

Capstone Project

Project Name: API Testing for Contact List Application

Domain Name: Telecom

Submitted By: Ritik Prajapat [StarAgile Student Trainee]

Submitted To:  [StarAgile Consulting Pvt Ltd]

GitHubLink:

<https://github.com/ritikprajapat8085/AutomationProjects>

Table of Contents

1. Project Overview
2. Objectives
3. Tools
4. Testing Flow
5. Design Testcase/Scenarios
6. Testcase Execution(Using Postman & RestAssured)
7. Test Result Analysis & Reporting
8. Conclusion

1. Project Overview

This capstone project focuses on testing the **APIs** of a sample Contact List Application provided by the <https://thinking-tester-contact-list.herokuapp.com/> application in the Telecom Domain.

The application allows users to **register, log in, and manage contacts** (add, update, delete). It mimics a telecom service where users manage and store their contact information

2. Objective

The goal is to validate the REST API for the Contact List Application to ensure that it behaves as expected under all conditions.

- Validate HTTP methods: GET, POST, PUT, DELETE, PATCH
- Verify if API responses (data, format, status code) are correct and consistent.
- Test authentication & authorization (login, token usage, protected endpoints)
- Create and run manual/automated test cases using tools like Postman or Rest Assured.
- Maintain documentation of test scenarios, expected vs actual result

3. Tools:

Tools	Purpose
Postman	Manual API Testing
Newman	Command-line test runs
Java + Rest Assured + TestNG	Automated API Testing
Excel/Google Sheets	For test case documentation
Git-GitHub	for tracking code changes & manage

4. Testing Flow:

Add User → Get user profile → Update user → Login user → Add Contact → Get contact list → Get contact → Update full contact → Update partial contact → Logout User

5. Design Testcase/Scenarios

Test Case ID	Test Scenario	HTTP method	Endpoint	Expected Result
TC01	Add New User	POST	https://thinking-tester-contact-list.herokuapp.com/users	201 Created

Test Case ID	Test Scenario	HTTP method	Endpoint	Expected Result
TC02	Get User Profile	GET	https://thinking-tester-contact-list.herokuapp.com/users/me	200 OK
TC03	Update user	PATCH	https://thinking-tester-contact-list.herokuapp.com/users/me	200 OK
TC04	Login User	POST	https://thinking-tester-contact-list.herokuapp.com/users/login	200 OK
TC05	Add Contact	POST	https://thinking-tester-contact-list.herokuapp.com/contacts	201 Created
TC06	Get Contact List	GET	https://thinking-tester-contact-list.herokuapp.com/contacts	200 OK
TC07	Get Contact	GET	https://thinking-tester-contact-list.herokuapp.com/contacts/	200 OK
TC08	Update Contact	PUT	https://thinking-tester-contact-list.herokuapp.com/contacts/{{userid1}}	200 OK
TC09	Update Contact	PATCH	https://thinking-tester-contact-list.herokuapp.com/contacts/{{userid1}}	200 OK
TC10	Logout User	POST	https://thinking-tester-contact-list.herokuapp.com/users/logout	200 OK

6. Testcase Execution:

6.1 Manual Approach : Using Postman

Test Case 1: Create a new user

Request Type: NEW POST

Endpoint: <https://thinking-tester-contact-list.herokuapp.com/users>

Request Body:

```
{  
  "firstName": "Test",  
  "lastName": "User",  
  "email": "test@fake.com",  
  "password": "myPassword"  
}
```

Response validation:

- Test for status code and status message 201 created.
 - A token will be generated. Use the token for further testing

The screenshot shows the Postman application interface. On the left, the sidebar displays the 'MyFirstWorkSpace' collection with various items like 'Assignment-API', 'MyCollection1', etc. The 'TelecomProject' item is expanded, showing 'AddNewUser' under its POST requests. The main workspace shows a POST request to 'https://thinking-tester-contact-list.herokuapp.com/users'. The 'Body' tab is selected, showing raw JSON input:

```
1 {
2   "firstName": "Ritik",
3   "lastName": "Prajapati",
4   "email": "ritikprajapati@gmail.com",
5   "password": "Ritik@123"
6 }
```

The response status is '400 Bad Request' with a message: "Email address is already in use".

POST | <https://thinking-tester-contact-list.herokuapp.com/users>

Send

Overview Params Authorization Headers (10) Body Scripts Settings Cookies

Pre-request

```

1 // for dynamic Data
2 let time = new Date().getTime(); // this will give current time
3 let UpdatedEmailID = "ritikprajapti"+time+"@gmail.com"; // add current time in email id to make it unique
4
5
6 //set the email to Env variable
7 pm.environment.set(" UpdatedEmail", UpdatedEmailID);

```

Packages </> Snippets

HTTP TelecomProject / AddNewUser

POST | <https://thinking-tester-contact-list.herokuapp.com/users>

Save Share

Overview Params Authorization Headers (10) Body Scripts Settings Cookies

Post-response

```

1 // validation of status code
2 pm.test("Status code is 201", function () {
3   {
4     pm.response.to.have.status(201);
5   });
6
7 pm.test("Status code is Created", function () {
8   {
9     pm.response.to.have.status("Created");
10 });

```

Packages </> Snippets

Body Cookies (1) Headers (12) Test Results (2/2)

201 Created 2.72 s 1.25 KB Save Response

PASSED Status code is 201
PASSED Status code is Created

Home Workspaces API Network

MyFirstWorkSpace New Import

Collections Environments Flows APIs Monitors History

HTTP TelecomProject / AddNewUser

POST | <https://thinking-tester-contact-list.herokuapp.com/users>

Search Postman Ctrl K

Upgrade

Save Share

Overview Params Authorization Headers (10) Body Scripts Settings

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

```

1 {
2   "firstName": "Ritik",
3   "lastName": "Prajapti",
4   "email": "{{ UpdatedEmail }}",
5   "password": "Ritik@123"
6 }

```

Body Cookies (1) Headers (12) Test Results (2/2)

201 Created 2.72 s 1.25 KB Save Response

{ } JSON Preview Visualize

```

1 {
2   "user": {
3     "_id": "6862865580257200151e857e",
4     "firstName": "Ritik",
5     "lastName": "Prajapti",
6     "email": "ritikprajapti1751287379318@gmail.com",
7     "__v": 1
8   },
9   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJfaWQ1Oii20DYY0DY1NTA4MjU3MjAwMTUxZTA1N2UiLCJpYXQiOjE3NTEx00c20DF9.eHxtBw1kMfeYffuFAqvT8YHU7PUZTt0cnMs15lcjNo"
10 }

```

Online Find and replace Console

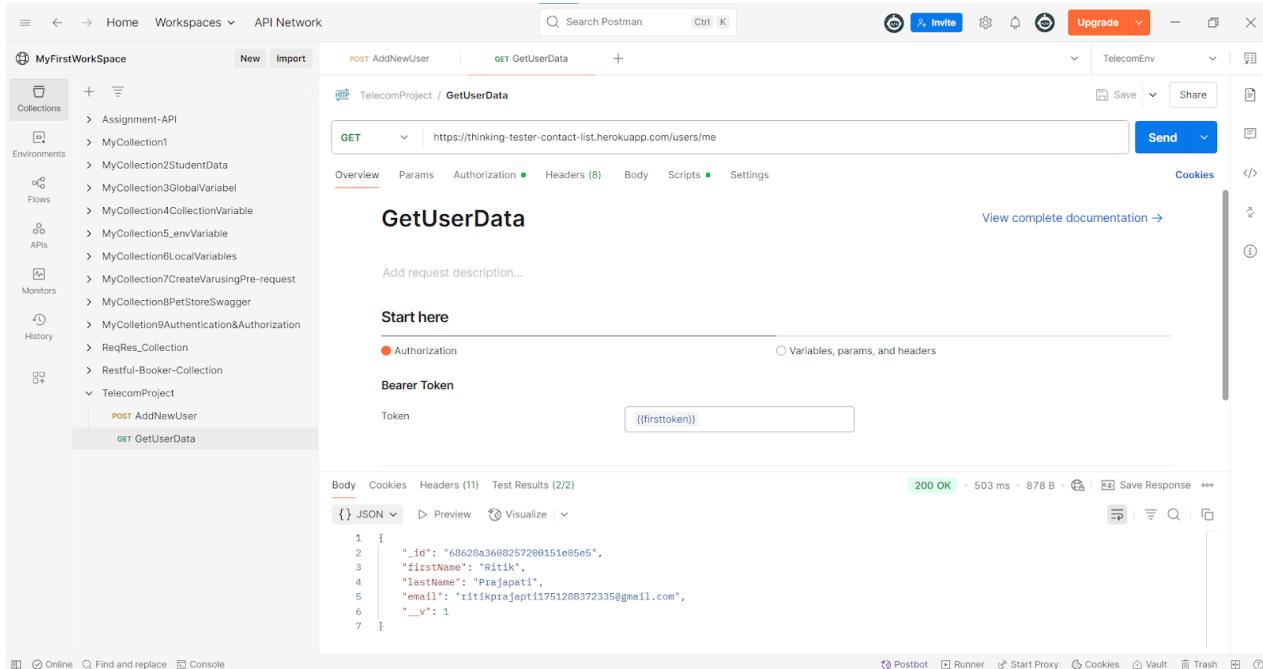
Test Case 2: Get user Profile

Request Type:  GET

Endpoint: <https://thinking-tester-contact-list.herokuapp.com/users/me>

Headers: Authorization Bearer {{token}}

Response validation: Test for status code and status message 200 OK

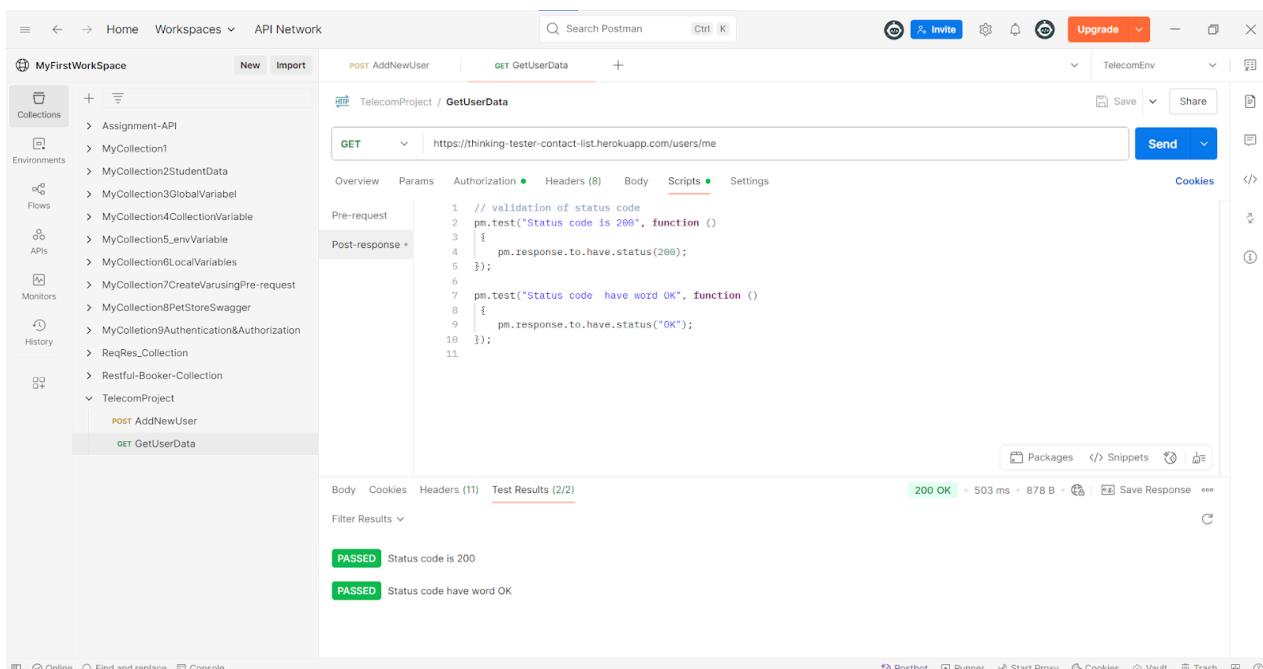


The screenshot shows the Postman application interface. On the left, the sidebar displays the workspace structure under 'MyFirstWorkSpace'. In the main area, a collection named 'TelecomProject' contains a 'GetUserData' endpoint. The 'Authorization' tab is selected, showing a 'Bearer Token' input field with the placeholder value `{{firsttoken}}`. The 'Body' tab shows a JSON response with the following content:

```

1 {
2   "_id": "60628a36600257200151e05e5",
3   "firstName": "Ritik",
4   "lastName": "Prajapati",
5   "email": "ritikprajapti1751288372335@gmail.com",
6   "__v": 1
7 }
  
```

The status bar at the bottom indicates a successful response: 200 OK, 503 ms, 878 B.



This screenshot shows the same Postman session after adding a 'Post-response' script. The 'Post-response' tab is selected, displaying the following JavaScript code:

```

1 // validation of status code
2 pm.test("Status code is 200", function () {
3   {
4     pm.response.to.have.status(200);
5   });
6
7 pm.test("Status code have word OK", function () {
8   {
9     pm.response.to.have.status("OK");
10 });
11
  
```

The 'Test Results' section at the bottom shows two green 'PASSED' results: 'Status code is 200' and 'Status code have word OK'.

Test Case 3: Update an existing user's details.

Request Type: 🖊 PATCH

Endpoint: <https://thinking-tester-contact-list.herokuapp.com/users/me>

Headers: Authorization Bearer {{token}}

Request Body:

```
{
  "firstName": "Updated",
  "lastName": "Username",
  "email": "test2@fake.com",
  "password": "myNewPassword"
}
```

Response validation: Test for status code and status message 200 OK .

The screenshot shows the Postman interface with two requests for updating a user:

- Request 1:** PATCH https://thinking-tester-contact-list.herokuapp.com/users/me
 - Authorization: Bearer Token ({{firsttoken}})
 - Body: JSON ({{firsttoken}})
- Request 2:** PATCH https://thinking-tester-contact-list.herokuapp.com/users/{{userid}}
 - Authorization: Bearer Token ({{token}})
 - Body: JSON ({{token}})

The bottom request's body is a JSON object with the following content:

```

1 {
2   "firstName": "Ritik",
3   "lastName": "Prajapati",
4   "email": "rituprajapati@gmail.com", // if there is issue for email then add "[{{ UpdatedEmail }}]"
5   "password": "Ritu@123"
6 }
7
8 // https://thinking-tester-contact-list.herokuapp.com/users/me: updating only email ,but not name or other data
9
10
11
12
13
    
```

The response for both requests is 200 OK with a response body showing the updated user details:

```

1 {
2   "_id": "68620a3660257200151e05e5",
3   "firstName": "Ritik",
4   "lastName": "Prajapati",
5   "email": "rituprajapati@gmail.com",
6   "__v": 1
7 }
    
```

POST TC01_AddNewUser | GET TC02_GetUserData | PATCH TC03_UpdateUser ● | GET Untitled Request | + v | TelecomEnv | 1

HTTP TelecomProject / **TC03_UpdateUser**

PATCH Save Share

Overview Params Authorization Headers (10) Body Scripts Settings Cookies Beautify

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

```

1 {
2   "firstName": "Ritu",
3   "lastName": "Prajapati",
4   // "email": "rituprajapati@gmail.com", // if there is issue for email then add "{{ UpdatedEmail }}"
5   "email": "ff UpdatedEmail",
6   "password": "Ritu@123"
7 }
8
9 // https://thinking-tester-contact-list.herokuapp.com/users/me: updating only email ,but not name or other data
10
11
12

```

Body Cookies Headers (11) Test Results (2/2) 200 OK 2.95 s 882 B Save Response

{ } JSON

```

1 {
2   "_id": "6662960be821bb00157110b5",
3   "firstName": "Rituprajapati",
4   "lastName": "Prajapati",
5   "email": "ritikprajapati1751291403921@gmail.com",
6   "__v": 1
7 }

```

HTTP TelecomProject / **TC03_UpdateUser**

PATCH Save Share

Overview Params Authorization Headers (10) Body Scripts Settings Cookies Beautify

Pre-request Post-response *

```

1 // validation of status code
2 pm.test("Status code is 200", function () {
3   {
4     pm.response.to.have.status(200);
5   });
6
7 pm.test("Status code have word OK", function () {
8   {
9     pm.response.to.have.status("OK");
10 });
11

```

Body Cookies Headers (11) Test Results (2/2) 200 OK 393 ms 865 B Save Response

Filter Results Status code is 200 Status code have word OK

Test Case 4: Login User

Request Type:  POST

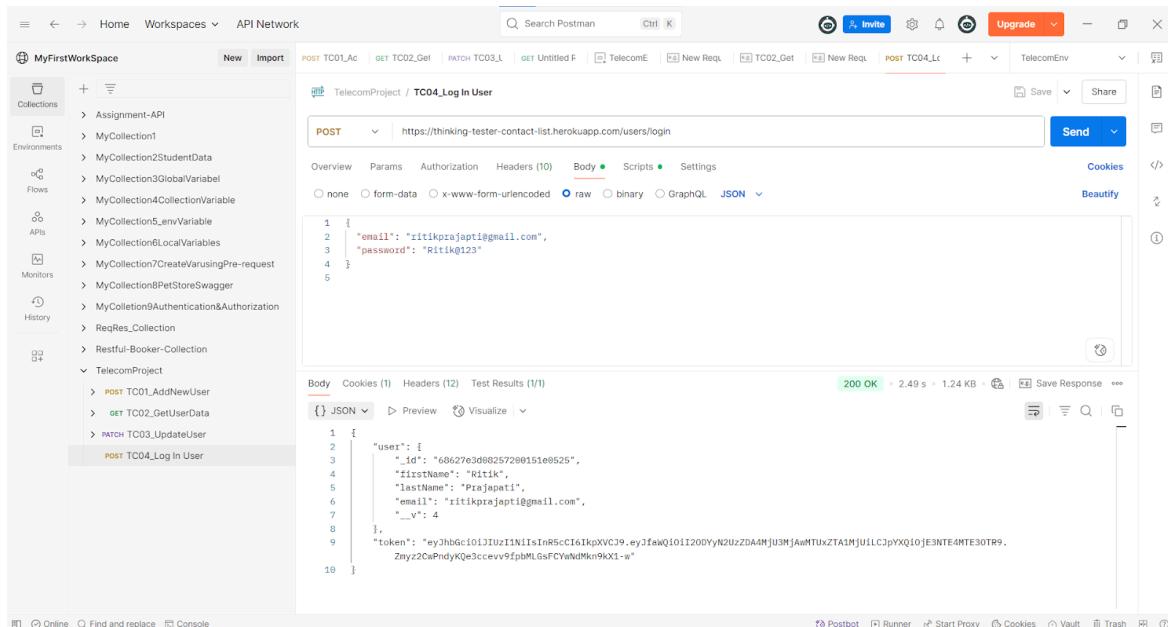
Endpoint: <https://thinking-tester-contact-list.herokuapp.com/users/login>

Request Body:

```
{
  "email": "test2@fake.com",
  "password": "myNewPassword"
}
```

Response validation:

- Test for status code and status message 200 OK
- A token will be generated. Use the token for further testing

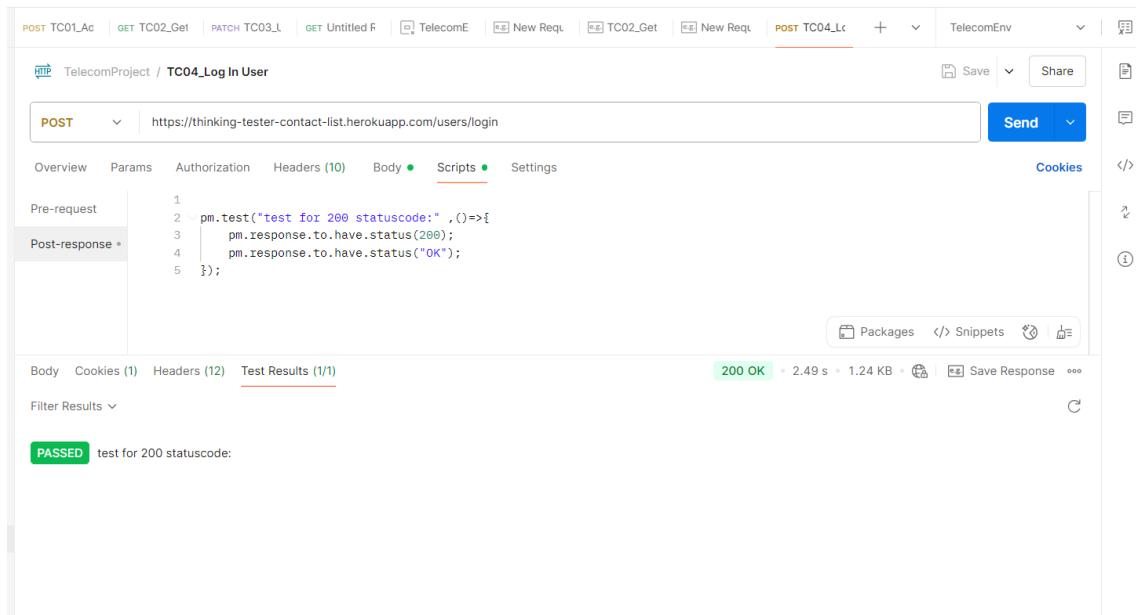


The screenshot shows the Postman interface with a project named 'MyFirstWorkSpace'. A collection named 'TelecomProject / TC04_Log In User' is selected. A POST request is made to <https://thinking-tester-contact-list.herokuapp.com/users/login>. The request body is a JSON object with 'email' and 'password' fields. The response is a 200 OK status with a response body containing user details and a token.

```

{
  "user": {
    "_id": "60627e3d0025200151e0525",
    "firstName": "Ritik",
    "lastName": "Prajapati",
    "email": "ritikprajapati@gmail.com",
    "__v": 4
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQ1Oii200YyN2UzD4MjU3MjAwMTUxZTA1MjU1LjpyXQiojE3NT4MTE3OTR9.Zmyz2CwPndyKQe3cccv9fpbMLGsFCYwNdMkn9Kx1-w"
}

```



The screenshot shows a test step for the POST request. The Pre-request script contains a pm.test block to check if the status code is 200. The Post-response script contains an assertion to check if the status is 'OK'.

```

pm.test("test for 200 statuscode:", ()=>{
  pm.response.to.have.status(200);
  pm.response.to.have.status("OK");
});

```

The test result is PASSED: test for 200 statuscode:

Test Case 5: Add Contact

Request Type: NEW POST

Endpoint: <https://thinking-tester-contact-list.herokuapp.com/contacts>

Headers: Authorization Bearer {{token}}

Request Body:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "birthdate": "1970-01-01",
  "email": "jdoe@fake.com",
  "phone": "8005555555",
  "street1": "1 Main St.",
  "street2": "Apartment A",
  "city": "Anytown",
  "stateProvince": "KS",
  "postalCode": "12345",
  "country": "USA"
}
```

Response validation: Test for status code and status message 201 Created

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'MyFirstWorkSpace' containing various collections, environments, flows, APIs, monitors, and history. The main area shows a collection named 'TelecomProject / TC05_add contact'. Underneath it, a specific POST request is defined with the URL <https://thinking-tester-contact-list.herokuapp.com/contacts>. The request body is set to JSON and contains the provided contact data. The 'Body' tab shows the raw JSON code. Below the request, the 'Test Results' section indicates a successful response with a status of 201 Created, a duration of 2.38 seconds, and a size of 1.04 KB. The response body is also displayed in JSON format, matching the input data.

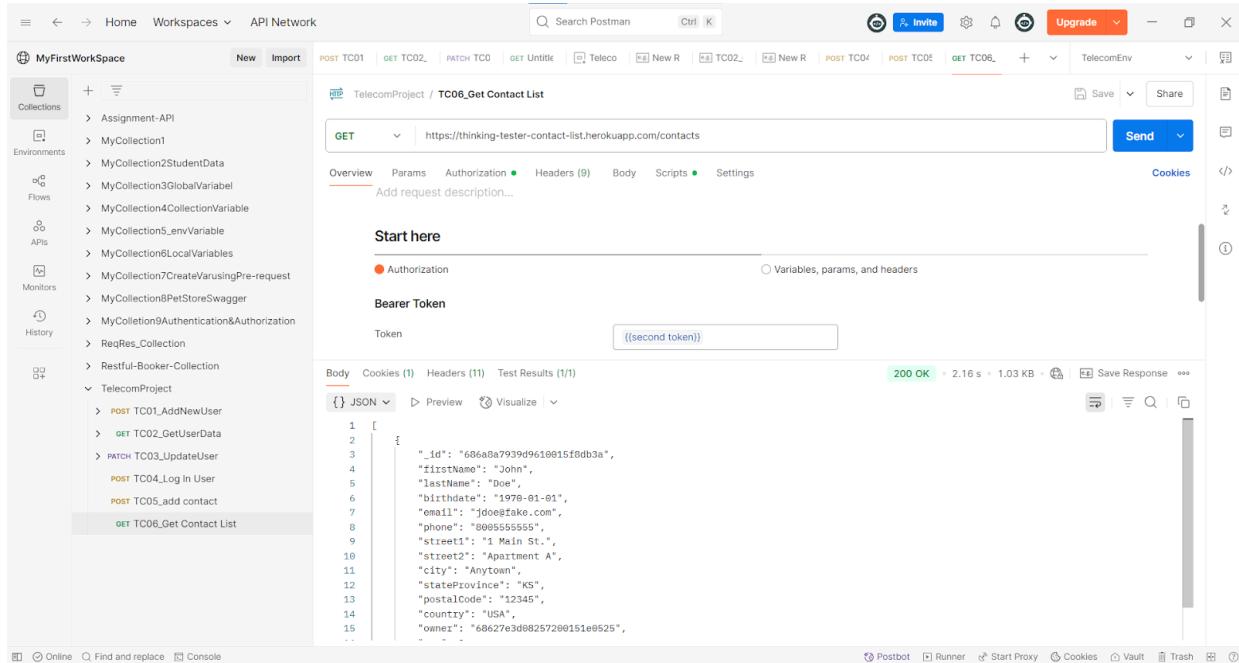
Test Case 6: Get Contact List

Request Type:  GET

Endpoint: <https://thinking-tester-contact-list.herokuapp.com/contacts>

Headers: Authorization Bearer {{token}}

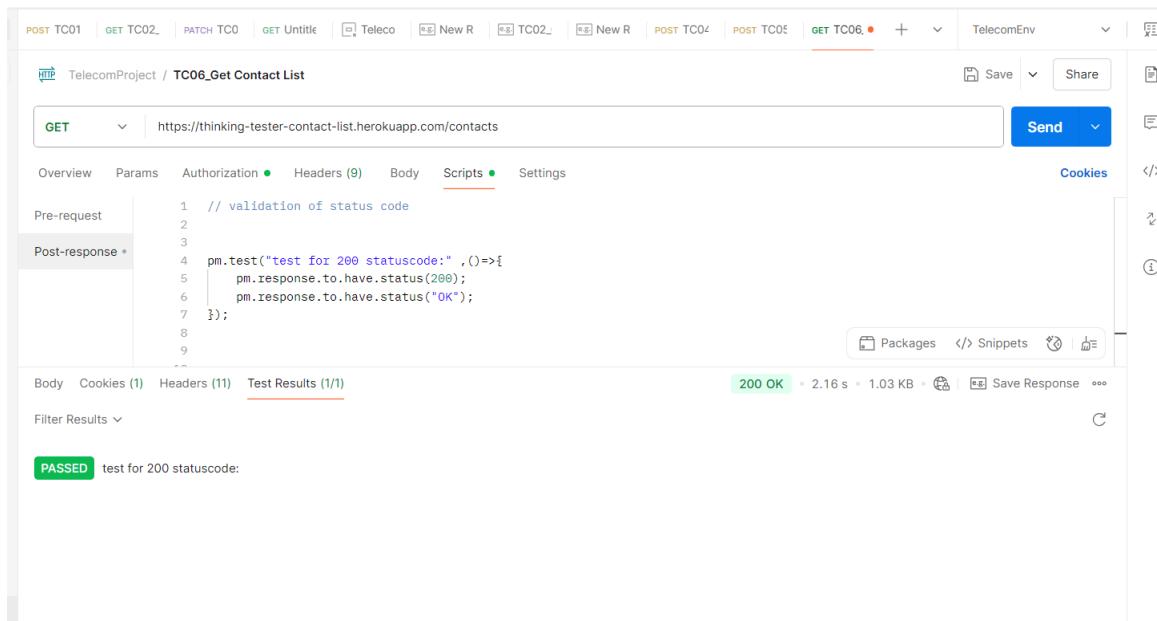
Response validation: Test for status code and status message 200 OK



The screenshot shows the Postman interface with a collection named "MyFirstWorkSpace". A new request is being created for "TC06_Get Contact List" with a GET method and the URL <https://thinking-tester-contact-list.herokuapp.com/contacts>. The "Authorization" tab is selected, showing a "Bearer Token" input field with the placeholder "Token" and a value of "{{second token}}". The response body is displayed as a JSON object:

```

1  [
2   {
3     "_id": "606a8a7939d9610015f0db3a",
4     "firstName": "John",
5     "lastName": "Doe",
6     "birthdate": "1970-01-01",
7     "email": "jdoe@fake.com",
8     "phone": "0000555555",
9     "street1": "1 Main St.",
10    "street2": "Apartment A",
11    "city": "Anytown",
12    "stateProvince": "KS",
13    "postalCode": "12345",
14    "country": "USA",
15    "owner": "68627e3d08257200151e0525",
16    ...
17  ]
  
```



The screenshot shows the Postman interface with a collection named "TelecomProject". A new request is being created for "TC06_Get Contact List" with a GET method and the URL <https://thinking-tester-contact-list.herokuapp.com/contacts>. The "Script" tab is selected, containing the following code:

```

1 // validation of status code
2
3
4 pm.test("test for 200 statuscode:", ()=>{
5   pm.response.to.have.status(200);
6   pm.response.to.have.status("OK");
7 });
8
9
  
```

The response body is displayed as a JSON object:

```

1  [
2   {
3     "_id": "606a8a7939d9610015f0db3a",
4     "firstName": "John",
5     "lastName": "Doe",
6     "birthdate": "1970-01-01",
7     "email": "jdoe@fake.com",
8     "phone": "0000555555",
9     "street1": "1 Main St.",
10    "street2": "Apartment A",
11    "city": "Anytown",
12    "stateProvince": "KS",
13    "postalCode": "12345",
14    "country": "USA",
15    "owner": "68627e3d08257200151e0525",
16    ...
17  ]
  
```

The status bar at the bottom indicates a successful response: 200 OK, 2.16 s, 1.03 KB.

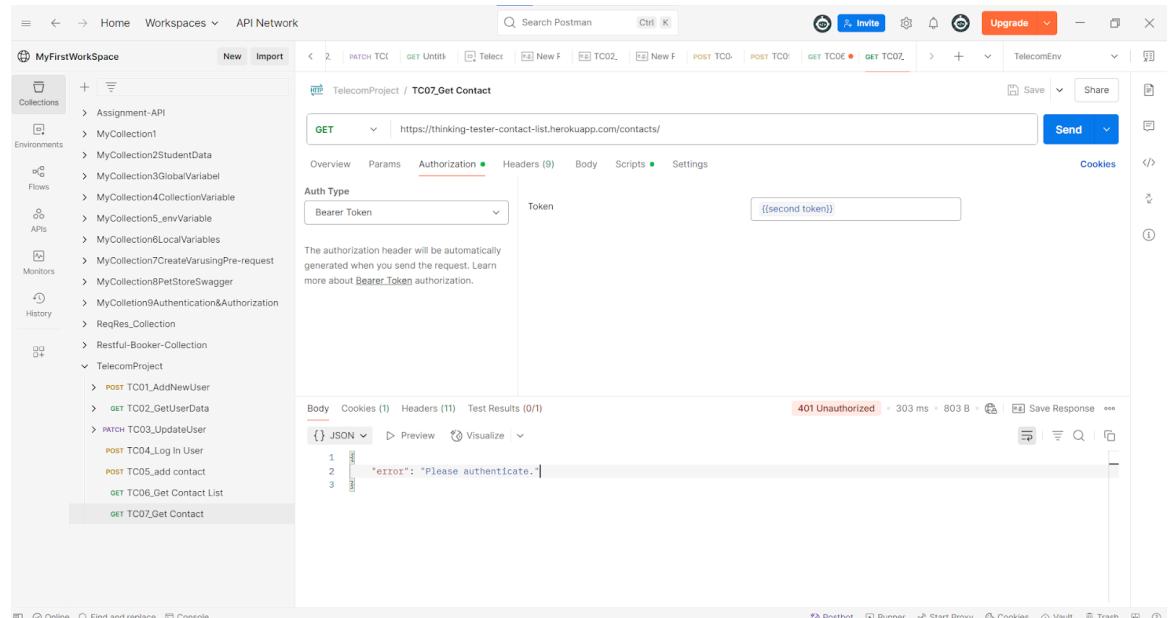
Test Case 7: Get Contact

Request Type:  GET

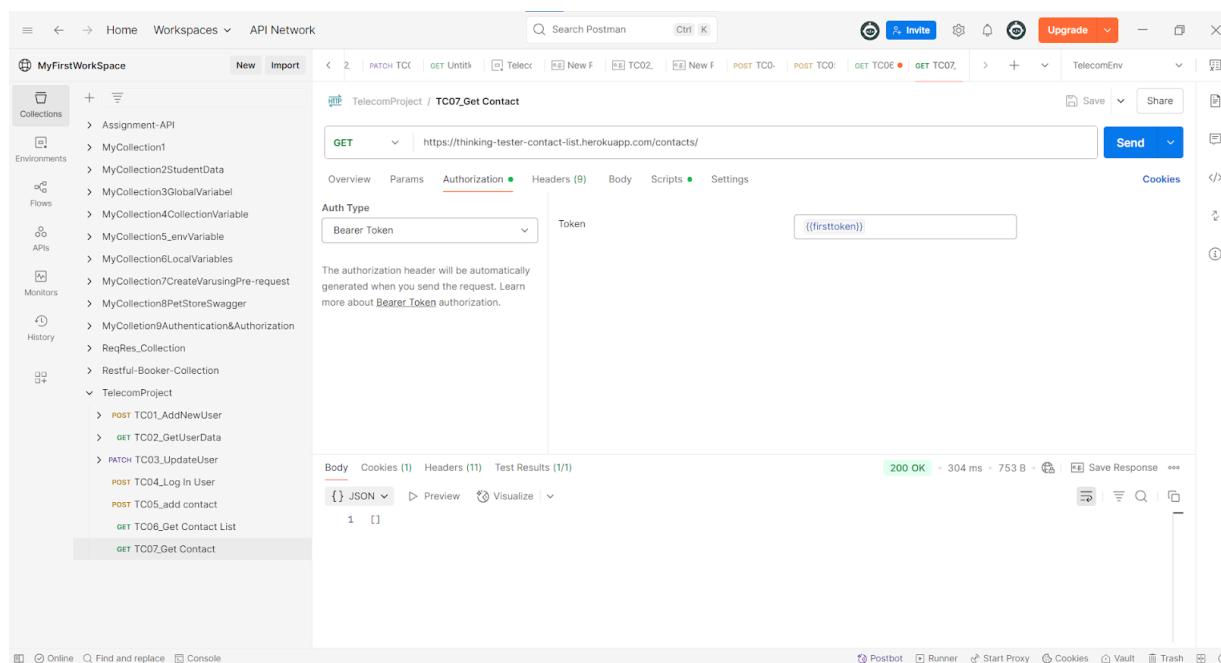
Endpoint: <https://thinking-tester-contact-list.herokuapp.com/contacts/>

Headers: Authorization Bearer {{token}}

Response validation: Test for status code and status message 200 OK



The screenshot shows the Postman interface with a failed API call. The URL is https://thinking-tester-contact-list.herokuapp.com/contacts/. The 'Authorization' tab is selected, showing 'Bearer Token' and token value {{second token}}. The response status is 401 Unauthorized with the error message "Please authenticate.".



The screenshot shows the Postman interface with a successful API call. The URL is https://thinking-tester-contact-list.herokuapp.com/contacts/. The 'Authorization' tab is selected, showing 'Bearer Token' and token value {{firsttoken}}. The response status is 200 OK.

Test Case 8: Update Contact

Request Type: 🖊 PUT

Endpoint: <https://thinking-tester-contact-list.herokuapp.com/contacts/>

Headers: Authorization Bearer {{token}}

Request Body:

```
{
  "firstName": "Amy",
  "lastName": "Miller",
  "birthdate": "1992-02-02",
  "email": "amiller@fake.com",
  "phone": "8005554242",
  "street1": "13 School St.",
  "street2": "Apt. 5",
  "city": "Washington",
  "stateProvince": "QC",
  "postalCode": "A1A1A1",
  "country": "Canada"
}
```

Response validation: Test for status code and status message 200 OK

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'MyFirstWorkSpace' containing various collections, environments, flows, and monitors. The main area shows a 'Telecom Project / TC08_Update Contact' collection. A specific 'PUT https://thinking-tester-contact-list.herokuapp.com/contacts/' request is selected. The 'Body' tab is active, displaying the JSON payload from the previous code block. The 'Headers' tab shows 'Content-Type: application/json'. Below the request, the 'Test Results' section indicates a '503 Service Unavailable' response with a duration of 32.59 s and a size of 1.19 KB. The response body is shown as an HTML error page with the title 'Application Error'.

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections like Collections, Environments, Flows, APIs, Monitors, and History. Under Collections, several items are listed, including 'Assignment-API', 'MyCollection1', 'MyCollection2StudentData', 'MyCollection3GlobalVariable', 'MyCollection4CollectionVariable', 'MyCollection5_envVariable', 'MyCollection6LocalVariables', 'MyCollection7CreateVarusingPre-request', 'MyCollection8PetStoreSwagger', 'MyCollection9Authentication&Authorization', 'ReqRes_Collection', 'Restful-Booker-Collection', and 'TelecomProject'. Under 'TelecomProject', there are several test cases: 'POST TC01_AddNewUser', 'GET TC02_GetUserData', 'PATCH TC03_UpdateUser', 'POST TC04_Log In User', 'POST TC05_add contact', 'GET TC06_Get Contact List', 'GET TC07_Get Contact', and 'PUT TC08_Update Contact'. The 'PUT TC08_Update Contact' item is currently selected.

The main workspace shows a 'TelecomProject / TC08_Update Contact' screen. The request type is 'PUT' and the URL is 'https://thinking-tester-contact-list.herokuapp.com/contacts/ {{userId}}'. The 'Body' tab is selected, showing a JSON payload:

```

1 {
2   "firstName": "Amy",
3   "lastName": "Miller",
4   "birthdate": "1992-02-02",
5   "email": "amiller@fake.com",
6   "phone": "8005554242",
7   "street1": "13 School St.",
8   "street2": "Apt. 5",
9   "city": "Washington",
10  "stateProvince": "QC",
11  "postalCode": "A1A1A1",
12  "country": "Canada"
13 }

```

The response section shows a green '200 OK' status with a response time of 370 ms and a size of 1.04 KB. The response body is identical to the request body.

Test Case 9: Update Contact

Request Type: PATCH

Endpoint: <https://thinking-tester-contact-list.herokuapp.com/contacts/>

Headers: Authorization Bearer {{token}}

Request Body:

```
{
  "firstName": "Anna"
}
```

Response validation: Test for status code and status message 200 OK .

This screenshot shows the same Postman interface as the previous one, but with a different test case selected. The 'PUT TC08_Update Contact' item from the previous screenshot has been replaced by a new 'PATCH TC09_Update Contact' item in the 'TelecomProject' collection.

The 'PATCH TC09_Update Contact' screen shows the request type is 'PATCH' and the URL is 'https://thinking-tester-contact-list.herokuapp.com/contacts/ {{userId}}'. The 'Body' tab is selected, showing a JSON payload:

```

1 {
2   "firstName": "Anna"
3 }

```

The response section shows a green '200 OK' status with a response time of 2.21 s and a size of 1.04 KB. The response body is identical to the request body.

Postman screenshot showing a successful PATCH request to update a contact. The request URL is <https://thinking-tester-contact-list.herokuapp.com/contacts/{userId1}>. The response status is 200 OK with a duration of 2.21 s and a size of 1.04 KB. A test script in the Scripts tab checks for a 200 status code.

```

1 // validation of status code
2
3
4 pm.test("test for 200 statuscode:", ()=>{
5     pm.response.to.have.status(200);
6     pm.response.to.have.status("OK");
7 });
8
9

```

Test Case 10: Logout User

Request Type: NEW POST

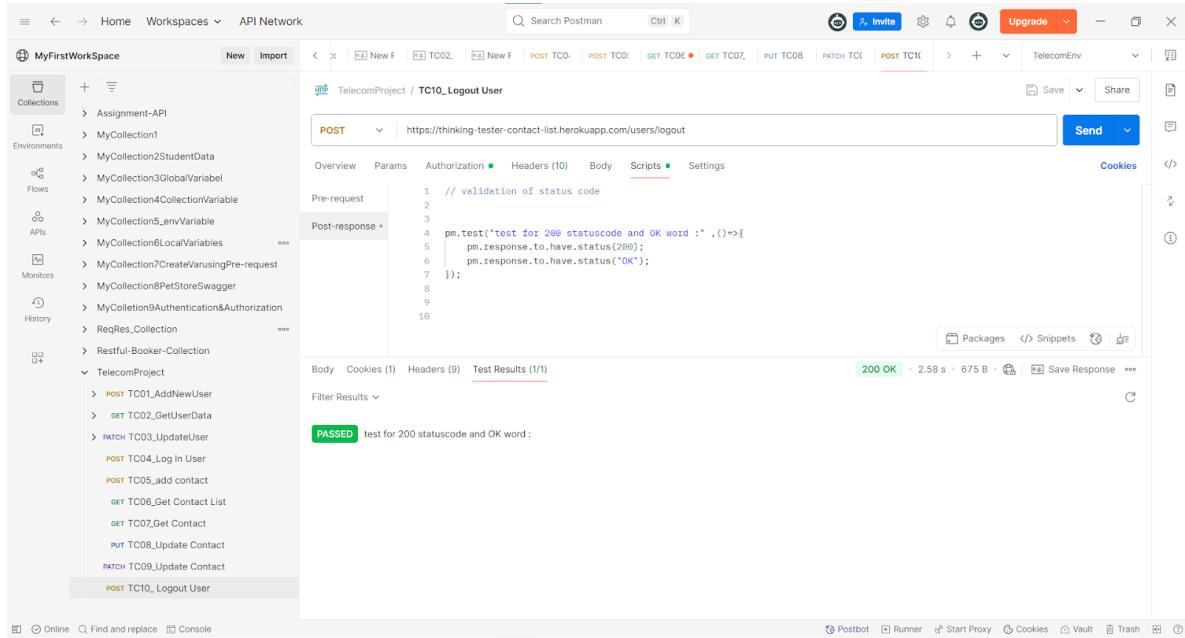
Endpoint: <https://thinking-tester-contact-list.herokuapp.com/users/logout>

Headers: Authorization Bearer {{token}}

Response validation: Test for status code and status message 201 created

Postman screenshot showing a successful POST request to logout a user. The request URL is <https://thinking-tester-contact-list.herokuapp.com/users/logout>. The response status is 200 OK with a duration of 2.58 s and a size of 675 B. A token {{firstToken}} is passed in the Authorization header.

Authorization
Bearer Token
Token: {{firstToken}}



6. Testcase Execution:

6.2 Automation Approach: Using RestAssured framework

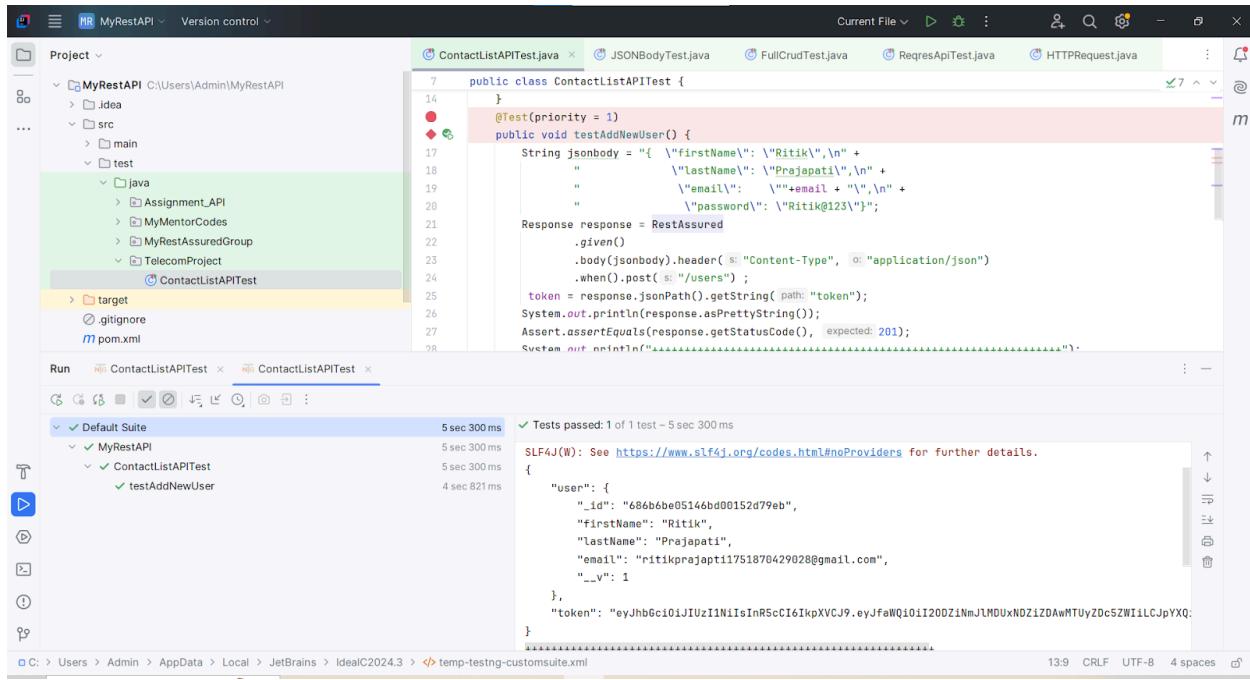
Test Case 1: Add New User

```

package TelecomProject;
import io.restassured.RestAssured;
import io.restassured.response.Response;
import org.testng.Assert;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class ContactListAPITest {
    String email = "ritikpj@gmail.com" + System.currentTimeMillis() + "@gmail.com";
    String token = "";
    // String contactId = "";
    @BeforeClass
    public void setup() {
        RestAssured.baseURI = "https://thinking-tester-contact-list.herokuapp.com";
    }
    @Test(priority = 1)
    public void testAddNewUser() {
        String jsonbody = "{ \"firstName\": \"Ritik\", \"lastName\": \"Projapati\", \"email\": \"" + email + "\", \"password\": \"Ritik@123\"}";
        Response response = RestAssured
                .given()
                .body(jsonbody).header("Content-Type", "application/json")
                .when().post("/users");
        token = response.jsonPath().getString("token");
        System.out.println(response.asPrettyString());
        Assert.assertEquals(response.getStatusCode(), 201);
        System.out.println("++++++");
    }
}
  
```

The screenshot shows the IntelliJ IDEA interface with the 'ContactListAPITest.java' file open. The code uses the RestAssured library to send a POST request to the '/users' endpoint with a JSON body containing 'firstName', 'lastName', 'email', and 'password' fields. The response is checked for a status code of 201.



```

public class ContactListAPITest {
    ...
    @Test(priority = 1)
    public void testAddNewUser() {
        String jsonbody = "{ \"firstName\": \"Ritik\", \"lastName\": \"Prajapati\", \"email\": \"ritikprajapati1751870429028@gmail.com\", \"password\": \"Ritik@123\"}";
        Response response = RestAssured
            .given()
            .body(jsonbody).header("Content-Type", "application/json")
            .when().post("/users");
        token = response.jsonPath().getString("token");
        System.out.println(response.asPrettyString());
        Assert.assertEquals(response.getStatusCode(), 201);
        System.out.println("+++++TC1 Passed : add new user+++++");
    }
}

```

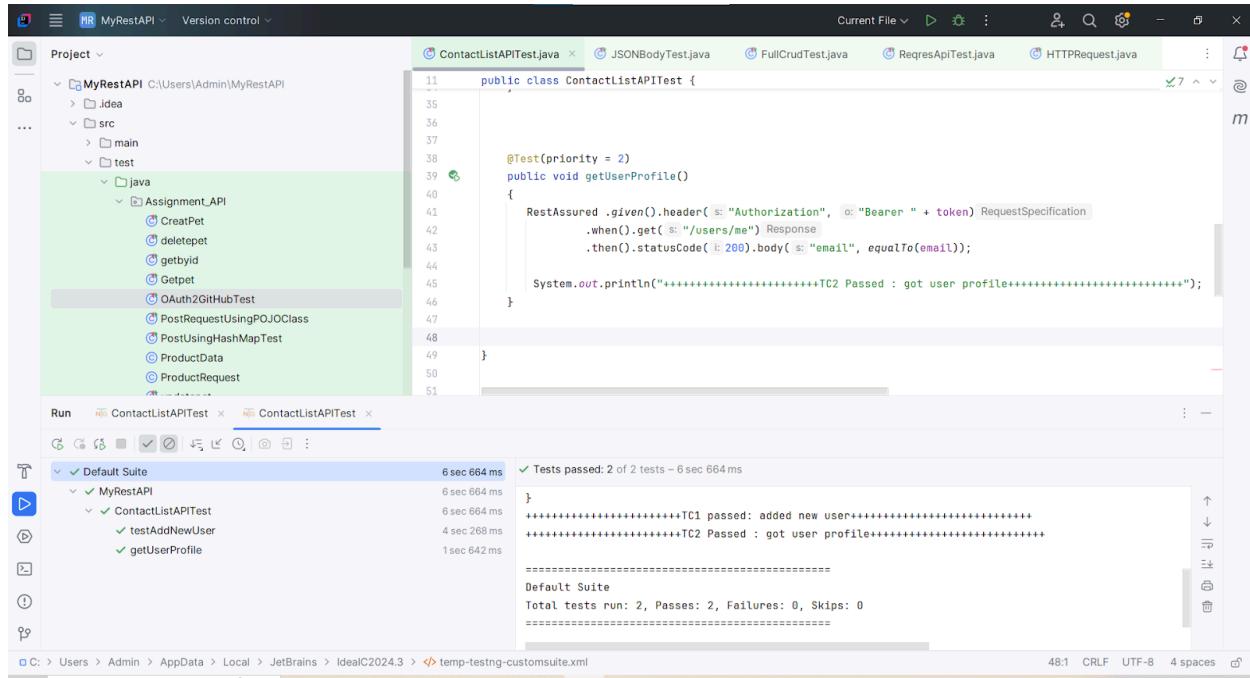
Tests passed: 1 of 1 test – 5 sec 300 ms

SLF4J(W): See <https://www.slf4j.org/codes.html#noProviders> for further details.

Default Suite
 MyRestAPI
 ContactListAPITest
 testAddNewUser

C: > Users > Admin > AppData > Local > JetBrains > IdeaIC2024.3 > temp-testing-customsuite.xml

Test Case 2: Get user Profile



```

public class ContactListAPITest {
    ...
    @Test(priority = 2)
    public void getUserProfile() {
        RestAssured.given().header("Authorization", "Bearer " + token)
            .when().get("/users/me")
            .then().statusCode(200).body("email", equalTo(email));
        System.out.println("+++++TC2 Passed : got user profile+++++");
    }
}

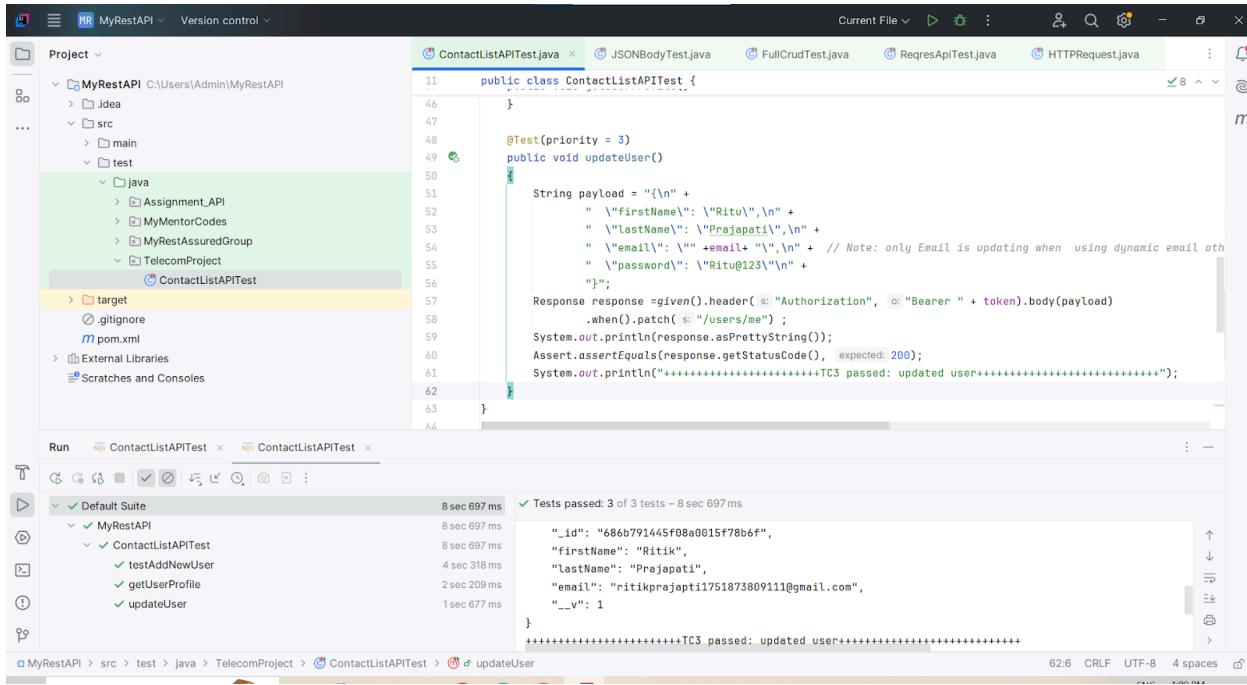
```

Tests passed: 2 of 2 tests – 6 sec 664 ms

Default Suite
 MyRestAPI
 ContactListAPITest
 testAddNewUser
 getUserProfile

C: > Users > Admin > AppData > Local > JetBrains > IdeaIC2024.3 > temp-testing-customsuite.xml

Test Case 3: Update User



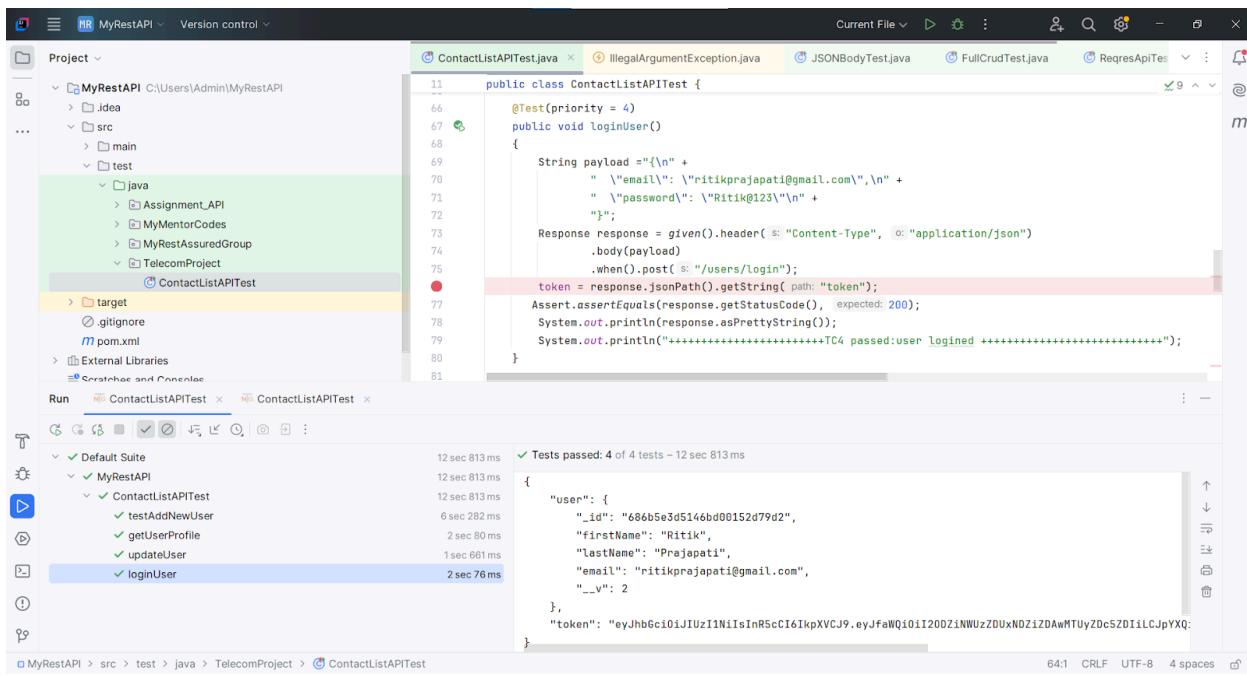
```

public class ContactListAPITest {
    @Test(priority = 3)
    public void updateUser() {
        String payload = "{\n" +
            "    \"firstName\": \"Ritu\",\n" +
            "    \"lastName\": \"Prajapati\",\n" +
            "    \"email\": \"\" +email+ \"\", \n" + // Note: only Email is updating when using dynamic email otherwise it will update all fields
            "    \"password\": \"Ritu@123\"\n" +
            "}";
        Response response = given().header( s: "Authorization", o: "Bearer " + token).body(payload)
            .when().patch( s: "/users/me");
        System.out.println(response.asPrettyString());
        Assert.assertEquals(response.getStatusCode(), expected: 200);
        System.out.println("*****TC3 passed: updated user*****");
    }
}

```

The screenshot shows the IntelliJ IDEA interface with the ContactListAPITest.java file open. The code defines a updateUser() method that constructs a JSON payload and performs a PATCH request to the '/users/me' endpoint. The response is printed to the console, and an assertion checks for a status code of 200. A message '*****TC3 passed: updated user*****' is also printed.

Test Case 4: Login User



```

public class ContactListAPITest {
    @Test(priority = 4)
    public void loginUser() {
        String payload = "{\n" +
            "    \"email\": \"ritikprajapati@gmail.com\",\n" +
            "    \"password\": \"Ritik@123\"\n" +
            "}";
        Response response = given().header( s: "Content-Type", o: "application/json")
            .body(payload)
            .when().post( s: "/users/login");
        token = response.jsonPath().getString(path: "token");
        Assert.assertEquals(response.getStatusCode(), expected: 200);
        System.out.println(response.asPrettyString());
        System.out.println("*****TC4 passed: user logged in *****");
    }
}

```

The screenshot shows the IntelliJ IDEA interface with the ContactListAPITest.java file open. The code defines a loginUser() method that constructs a JSON payload and performs a POST request to the '/users/login' endpoint. The response is parsed for a 'token' value, and an assertion checks for a status code of 200. A message '*****TC4 passed: user logged in *****' is also printed.

Test Case 5: Add Contact

```

public class ContactListAPITest {
    @Test(priority = 5)
    public void AddContact() {
        HashMap<String, String> map = new HashMap<>(); // using HashMap for complex json data
        map.put("firstName", "John");
        map.put("lastName", "Doe");
        map.put("birthdate", "1970-01-01");
        map.put("email", "jdoe@fake.com");
        map.put("phone", "8005555555");
        map.put("street1", "1 Main St.");
        map.put("street2", "Apartment A");
        map.put("city", "Anytown");
        map.put("stateProvince", "KS");
        map.put("postalCode", "12345");
        map.put("country", "USA");

        Response response = RestAssured
            .given().body(map).header("Authorization", "Bearer " + token).header("Content-Type", "application/json")
            .when().post("/contacts");

        contactId = response.jsonPath().getString("id");
        response.then().statusCode(201);

        System.out.println(response.asPrettyString());
        System.out.println("*****TCS Passed : User added*****");
    }
}

```

Tests passed: 5 of 5 tests – 16 sec 788 ms

```

{
    "_id": "686b9eb345f08a0015f78d20",
    "firstName": "John",
    "lastName": "Doe",
    "country": "USA",
    "birthdate": "1970-01-01",
    "phone": "8005555555",
    "city": "Anytown",
    "postalCode": "12345",
    "stateProvince": "KS",
    "street1": "1 Main St.",
    "street2": "Apartment A",
    "email": "jdoe@fake.com",
    "owner": "686b9eb345f08a0015f78d20",
    "__v": 0
}

```

Tests passed: 5 of 5 tests – 16 sec 788 ms

Test Case 6: Get Contact List

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "MyRestAPI" located at "C:\Users\Admin\MyRestAPI". It contains "src" and "test" directories, with "test" further divided into "java" and "TelecomProject".
- Code Editor:** The file "ContactListAPITest.java" is open, showing Java code for testing a contact list API. The code includes annotations like @Test(priority = 6) and RestAssured methods to make a GET request to "/contacts" and check the response status.
- Run Tab:** The "Run" tab is selected, showing the execution results for the "ContactListAPITest" suite. It lists several tests: "Default Suite", "MyRestAPI", "ContactListAPITest", and individual test methods like "testAddNewUser", "getUserId", etc. All tests are marked as passed with green checkmarks.
- Output Tab:** The right side of the interface shows the "Tests passed" output, which includes a JSON object representing a contact record and a message indicating the test was successful.
- Bottom Status Bar:** The status bar shows "133.1 CRLF UTF-8 4 spaces".

Test Case 7: Get Contact

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "MyRestAPI" located at "C:\Users\Admin\MyRestAPI". It contains "src" and "test" directories, with "test" further divided into "java" and "TelecomProject".
- Code Editor:** The file "ContactListAPITest.java" is open, showing Java code for testing a contact by ID. The code uses RestAssured to make a GET request to "/contacts/" + contactId and verify the response body.
- Run Tab:** The "Run" tab is selected, showing the execution results for the "ContactListAPITest" suite. It lists several tests: "Default Suite", "MyRestAPI", "ContactListAPITest", and individual test methods like "testAddNewUser", "getUserId", etc. All tests are marked as passed with green checkmarks.
- Output Tab:** The right side of the interface shows the "Tests passed" output, which includes a message indicating the test was successful.
- Bottom Status Bar:** The status bar shows "135.1 CRLF UTF-8 4 spaces".

Test Case 8: Update Contact (PUT)

```

1 package TelecomProject;
2 public class POJOCLASS { no usages
3     private String firstName, lastName, birthdate, email, phone, street1, street2, city, stateProvince, postalCode;
4     // Getters and Setters method
5     public String getFirstName() { return firstName; } no usages
6     public void setFirstName(String firstName) { this.firstName = firstName; } no usages
7     public String getLastname() { return lastName; } no usages
8     public void setLastName(String lastName) { this.lastName = lastName; } no usages
9     public String getBirthdate() { return birthdate; } no usages
10    public void setBirthdate(String birthdate) { this.birthdate = birthdate; } no usages
11    public String getEmail() { return email; } no usages
12    public void setEmail(String email) { this.email = email; } no usages
13    public String getPhone() { return phone; } no usages
14    public void setPhone(String phone) { this.phone = phone; } no usages
15    public String getStreet1() { return street1; } no usages
16    public void setStreet1(String street1) { this.street1 = street1; } no usages
17    public String getStreet2() { return street2; } no usages
18    public void setStreet2(String street2) { this.street2 = street2; } no usages
19    public String getCity() { return city; } no usages
20    public void setCity(String city) { this.city = city; } no usages
21    public String getStateProvince() { return stateProvince; } no usages
22    public void setStateProvince(String stateProvince) { this.stateProvince = stateProvince; } no usages
23    public String getPostalCode() { return postalCode; } no usages
24    public void setPostalCode(String postalCode) { this.postalCode = postalCode; } no usages
25    public String getCountry() { return country; } no usages
26    public void setCountry(String country) { this.country = country; } no usages
27 }
28
29
30
31

```

Run ContactListAPITest ContactListAPITest

```

13 public class ContactListAPITest {
133
134
135
136
137     @Test(priority = 8)
138     public void updateFullContact() {
139         POJOCLASS contact = new POJOCLASS();
140         contact.setFirstName("Amy");
141         contact.setLastName("Miller");
142         contact.setBirthdate("1992-02-02");
143         contact.setEmail("smiller@fake.com");
144         contact.setPhone("8005552424");
145         contact.setStreet1("13 School St.");
146         contact.setStreet2("Apt. 9");
147         contact.setCity("Washington");
148         contact.setStateProvince("QC");
149         contact.setPostalCode("A1A1A1");
150         contact.setCountry("Canada");
151
152         Response response = RestAssured.given()
153             .header("Authorization", "Bearer " + token)
154             .header("Content-Type", "application/json")
155             .body(contact)
156             .when()
157             .put("/contacts/" + contactId);
158
159         assertEquals(response.getStatusCode(), expected: 200);
160         System.out.println(response.asPrettyString());
161         System.out.println("++++++ TC8 passed: Contact updated using POJO successfully ++++++");
162     }
163

```

Run ContactListAPITest ContactListAPITest

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "MyRestAPI" located at C:\Users\Admin\MyRestAPI. The "test" directory contains several Java files: Assignment_API, MyMentorCodes, MyRestAssuredGroup, and ContactListAPITest.
- ContactListAPITest.java:** The code defines a test class with methods for adding, updating, and deleting contacts using POJOs.
- Test Results:** The "Default Suite" run shows 10 tests passed, including "updateFullContact". The output for "updateFullContact" shows a successfully updated contact object with fields like id, firstName, lastName, etc.
- Bottom Status Bar:** Shows the file path as "MyRestAPI > src > test > java > TelecomProject > ContactListAPITest" and encoding as "163:1 CRLF UTF-8 4 spaces".

Test Case 9: Update Contact(PATCH)

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "MyRestAPI" located at C:\Users\Admin\MyRestAPI. The "test" directory contains several Java files: Assignment_API, MyMentorCodes, MyRestAssuredGroup, and ContactListAPITest.
- ContactListAPITest.java:** The code defines a test class with a method for partial update.
- Test Results:** The "Default Suite" run shows 8 tests passed, including "partialUpdateContact". The output for "partialUpdateContact" shows a successfully updated contact object with the first name "Anna".
- Bottom Status Bar:** Shows the file path as "MyRestAPI > src > test > java > TelecomProject > ContactListAPITest" and encoding as "147:9 CRLF UTF-8 4 spaces".

ContactListAPITest.java

```

13 public class ContactListAPITest {
14
15     @Test(priority = 9)
16     public void partialUpdateContact() {
17
18         String payload = "{\n" +
19             "    \"firstName\": \"Anna\"\n" +
20             "}";
21
22     }
23
24 }

```

Tests passed: 8 of 8 tests – 21 sec 199 ms

Default Suite

- MyRestAPI
 - ContactListAPITest
 - testAddNewUser
 - getUserProfile
 - updateUser
 - loginUser
 - AddContact
 - getContactList
 - getContactById
 - partialUpdateContact

21 sec 199 ms
21 sec 199 ms
21 sec 199 ms
6 sec
2 sec 168 ms
2 sec 197 ms
1 sec 980 ms
2 sec 370 ms
1 sec 978 ms
1 sec 873 ms
1 sec 924 ms

147:1 CRLF UTF-8 4 spaces

Test Case 10: Logout User

ContactListAPITest.java

```

13 public class ContactListAPITest {
14
15     @Test(priority = 10)
16     public void logoutUser() {
17
18         given().header( s: "Authorization", o: "Bearer " + token) RequestSpecification
19             .when().post( s: "/users/logout") Response
20             .then().statusCode( i: 200);
21
22         System.out.println("*****TC10 Passed : user logout *****");
23
24     }
25
26 }

```

Tests passed: 9 of 9 tests – 23 sec 147 ms

Default Suite

- MyRestAPI
 - ContactListAPITest
 - testAddNewUser
 - getUserProfile
 - updateUser
 - loginUser
 - AddContact
 - getContactList
 - getContactById
 - partialUpdateContact
 - logoutUser

23 sec 147 ms
23 sec 147 ms
23 sec 147 ms
5 sec 859 ms
2 sec 297 ms
2 sec 58 ms
1 sec 881 ms
2 sec 575 ms
1 sec 766 ms
2 sec
2 sec 31 ms
1 sec 892 ms

164:1 CRLF UTF-8 4 spaces

7. Test Result Analysis & Reporting

All designed test cases were executed using **Postman & rest-assured** Framework
The API worked as expected for valid scenarios. Proper error codes were returned for invalid and unauthorised access.

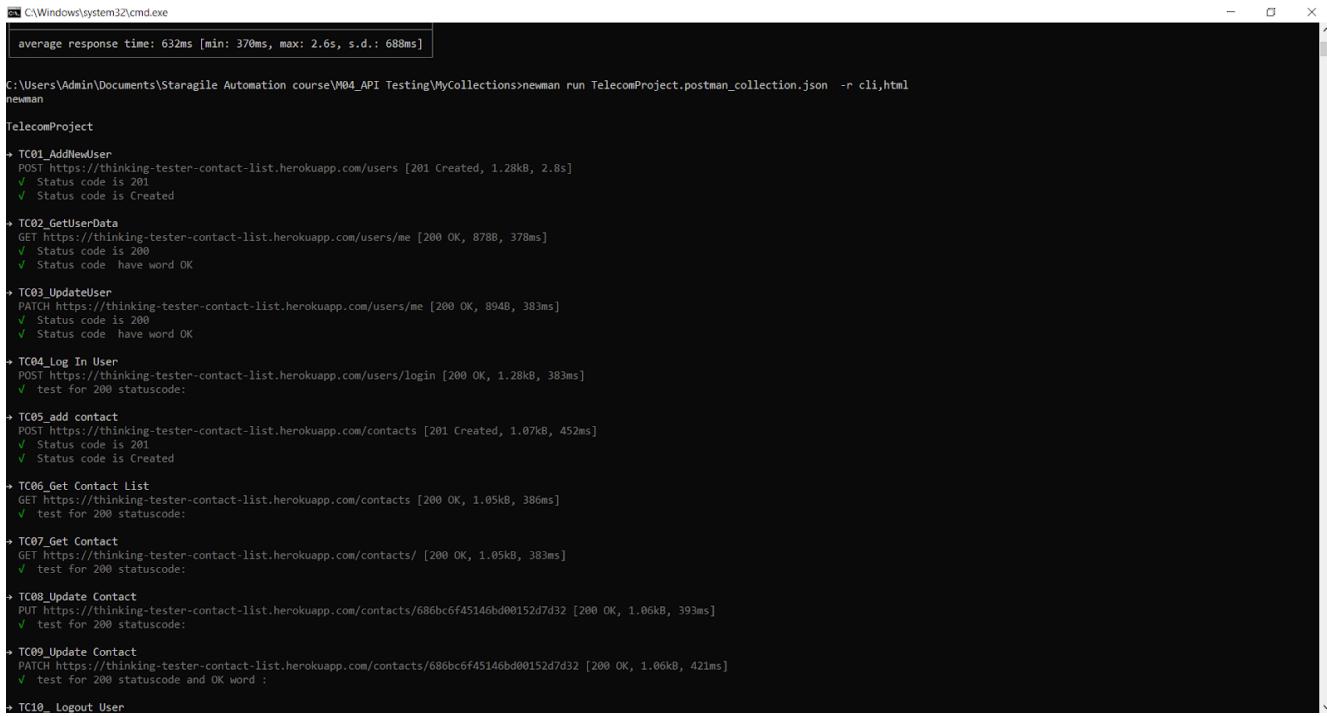
Total Test Cases: 10

- Passed: 10
- Failed: 0

Postman: Report generated from Newman(command line tool).

Link:

file:///C:/Users/Admin/Documents/Staragile%20Automation%20course/M04_API%20Testing/MyCollections/newman/API-TelecomProject.html



```
C:\Windows\system32\cmd.exe
average response time: 632ms [min: 370ms, max: 2.6s, s.d.: 689ms]

C:\Users\Admin\Documents\Staragile Automation course\M04_API Testing\MyCollections>newman run TelecomProject.postman_collection.json -r cli,html

TelecomProject

-> TC01_AddNewUser
  POST https://thinking-tester-contact-list.herokuapp.com/users [201 Created, 1.28kB, 2.8s]
    ✓ Status code is 201
    ✓ Status code is Created

-> TC02_GetUserData
  GET https://thinking-tester-contact-list.herokuapp.com/users/me [200 OK, 878B, 378ms]
    ✓ Status code is 200
    ✓ Status code have word OK

-> TC03_UpdateUser
  PATCH https://thinking-tester-contact-list.herokuapp.com/users/me [200 OK, 894B, 383ms]
    ✓ Status code is 200
    ✓ Status code have word OK

-> TC04_Log In User
  POST https://thinking-tester-contact-list.herokuapp.com/users/login [200 OK, 1.28kB, 383ms]
    ✓ test for 200 statuscode:

-> TC05_add contact
  POST https://thinking-tester-contact-list.herokuapp.com/contacts [201 Created, 1.07kB, 452ms]
    ✓ Status code is 201
    ✓ Status code is Created

-> TC06_Get Contact List
  GET https://thinking-tester-contact-list.herokuapp.com/contacts [200 OK, 1.05kB, 386ms]
    ✓ test for 200 statuscode:

-> TC07_Get Contact
  GET https://thinking-tester-contact-list.herokuapp.com/contacts/ [200 OK, 1.05kB, 383ms]
    ✓ test for 200 statuscode:

-> TC08_Update Contact
  PUT https://thinking-tester-contact-list.herokuapp.com/contacts/686bc6f45146bd00152d7d32 [200 OK, 1.06kB, 393ms]
    ✓ test for 200 statuscode and OK word:

-> TC09_Update Contact
  PATCH https://thinking-tester-contact-list.herokuapp.com/contacts/686bc6f45146bd00152d7d32 [200 OK, 1.06kB, 421ms]
    ✓ test for 200 statuscode and OK word:

-> TC10_Logout User
```

```
ok C:\Windows\system32\cmd.exe
> TC07_Get Contact
GET https://thinking-tester-contact-list.herokuapp.com/contacts/ [200 OK, 1.05kB, 383ms]
✓ test for 200 statuscode:

> TC08_Update Contact
PUT https://thinking-tester-contact-list.herokuapp.com/contacts/686bc6f45146bd00152d7d32 [200 OK, 1.06kB, 393ms]
✓ test for 200 statuscode:

> TC09_Update Contact
PATCH https://thinking-tester-contact-list.herokuapp.com/contacts/686bc6f45146bd00152d7d32 [200 OK, 1.06kB, 421ms]
✓ test for 200 statuscode and OK word :

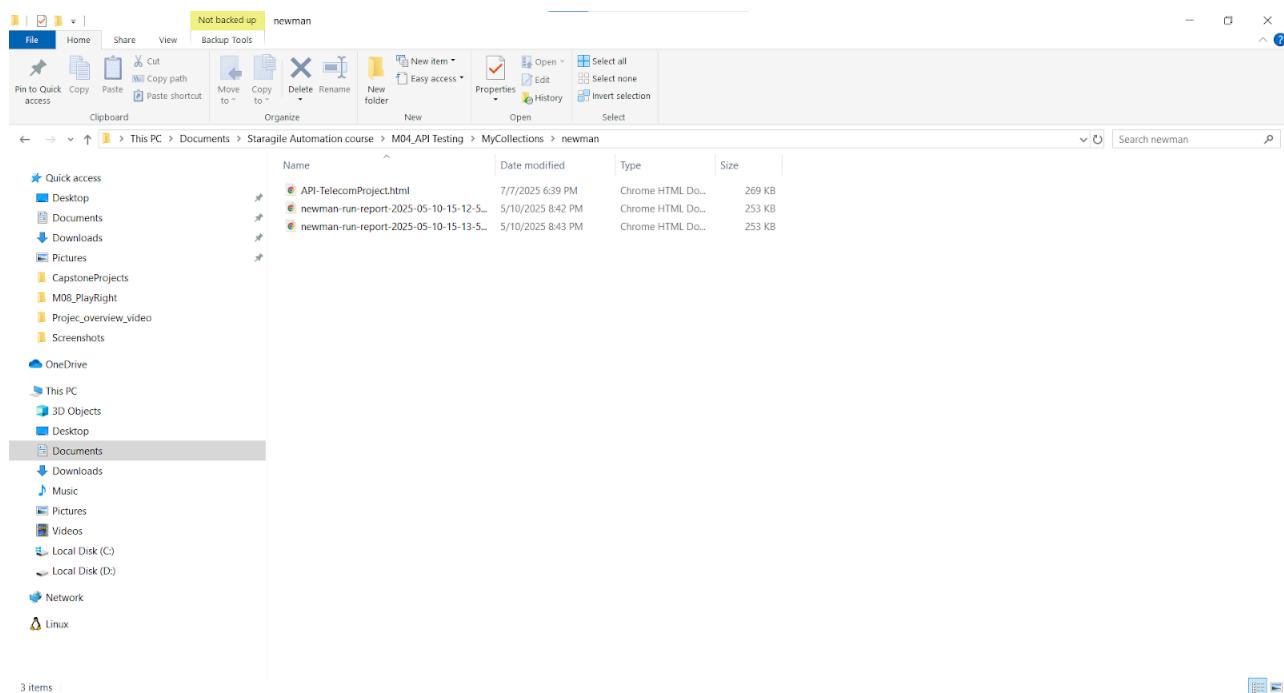
> TC10_Logout User
POST https://thinking-tester-contact-list.herokuapp.com/users/logout [200 OK, 655B, 403ms]
✓ test for 200 statuscode and OK word :



|                    | executed | failed |
|--------------------|----------|--------|
| iterations         | 1        | 0      |
| requests           | 10       | 0      |
| test-scripts       | 10       | 0      |
| prerequest-scripts | 1        | 0      |
| assertions         | 14       | 0      |


total run duration: 7.2s
total data received: 2.4kB (approx)
average response time: 641ms [min: 378ms, max: 2.8s, s.d.: 729ms]

C:\Users\Admin\Documents\Staragile Automation course\W04_API Testing\MyCollections>
```



Newman Report

Collection: TelecomProject
Time: Mon Jul 07 2025 18:39:11 GMT+0530 (India Standard Time)
Exported with: Newman v5.2.1

	Total	Failed
Iterations	1	0
Requests	10	0
Prerequest Scripts	1	0
Test Scripts	10	0
Assertions	14	0

Total run duration: 7.2s
Total data received: 2.36KB (approx)
Average response time: 64ms

Total Failures: 0

Requests:

TC01_AddNewUser

Method	URL
POST	https://thinking-tester-contact-list.herokuapp.com/users

Mean time per request: 2.8s
Mean size per request: 301B

Total passed tests: 2
Total failed tests: 0

Status code: 201

Tests	Name	Pass count	Fail count
	Status code is 201	1	0
	Status code is Created	1	0

TC02_GetUserData

TC02_GetUserData

Status code	Count
Created	1
Ok	0

Method: GET
URL: https://thinking-tester-contact-list.herokuapp.com/users/me

Mean time per request: 239ms
Mean size per request: 132B

Total passed tests: 2
Total failed tests: 0

Status code: 200

Tests	Name	Pass count	Fail count
	Status code is 200	1	0
	Status code have word Ok	1	0

TC03_UpdateUser

Status code	Count
Ok	2
Created	0

Method: PATCH
URL: https://thinking-tester-contact-list.herokuapp.com/users/me

Mean time per request: 363ms
Mean size per request: 152B

Total passed tests: 2
Total failed tests: 0

Status code: 200

Tests	Name	Pass count	Fail count
	Status code is 200	1	0
	Status code have word Ok	1	0

TC04_Log In User

Status code	Count
Ok	1
Created	0

Method: POST
URL: https://thinking-tester-contact-list.herokuapp.com/users/login

Mean time per request: 353ms
Mean size per request: 200B

Total passed tests: 1
Total failed tests: 0

Status code: 200

Tests	Name	Pass count	Fail count
	test for 200 statuscode	1	0

Postman Report

C:/Users/Admin/Documents/Staragile%20Automation%20course/M04_API%20Testing/MyCollections/newman/API-TelecomProject.html

TC06_add contact

Method	POST	
URL	https://thinking-tester-contact-list.herokuapp.com/contacts	
Mean time per request	452ms	
Mean size per request	305B	
Total passed tests	2	
Total failed tests	0	
Status code	201	
Tests		
Name	Pass count	Fail count
Status code is 201	1	0
Status code is Created	1	0

TC06_Get Contact List

Method	GET	
URL	https://thinking-tester-contact-list.herokuapp.com/contacts	
Mean time per request	300ms	
Mean size per request	307B	
Total passed tests	1	
Total failed tests	0	
Status code	200	
Tests		
Name	Pass count	Fail count
test for 200 statuscode	1	0

TC07_Get Contact

Method	GET	
URL	https://thinking-tester-contact-list.herokuapp.com/contacts/	
Mean time per request	353ms	
Mean size per request	307B	
Total passed tests	1	
Total failed tests	0	
Status code	200	
Tests		
Name	Pass count	Fail count
test for 200 statuscode	1	0

Postman Report

C:/Users/Admin/Documents/Staragile%20Automation%20course/M04_API%20Testing/MyCollections/newman/API-TelecomProject.html

TC08_Update Contact

Status code	200	
Tests		
Name	Pass count	Fail count
test for 200 statuscode	1	0

TC09_Update Contact

Method	PUT	
URL	https://thinking-tester-contact-list.herokuapp.com/contacts/989ed94014be000f52d7452	
Mean time per request	250ms	
Mean size per request	215B	
Total passed tests	1	
Total failed tests	0	
Status code	200	
Tests		
Name	Pass count	Fail count
test for 200 statuscode	1	0

TC10_Logout User

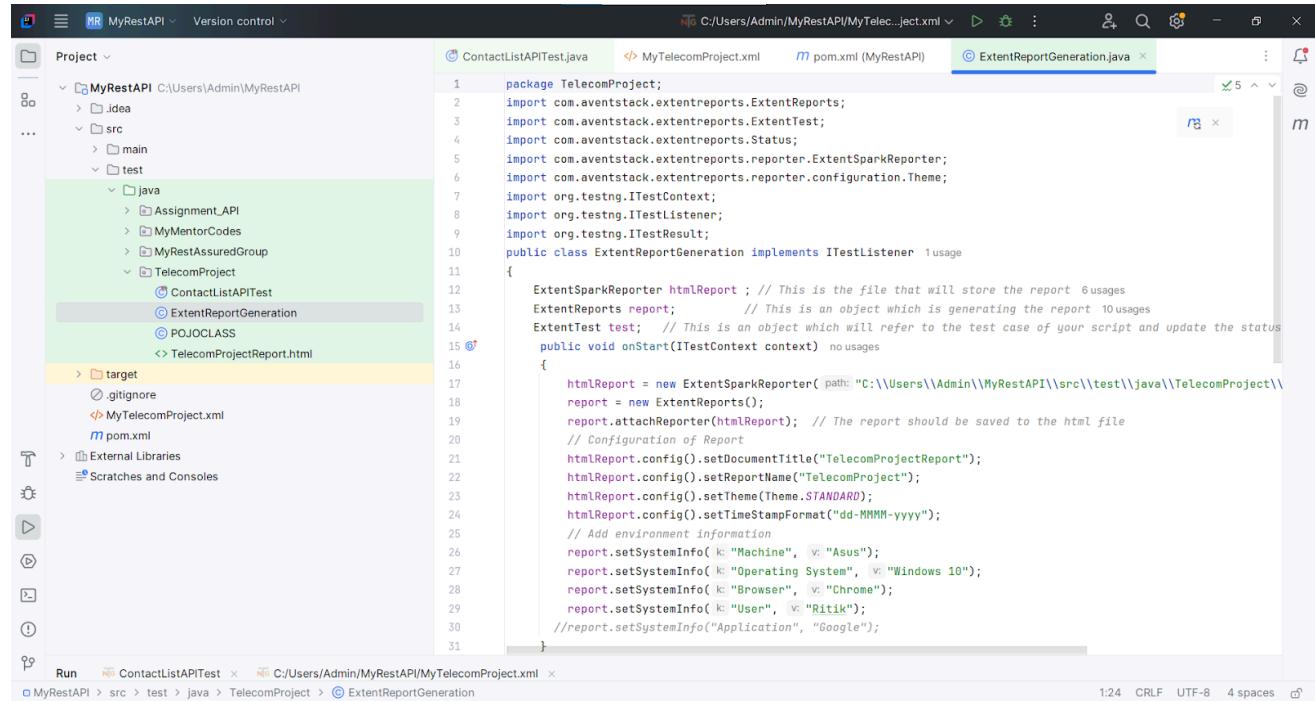
Method	PATCH	
URL	https://thinking-tester-contact-list.herokuapp.com/users/logout	
Mean time per request	421ms	
Mean size per request	215B	
Total passed tests	1	
Total failed tests	0	
Status code	200	
Tests		
Name	Pass count	Fail count
test for 200 statuscode and OK word	1	0

TC10_Logout User

Method	POST	
URL	https://thinking-tester-contact-list.herokuapp.com/users/logout	
Mean time per request	403ms	
Mean size per request	68B	
Total passed tests	1	
Total failed tests	0	
Status code	200	
Tests		
Name	Pass count	Fail count
test for 200 statuscode and OK word	1	0

Rest Assured: Report via TestNG/Extent

Step:1 Create a Utility Class with Extent Report Setup

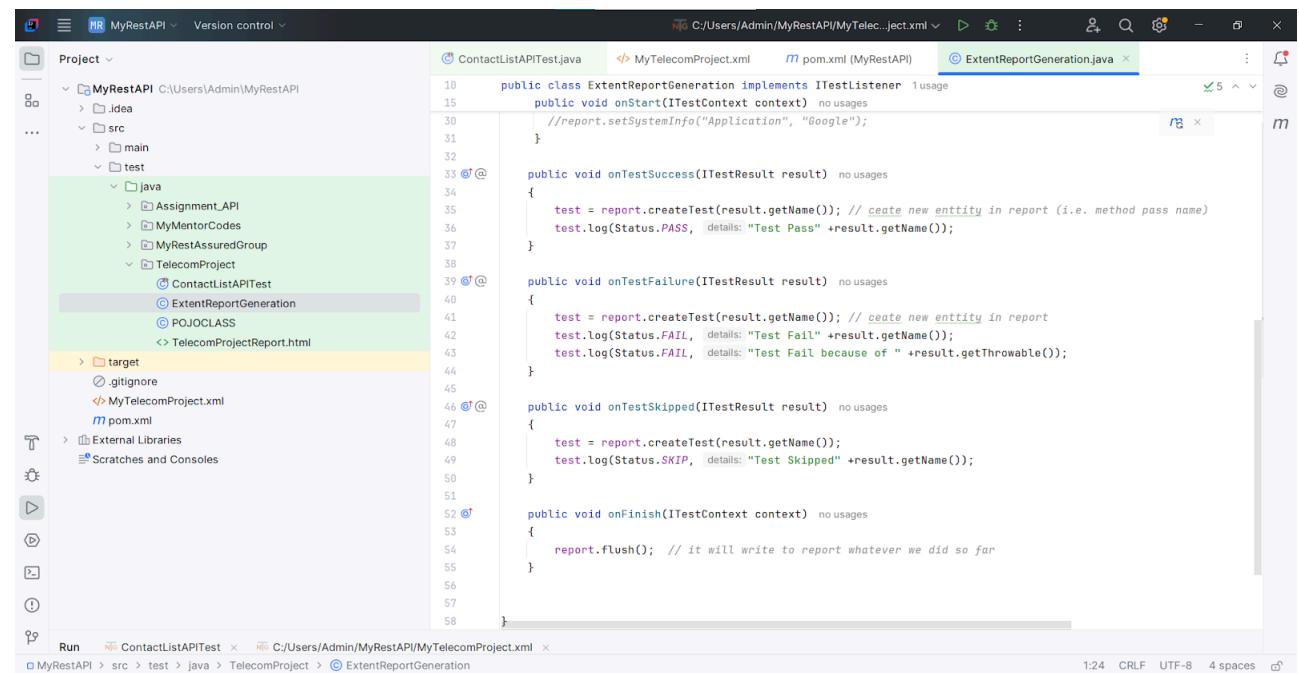


The screenshot shows the IntelliJ IDEA interface with the project 'MyRestAPI' open. The left sidebar displays the project structure, including the 'src' directory which contains 'main' and 'test' packages. The 'test' package is selected. Inside 'test', there is a 'java' folder containing several files: 'Assignment_API', 'MyMentorCodes', 'MyRestAssuredGroup', 'TelecomProject', 'ContactListAPITest', 'ExtentReportGeneration', 'POJOCLASS', and 'TelecomProjectReport.html'. The 'ExtentReportGeneration.java' file is currently selected and is displayed in the main editor window. The code implements the `ExtentReportGeneration` class to generate reports using ExtentReports.

```

package TelecomProject;
import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.Status;
import com.aventstack.extentreports.reporter.ExtentSparkReporter;
import com.aventstack.extentreports.reporter.configuration.Theme;
import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;
public class ExtentReportGeneration implements ITestListener {
{
    ExtentSparkReporter htmlReport ; // This is the file that will store the report 6 usages
    ExtentReports report; // This is an object which is generating the report 10 usages
    ExtentTest test; // This is an object which will refer to the test case of your script and update the status
    public void onStart(ITestContext context) no usages
    {
        htmlReport = new ExtentSparkReporter( path: "C:\\Users\\Admin\\MyRestAPI\\src\\test\\java\\TelecomProject\\");
        report = new ExtentReports();
        report.attachReporter(htmlReport); // The report should be saved to the html file
        // Configuration of Report
        htmlReport.config().setDocumentTitle("TelecomProjectReport");
        htmlReport.config().setReportName("TelecomProject");
        htmlReport.config().setTheme(Theme.STANDARD);
        htmlReport.config().setTimeStampFormat("dd-MMM-yyyy");
        // Add environment information
        report.setSystemInfo( k: "Machine", v: "Asus");
        report.setSystemInfo( k: "Operating System", v: "Windows 10");
        report.setSystemInfo( k: "Browser", v: "Chrome");
        report.setSystemInfo( k: "User", v: "Ritik");
        //report.setSystemInfo("Application", "Google");
    }
}

```



The screenshot shows the IntelliJ IDEA interface with the project 'MyRestAPI' open. The left sidebar displays the project structure, including the 'src' directory which contains 'main' and 'test' packages. The 'test' package is selected. Inside 'test', there is a 'java' folder containing several files: 'Assignment_API', 'MyMentorCodes', 'MyRestAssuredGroup', 'TelecomProject', 'ContactListAPITest', 'ExtentReportGeneration', 'POJOCLASS', and 'TelecomProjectReport.html'. The 'ExtentReportGeneration.java' file is currently selected and is displayed in the main editor window. The code now includes additional methods for handling test success, failure, skipping, and finishing.

```

public class ExtentReportGeneration implements ITestListener 1 usage
public void onStart(ITestContext context) no usages
{
    //report.setSystemInfo("Application", "Google");
}

public void onTestSuccess(ITestResult result) no usages
{
    test = report.createTest(result.getName()); // create new entity in report (i.e. method pass name)
    test.log(Status.PASS, details: "Test Pass" +result.getName());
}

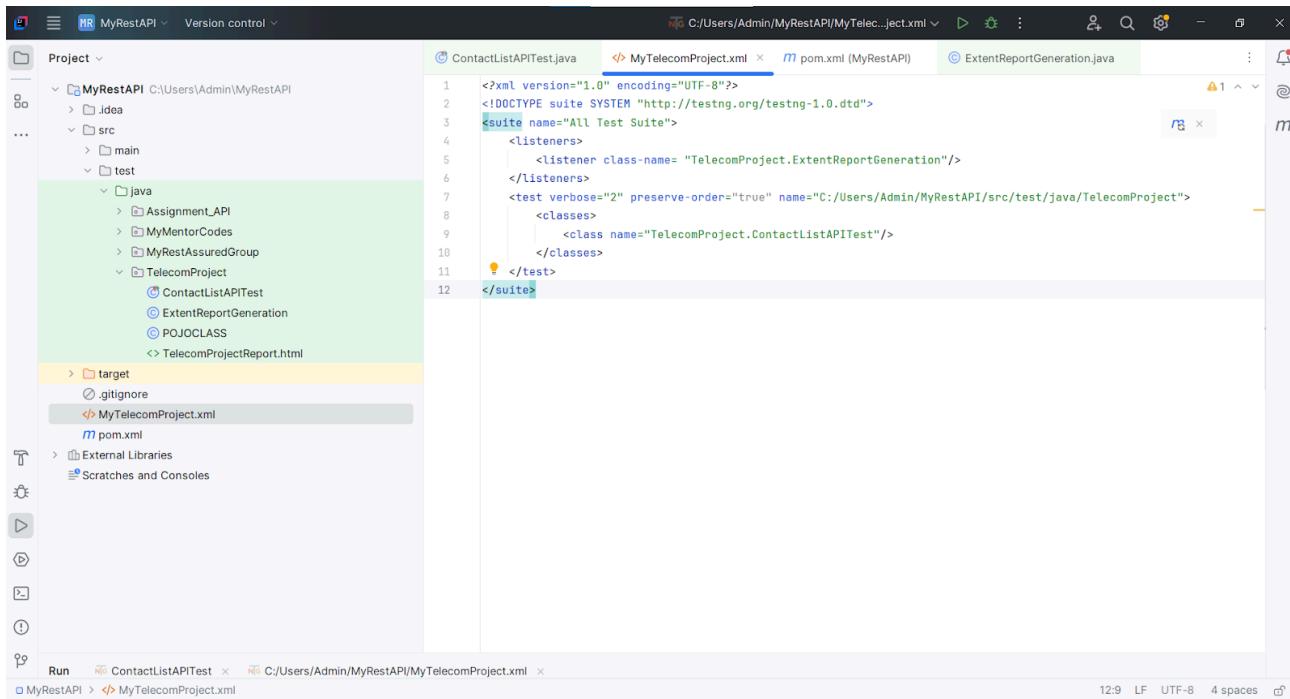
public void onTestFailure(ITestResult result) no usages
{
    test = report.createTest(result.getName()); // create new entity in report
    test.log(Status.FAIL, details: "Test Fail" +result.getName());
    test.log(Status.FAIL, details: "Test Fail because of " +result.getThrowable());
}

public void onTestSkipped(ITestResult result) no usages
{
    test = report.createTest(result.getName());
    test.log(Status.SKIP, details: "Test Skipped" +result.getName());
}

public void onFinish(ITestContext context) no usages
{
    report.flush(); // it will write to report whatever we did so far
}

```

Step 2: Create testng.xml



The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code editor displays the `testng.xml` file:

```

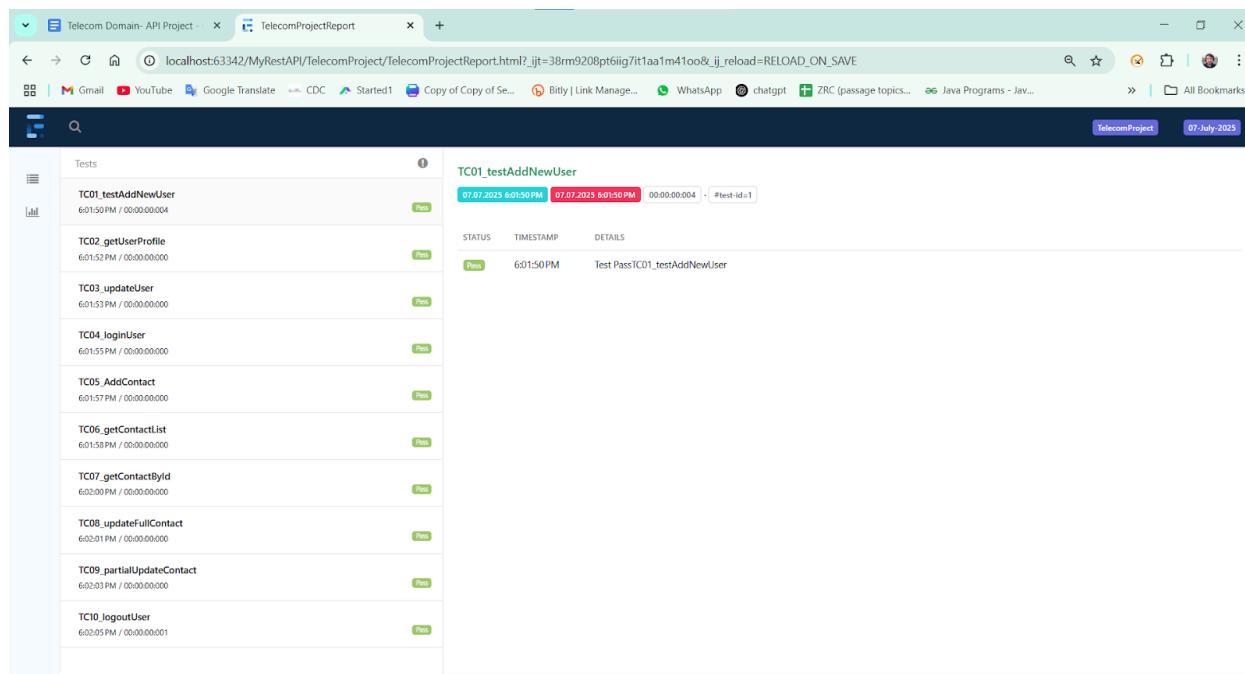
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="All Test Suite">
    <listeners>
        <listener class-name= "TelecomProject.ExtentReportGeneration"/>
    </listeners>
    <test verbose="2" preserve-order="true" name="C:/Users/Admin/MyRestAPI/src/test/java/TelecomProject">
        <classes>
            <class name="TelecomProject.ContactListAPITest"/>
        </classes>
    </test>
</suite>

```

Step 3: Check Report

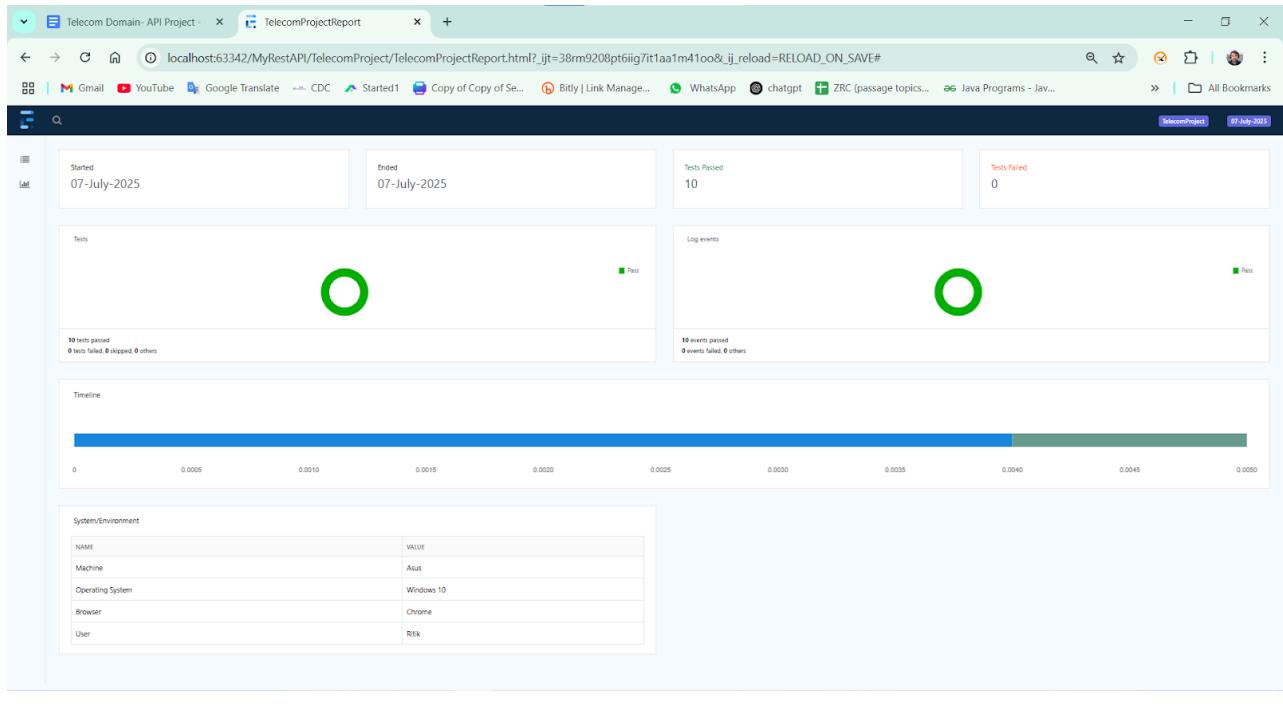
Link:

[http://localhost:63342/MyRestAPI/TelecomProject/TelecomProjectReport.html
?_ijt=38rm9208pt6iig7it1aa1m41oo&_ij_reload=RELOAD_ON_SAVE](http://localhost:63342/MyRestAPI/TelecomProject/TelecomProjectReport.html?_ijt=38rm9208pt6iig7it1aa1m41oo&_ij_reload=RELOAD_ON_SAVE)



The screenshot shows a web browser window displaying a test report. The URL is `localhost:63342/MyRestAPI/TelecomProject/TelecomProjectReport.html?_ijt=38rm9208pt6iig7it1aa1m41oo&_ij_reload=RELOAD_ON_SAVE`. The report lists 10 test cases (TC01 to TC10) with their status, timestamp, and details.

Test Case	Status	Timestamp	Details
TC01_testAddNewUser	Pass	07.07.2025 6:01:50 PM / 06/00/0004	07.07.2025 6:01:50 PM 07.07.2025 6:01:50 PM 00:00:00:04 - #test-id:1
TC02_getUserProfile	Pass	07.07.2025 6:01:52 PM / 06/00/0000	
TC03_updateUser	Pass	07.07.2025 6:01:53 PM / 06/00/0000	
TC04_loginUser	Pass	07.07.2025 6:01:55 PM / 06/00/0000	
TC05_AddContact	Pass	07.07.2025 6:01:57 PM / 06/00/0000	
TC06_getContactList	Pass	07.07.2025 6:01:58 PM / 06/00/0000	
TC07_getContactById	Pass	07.07.2025 6:02:00 PM / 06/00/0000	
TC08_updateFullContact	Pass	07.07.2025 6:02:01 PM / 06/00/0000	
TC09_partialUpdateContact	Pass	07.07.2025 6:02:03 PM / 06/00/0000	
TC10_logoutUser	Pass	07.07.2025 6:02:05 PM / 06/00/0001	



8. Conclusion

In this project, I successfully designed and implemented a comprehensive API testing framework for a telecom domain application using REST Assured, TestNG, and Postman.

The API for the Contact List Application performs well under both valid and invalid conditions. This project covered a wide range of scenarios, including user registration, authentication, contact management, data updates, and logout flows, effectively.

The test cases were structured and executed via `testng.xml` and integrated with Extent Reports for visual test analysis. I used the Newman tool for report generation for Postman. Dynamic data handling (like random emails) and POJO-based payloads ensured reusability and scalability. The automation scripts validated both positive and negative scenarios with proper assertion of status codes, response bodies, and tokens.

----- END -----