Capstone Project: Telecom Domain Project Requirements

Project Overview

Project Name: API Testing for Contact List Application Domain: Telecom

Project Description: This project aims to perform comprehensive testing of the APIs provided by the https://thinking-tester-contact-list.herokuapp.com/ application.

The testing will cover various endpoints to ensure the functionality and reliability of the APIs.

Objectives:

- **Verify the correctness and consistency of API responses.**
- **Validate authentication and authorization mechanisms (if applicable).**
- **Document test cases and results for future reference. Instructions:**

Follow the API document and test all the test cases in Postman apply the chai library assertion and prepare the script in RestAssured & generate the report.

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

RestAssured:

Utility Class:

```
package Telecom_Domain;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.reporter.ExtentSparkReporter;

import com.aventstack.extentreports.reporter.configuration.Theme;
```

```
public class ExtentUtility {
	private static ExtentReports extent; //  1
	private  static  ExtentTest test;  //    2

	public static ExtentReports setupExtentReport()
	{


			ExtentSparkReporter htmlReporter = new
ExtentSparkReporter("target/APISparkReport.html");
			htmlReporter.config().setDocumentTitle("API Test Report");
			htmlReporter.config().setReportName("RestAssured API Testing");
			htmlReporter.config().setTheme(Theme.STANDARD);

			extent=new ExtentReports();
			extent.attachReporter(htmlReporter);
			return extent;



	}

	//ExtentTest
	public static ExtentTest craeteTest(String testName)
	{


			test=extent.createTest(testName);
			return test;
```

```
	}

	public static void  flushReport()
	{

		extent.flush();

	}

}
```

Client Class :

```
package Telecom_Domain;
import static io.restassured.RestAssured.given;
import java.util.HashMap;
import java.util.Map;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.Status;

import io.restassured.RestAssured;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
```

```
public class RestassuredApiTest {
        private static final String BASE_URL = "https://thinking-tester-contact-list.herokuapp.com";
         private static String token;
         private static String Ltoken;
         private static String id;

         private static ExtentReports extent;
         private static ExtentTest test;

         @BeforeTest
         public void setupBaseUrl() {
            RestAssured.baseURI = BASE_URL;
            extent = ExtentUtility.setupExtentReport();
         }

         @Test(priority = 1)
         public void addNewUser() {
            test = ExtentUtility.craeteTest("Add New User_01");

            Map<String, String> userPayload = new HashMap<>();
            userPayload.put("firstName", "Nagaraj");
            userPayload.put("lastName", "Naik");
            userPayload.put("email", "nagarajnaiknw02@gmail.com");
            userPayload.put("password", "NagarajNaik@2025");

            Response response = given()
                  .header("Content-Type", "application/json")
                  .body(userPayload)
                  .when()
                  .post("/users")
                  .then()
```

```
            .extract().response();

    test.log(Status.INFO, "Creating a new user");

    try {
        Assert.assertEquals(response.getStatusCode(), 201, "Status code does not match!");
        System.out.println("Response Body: "+ response.getBody().asString());
        token = response.jsonPath().getString("token");
        Assert.assertNotNull(token, "Token should not be null");

        test.log(Status.PASS, "User created successfully. Token: " + token);
    } catch (AssertionError e) {
        test.log(Status.FAIL, e.getMessage());
        throw e;
    }
}

@Test(priority = 2)
public void testGetUserProfile() {
    test = ExtentUtility.craeteTest("testGetUserProfile_02");

    Response response = given()
            .header("Authorization", "Bearer " + token)
            .when()
            .get("/users/me")
            .then()
            .extract().response();

    try {
        Assert.assertEquals(response.getStatusCode(), 200, "Status code should be 200 OK");
        test.log(Status.PASS, "User profile retrieved successfully.");
```

```
    } catch (AssertionError e) {
        test.log(Status.FAIL, e.getMessage());
        throw e;
    }
}

@Test(priority = 3)
public void updateUserProfile() {
    test = ExtentUtility.craeteTest("UpdateUserProfile_03");

    Map<String, String> userPayload = new HashMap<>();
    userPayload.put("firstName", "Nagarajaa");
    userPayload.put("lastName", "M Naik");
    userPayload.put("email", "nagarajnaiknw02@gmail.com");
    userPayload.put("password", "Nagaraj@2025");

    Response response = given()
            .header("Content-Type", "application/json")
            .header("Authorization", "Bearer " + token)
            .body(userPayload)
            .when()
            .patch("/users/me")
            .then()
            .extract().response();

    try {
        Assert.assertEquals(response.getStatusCode(), 200, "Status code does not match!");
        test.log(Status.PASS, "User profile updated successfully.");
    } catch (AssertionError e) {
        test.log(Status.FAIL, e.getMessage());
        throw e;
```

```
    }
}

@Test(priority = 4)
public void loginUser() {
    test = ExtentUtility.craeteTest("loginUser_04");

    Map<String, String> userPayload = new HashMap<>();
    userPayload.put("email", "nagarajnaiknw02@gmail.com");
    userPayload.put("password", "Nagaraj@2025");

    Response response = given()
            .header("Content-Type", "application/json")
            .body(userPayload)
            .when()
            .post("/users/login")
            .then()
            .extract().response();

    try {
        Assert.assertEquals(response.getStatusCode(), 200, "Status code does not match!");
        Ltoken = response.jsonPath().getString("token");
        Assert.assertNotNull(Ltoken, "Token should not be null");
        test.log(Status.PASS, "User logged in successfully. Token: " + Ltoken);
    } catch (AssertionError e) {
        test.log(Status.FAIL, e.getMessage());
        throw e;
    }
}

@Test(priority = 5)
```

```
public void addContact() {
    test = ExtentUtility.craeteTest("AddContact_05");

    Map<String, String> contactPayload = new HashMap<>();
    contactPayload.put("firstName", "Pavitr");
    contactPayload.put("lastName", "Maa Naik");
    contactPayload.put("birthdate", "2000-01-01");
    contactPayload.put("email", "pavinaiknew901@gmail.com");
    contactPayload.put("Mobilenumber", "9353127257");
    contactPayload.put("street1", "1 Main St");
    contactPayload.put("street2", "Apartment Asss");
    contactPayload.put("city", "Bangalore");
    contactPayload.put("stateProvince", "Karnataka");
    contactPayload.put("postalCode", "560061");
    contactPayload.put("country", "India");

    Response response = given()
            .header("Content-Type", "application/json")
            .header("Authorization", "Bearer " + Ltoken)
            .body(contactPayload)
            .when()
            .post("/contacts")
            .then()
            .extract().response();

    try {
        Assert.assertEquals(response.getStatusCode(), 201, "Status code does not match!");
        id = response.jsonPath().getString("_id");
        Assert.assertNotNull(id, "Contact ID should not be null");
        test.log(Status.PASS, "Contact added successfully. ID: " + id);
    } catch (AssertionError e) {
```

```
                test.log(Status.FAIL, e.getMessage());
                throw e;
            }
        }


    //Get Contact List
    @Test(priority =6)

    public void getContactList()
    {

        test=ExtentUtility.craeteTest("getContactList_06");
      Response response = given()
                .header("Authorization", "Bearer " + Ltoken)
         .when()
         .get("/contacts")
        .then()
         .statusCode(200)
         .extract().response();
try {
      Assert.assertEquals(response.statusCode(), 200, "Status code  be 200 OK");
      JsonPath data=response.jsonPath();
      System.out.println("Json data"+data.prettify());

      System.out.println("Get Contact List Response: " + response.getBody().asString());
      test.log(Status.PASS, "Token received: " + Ltoken);

    }catch (AssertionError e) {
            test.log(Status.FAIL, e.getMessage());
```

```
            throw e;
    }
   }



   //Get Contact

   @Test(priority=7)
public void getContact()

{
          test=ExtentUtility.craeteTest("getContact_07");
           Response response = given()
            .header("Authorization", "Bearer " + Ltoken)
            .when()
            .get("/contacts/" + id)
             .then()
            .statusCode(200)
            .extract().response();

           try {
       System.out.println("Get Contact Response: " + response.getBody().asString());

       Assert.assertEquals(response.statusCode(), 200, "Status code should not be 200 OK");
      JsonPath data=response.jsonPath();
      System.out.println("Json data"+data.prettify());
           }
      catch (AssertionError e) {
         test.log(Status.FAIL, e.getMessage());
         throw e;
```

```
}          }


//Update Contact


   // Update Contact using PUT (full update) including all required fields
   @Test (priority=8)
   public void UpdateContact() {

         test=ExtentUtility.craeteTest("UpdateContact_08");

      Map<String, String> userPayload = new HashMap<>();
      userPayload.put("firstName", "abhi");
      userPayload.put("lastName", "Naik");
      userPayload.put("birthdate", "1998-01-01");
      userPayload.put("email", "Abhinaiknew55@gmail.com");
      userPayload.put("Mobilenumber", "9353129257"); // Ensure this key matches the API spec
      userPayload.put("street1", "1 Main St");
      userPayload.put("street2", "Apartment ss");
      userPayload.put("city", "Bangalore");
      userPayload.put("stateProvince", "Karnataka");
      userPayload.put("postalCode", "560888");
      userPayload.put("country", "India");


      Response response = given()
            .header("Content-Type", "application/json")
            .header("Authorization", "Bearer " + Ltoken)
            .body(userPayload)
```

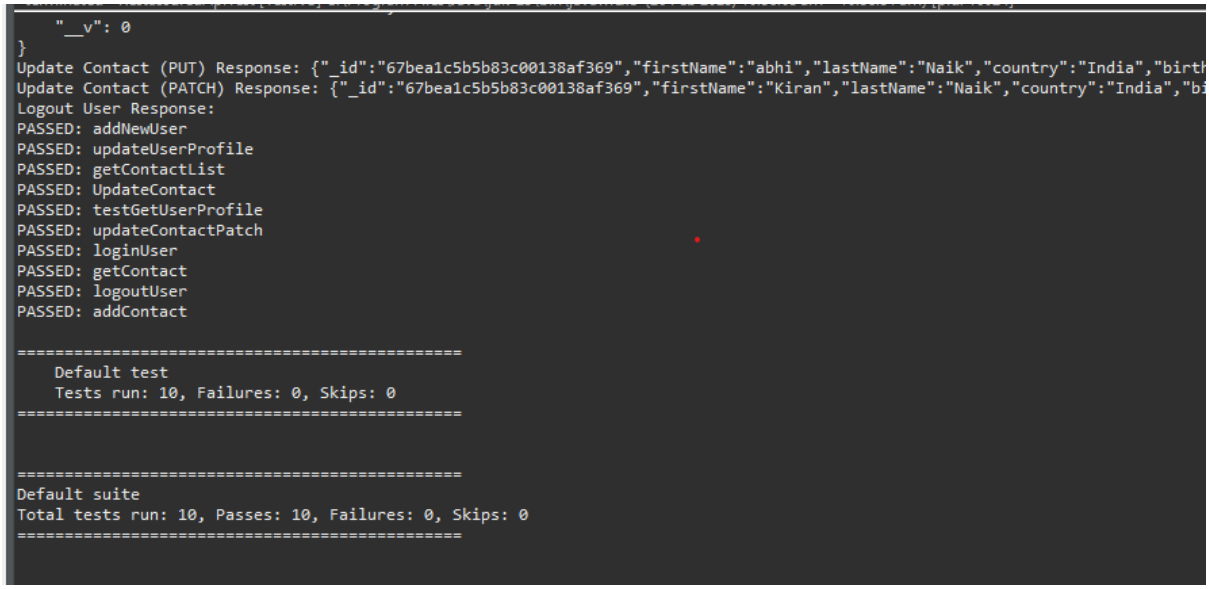
```
            .when()
            .put("/contacts/" + id)
            .then()
            .statusCode(200)
            .extract().response();

    try {

          Assert.assertEquals(response.getStatusCode(), 200, "Status code does not match!");
        test.log(Status.PASS, "Contact list updated successfully.");
         System.out.println("Update Contact (PUT) Response: " + response.getBody().asString());
    // Validate that the email was updated



    }
    catch (AssertionError e) {
        test.log(Status.FAIL, e.getMessage());
        throw e;
}

}


@Test(priority = 9)
public void updateContactPatch()
{

          test=ExtentUtility.craeteTest("updateContactPatch_09");

    Map<String, String> payload = new HashMap<>();
    payload.put("firstName", "Kiran");
```

```
    Response response = given()
            .header("Content-Type", "application/json")
            .header("Authorization", "Bearer " + Ltoken)
            .body(payload)
            .when()
            .patch("/contacts/" + id)
            .then()
            .statusCode(200)
            .extract().response();


    try {


      Assert.assertEquals(response.getStatusCode(), 200, "Status code does not match!");


    System.out.println("Update Contact (PATCH) Response: " + response.getBody().asString());
    String updatedFirstName = response.jsonPath().getString("firstName");
    Assert.assertEquals(updatedFirstName, "Kiran", "First name should be updated to Kiran");
    test.log(Status.PASS, "Contact list updated successfully.");

  } catch (AssertionError e) {
    test.log(Status.FAIL, e.getMessage());
    throw e;

  }}

// Test Case 10: Logout User
```

```
@Test(priority=10)
public void logoutUser() {

        test=ExtentUtility.craeteTest("logoutUser_10");
 Response response = given()
          .header("Authorization", "Bearer " + Ltoken)  // Using the login token
          .when()
          .post("/users/logout")
          .then()
         .extract().response();

   System.out.println("Logout User Response: " + response.getBody().asString());
   try {



      Assert.assertEquals(response.getStatusCode(), 200, "Status code does not match!");
      test.log(Status.PASS, "Log out successfully.");


   }


   catch (AssertionError e) {
         test.log(Status.FAIL, "Logout failed: " + e.getMessage());
      throw e;
   }
}



@AfterTest
public void tearDown() {
        ExtentUtility.flushReport();
```

```
    }

}
```

OUTPUT :



```
    "__v": 0
}
Update Contact (PUT) Response: {"_id":"67bea1c5b5b83c00138af369","firstName":"abhi","lastName":"Naik","country":"India","birth
Update Contact (PATCH) Response: {"_id":"67bea1c5b5b83c00138af369","firstName":"Kiran","lastName":"Naik","country":"India","bi
Logout User Response:
PASSED: addNewUser
PASSED: updateUserProfile
PASSED: getContactList
PASSED: UpdateContact
PASSED: testGetUserProfile
PASSED: updateContactPatch
PASSED: loginUser
PASSED: getContact
PASSED: logoutUser
PASSED: addContact

===============================================
    Default test
    Tests run: 10, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 10, Passes: 10, Failures: 0, Skips: 0
===============================================
```