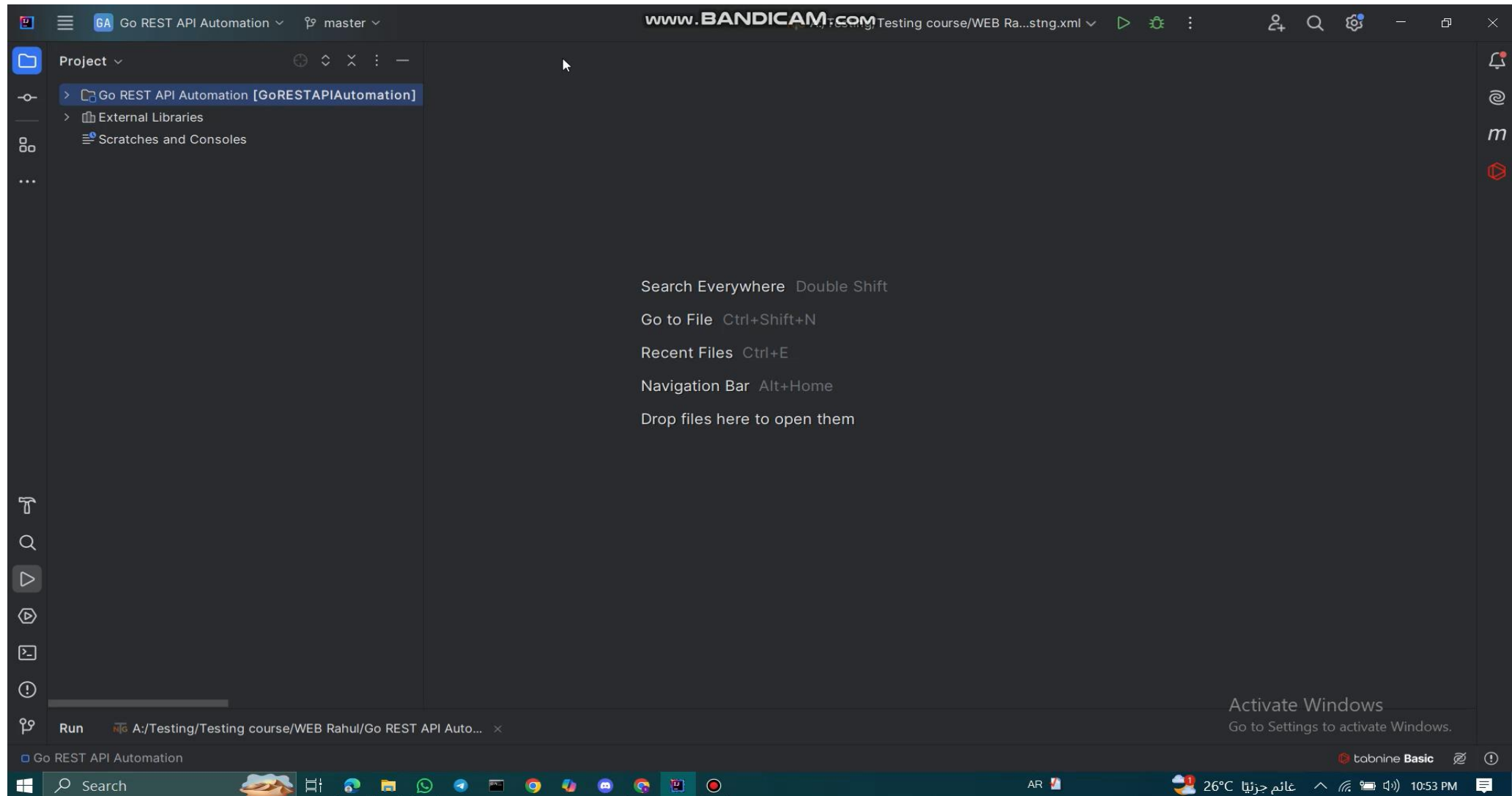
```
@Test(priority = 3) new *
public void updateWithInvalidEmailFormat() {
    CreateUserRequest invalidRequest = new CreateUserRequest()
        .setName(name)
        .setEmail(Builder.INVALID_EMAIL)
        .setGender(gender)
        .setStatus(status);

    CreateUserWithInvalidbodyResponse[] errors =
        UserReq.updateUserUsingInvalidBody(invalidRequest, id);

    Assert.assertEquals(errors[0].getField(), expected: "email");
    Assert.assertEquals(errors[0].getMessage(), expected: "is invalid");
}
```



Automation Testing



Thank You

