REST Assured API Automation Project - Assessment

1. Project Abstract

This assessment project focuses on testing an API that manages country-related data using REST Assured. The project revolves around performing CRUD operations and content negotiation for countries, along with additional features like bulk addition and searching countries by name or code.

Participants are provided with a Spring Boot application (CountryApiApplication.java) and a REST controller (CountryController.java) that defines all the necessary endpoints. They are expected to complete the RestUtils.java class by implementing methods that will automate the testing of the API endpoints using REST Assured.

2. Test Cases

Below are the specific API test cases that participants are required to implement as part of the assessment in RestUtils.java. For each test case, the goal is to use REST Assured to send the appropriate HTTP requests, verify the response, and ensure the API behaves as expected.

TC-01: Retrieve All Countries

- **Description**: Fetch a list of all available countries.
- **Endpoint**: GET /api/countries/get/{id}
- **Request Headers**:
    - Accept: application/json
- **Expected Output**: Array of country details.
- **Test**: Verify that the status code is 200 and the response body contains an array of countries.

TC-02: Add a New Country

- **Description**: Submit a new country to be added to the database.
- **Endpoint**: POST /api/countries/add
- **Request Headers**:
    - Content-Type: application/json
- **Request Body**:

{

"name": "Australia"

}

- **Expected Output**: Confirmation message with the new country ID.
- **Test**: Ensure the status code is 201 and the response contains the new country ID.

TC-03: Search Country by Name

- **Description**: Search for a country by its name using a query parameter.
- **Endpoint**: GET /api/countries/search
- **Request Parameters**:
    - name=India
- **Expected Output**: Details of the country matching the search query.
- **Test**: Confirm that the status code is 200 and the response includes the correct country details.

TC-04: Update an Existing Country

- **Description**: Update the details of an existing country.
- **Endpoint**: PUT /api/countries/update/{id}
- **Request Headers**:
    - Content-Type: application/json
- **Request Body**:

{

"name": "New Zealand"

}

- **Expected Output**: Confirmation of the updated country details.
- **Test**: Verify that the status code is 200 or 204 and the response body confirms the country details have been updated.

TC-05: Delete a Country

- **Description**: Remove a country from the database.
- **Endpoint**: DELETE /api/countries/delete/{id}

- **Expected Output**: A confirmation message that the country has been deleted.
- **Test**: Ensure the status code is 204 and that the country is no longer in the list of countries.

TC-06: Bulk Add Multiple Countries

- **Description**: Add multiple countries in a single request.
- **Endpoint**: POST /api/countries/bulkAdd
- **Request Headers**:
    - Content-Type: application/json
- **Request Body**:

```
{
 "1": "Brazil",
 "2": "South Africa",
 "3": "Argentina"
}
```

- **Expected Output**: Confirmation that all countries have been added.
- **Test**: Ensure the status code is 201 and the response body confirms all countries were added.

TC-07: Partially Update a Country

- **Description**: Partially update the name of an existing country.
- **Endpoint**: PATCH /api/countries/patch/{id}
- **Request Headers**:
    - Content-Type: application/json
- **Request Body**:

```
{
 "name": "United Kingdom"
}
```

- **Expected Output**: Confirmation of the updated country name.

- **Test**: Verify that the status code is 200 and the country name has been updated.

TC-08: Retrieve Country by Code

- **Description**: Fetch a country by its code.
- **Endpoint**: GET /api/countries/code/{code}
- **Request Parameters**:
  - code=IN
- **Expected Output**: Country details for the provided code.
- **Test**: Validate that the status code is 200 and the response body contains the details of the correct country.

TC-09: Handle Invalid Country ID

- **Description**: Test the behavior when retrieving a country with an invalid ID.
- **Endpoint**: GET /api/countries/get/{id}
- **Expected Output**: Error message indicating the country was not found.

Test: Ensure the status code is 404 and the response body contains an appropriate error message.

EXECUTION STEPS TO FOLLOW

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **To build your project use command:**

   mvn clean package -Dmaven.test.skip
4. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**

   java -jar <your application jar file name>
5. **This editor Auto Saves the code.**
6. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
7. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
8. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN. Please use 127.0.0.1 instead of localhost to test rest endpoints.**
9. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
10. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**

11. **To login to mysql instance: Open new terminal and use following command:**

    a. **sudo systemctl enable mysql**

    b. **sudo systemctl start mysql**

    NOTE: After typing any of the above commands you might encounter any warnings.

    >> Please note that this warning is expected and can be disregarded. Proceed to the next step.

    c. **mysql -u root -p**

    The last command will ask for password which is 'pass@word1'

12. **Mandatory: Before final submission run the following command:**

    mvn test

13. **NOTE: Ensure the project is in a running state before executing the mvn test command. This ensures that all configurations and dependencies are properly set up for the tests to run successfully.**

14. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**