| REST-ASSURED  ( API - AUTOMATION PROJECT – PL1) |  |
| --- | --- |

Pre-requisite:

Before you start working on your project, execute the runner file present in your project folder (Simply by double click). This is mandatory.

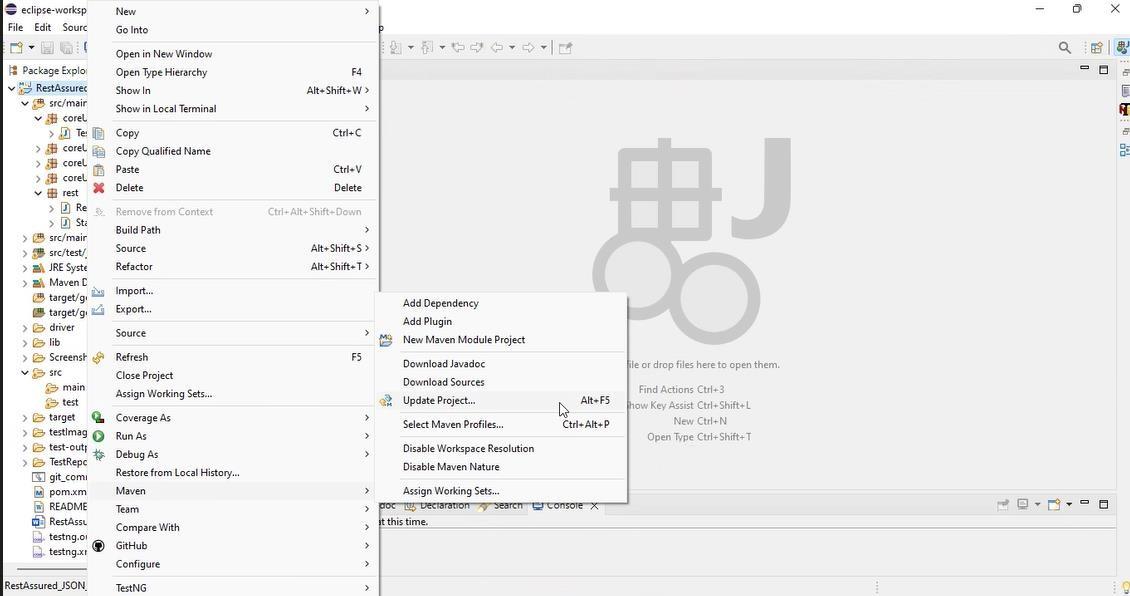
|  |
| --- |

This will launch a command terminal for you where it will keep on pushing your updated code to GIT on regular intervals. Keep that command terminal open at backend and you can continue working on your project.

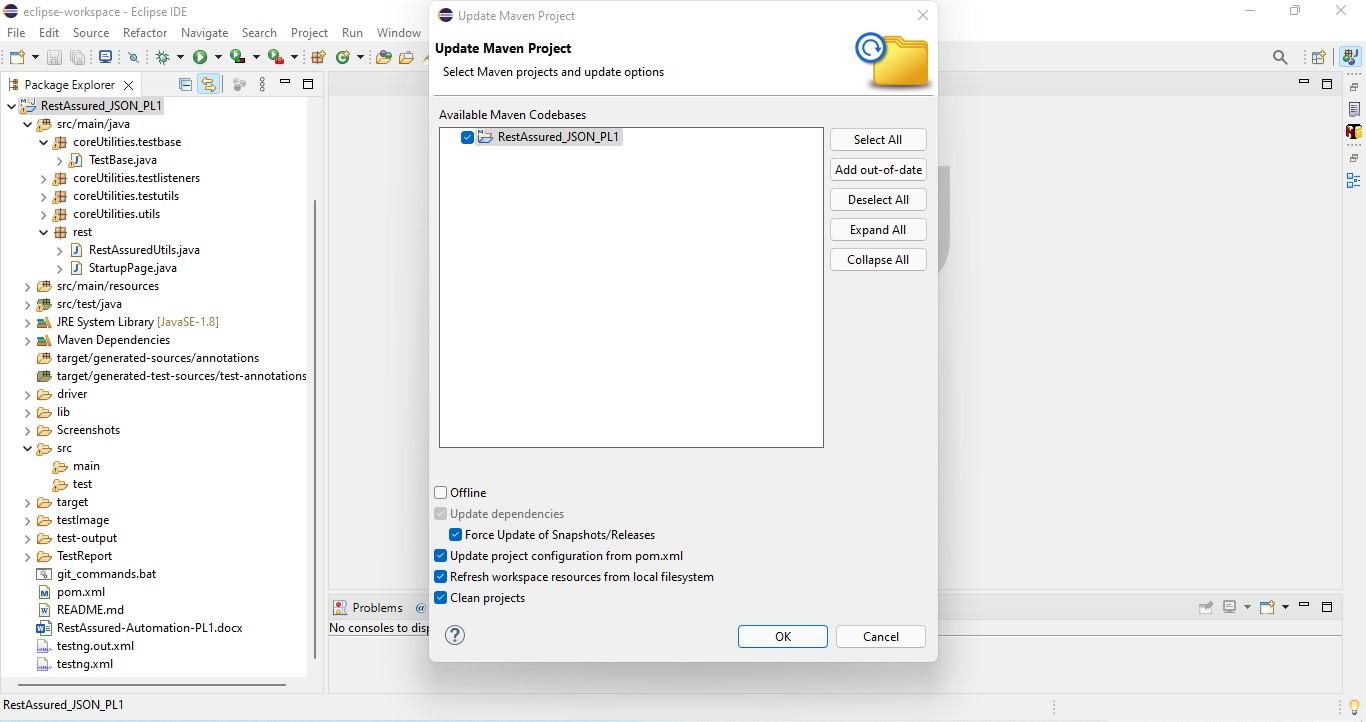
|  |
| --- |

As soon as you import project in eclipse, update the project using maven update option as below. This is to resolve issue if any maven dependency not downloaded properly:

1. Right click on project : Go to “Maven” : Select “Update Project”



1. In Update Maven Project Box Select “Force Update of Snapshots/Releases” and click OK



Template Code Structure:

* Below are the packages and files you will be required to work upon.
* Other Files and packages you can ignore.
* In other Files and packages do not do any changes. It would affect your evaluation.
* You are not required to work in “Test” Folder. Files there are noneditable. Editing those files and trying to save them will throw error and would affect your evaluation.

| **Package** | **Class/File** | **Description** |
| --- | --- | --- |
| src/main/java/coreUtilities/utils/ | FileOperations.java | 1. It contains methods to read data from Excel files. 2. The method is in templated form. 3. You will be required to implement these methods as the first activity, because even the URL to navigate to is read using these methods. |
| /src/main/java/pages | ApiUtil.java | 1. All core activities to be performed here. 2. The comments associated with each templated method here describe the expectation. 3. Declare any 4. variable/object you need to share data/status between different methods. 5. Do not modify the signature of methods declared here. 6. You can create additional supportive common methods in 7. CommonEvents class. |
| /src/main/resources/ | Config.xlsx | 1. Data present to be used in creating a new Post. |
|  | expected\_data.xlsx | 1. Contains data to fill in the form |
| /src/main/java/coreUtilities/utils | CommonEvents.java | 1. Contains all common activities. 2. Certain templated common methods are declared here. 3. You implement them as per your needs. 4. You can add any additional method for common activity here |
|  | Testng.xml | Execution needs to be kick-started from TestNG.xml |

PROBLEM STATEMENT

Need to automate the following activities using RestAssured.

Key Activities to implement:

| # | **Summary** | **Action** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Retrieve sub units tree with Valid Cookie Authentication using method:  getSubunitsTree(String endpoint, String cookieValue, Map<String, String> body) | 1. Construct the final URL by combining the BASE\_URL and the provided endpoint parameter:    1. https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/subunits?mode=tree 2. Initialize a RequestSpecification using RestAssured:    1. Add a cookie with name "orangehrm" and the value passed as cookieValue.    2. Set the "Content-Type" header to "application/json". 3. Trigger a GET request to the above URL. 4. Create a CustomResponse object with:    1. Full Response    2. Extracted statusCode    3. Extracted statusLine    4. Extracted list id    5. Extracted list unitId    6. Extracted list name    7. Extracted list level    8. Extracted list children 5. Return the CustomResponse object. | 1. Method should return a fully initialized CustomResponse object with:    1. statusCode = 200    2. status = OK (or valid status line like "HTTP/1.1 200 OK")    3. A non-null and non-empty languages list    4. A non-null and non-empty id list    5. A non-null and non-empty unitId list    6. A non-null and non-empty unitId name    7. A non-null and non-empty unitId level    8. A non-null and non-empty unitId children    9. The complete raw Response object |
| **2** | Retrieve admin skills using session cookie authentication through method:  getAdminSkills(String endpoint, String cookieValue, Map<String, String> body) | 1. Construct the complete URL by concatenating the BASE\_URL and the provided endpoint:    1. https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/skills?limit=50&offset=0 2. Create a RequestSpecification using RestAssured:    1. Add a session cookie named "orangehrm" with the provided cookieValue.    2. Add a header "Content-Type" with the value "application/json". 3. Send a GET request to the constructed URL. 4. Wrap the extracted fields into lists<Object>:    1. id, name, description 5. Create and return a CustomResponse object initialized with:    1. The complete Response    2. Extracted statusCode    3. Extracted statusLine    4. id, name, description | 1. The method should return a CustomResponse object with:    1. statusCode = 200    2. status equal to a valid HTTP status line like "HTTP/1.1 200 OK"    3. Non-null and non-empty lists for:       1. id       2. name       3. description    4. The complete raw Response object |
| **3** | Retrieve admin educations using session-based authentication through the method: **getAdminEdu(String endpoint, String cookieValue, Map<String, String> body)** | 1. Construct the final URL by combining the BASE\_URL and the provided endpoint:    * https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/educations?limit=50&offset=0 2. Create a RequestSpecification using RestAssured:    * Add a session cookie named "orangehrm" using the provided cookieValue.    * Set the request header "Content-Type" to "application/json". 3. Wrap each of these fields into separate List<Object> collections:    * id    * name 4. Create a CustomResponse object using:  * The complete Response object * Extracted statusCode and statusLine * id and name field lists  1. Return the populated CustomResponse object. | 1. The method should return a CustomResponse object containing:    * statusCode = 200    * A valid status line such as "HTTP/1.1 200 OK"    * Non-null and non-empty lists for the following keys: id and name    * The complete Response object from the API call |
| **4** | Retrieve admin licenses using session-based authentication through method:  **getAdminLicenses(String endpoint, String cookieValue, Map<String, String> body)** | 1. Construct the final URL by concatenating BASE\_URL and the given endpoint:   https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/licenses?limit=50&offset=0   1. Create a **RequestSpecification** using **RestAssured**:    * Include the session cookie "orangehrm" using the provided cookieValue    * Add the header "Content-Type" with value "application/json" 2. Trigger a **GET** request to the above URL. 3. Store each field as a List<Object>:    * id, name 4. Return a CustomResponse object populated with:    * Full Response object    * statusCode and statusLine    * Extracted field lists | * Method must return a valid CustomResponse object containing:   + statusCode = 200   + status like "HTTP/1.1 200 OK"   + Non-null and non-empty lists for:     - id     - name   + The full raw Response object |
| **5** | Retrieve admin languages using session-based cookie authentication via method:  **getAdminLanguages(String endpoint, String cookieValue, Map<String, String> body)** | 1. Construct the final URL by appending the endpoint to the base URL:   https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/licenses?limit=50&offset=0   1. Use **RestAssured** to initialize a RequestSpecification:    * Add the session cookie "orangehrm" using the provided cookieValue    * Set the request header "Content-Type" to "application/json" 2. If a request body (body) is not null, include it in the request. 3. Store all extracted fields in respective List<Object> collections.    * id    * name 4. Return a CustomResponse object initialized with:    * The full Response    * statusCode and statusLine | * The method must return a CustomResponse containing:   + statusCode = 200   + A valid status line like "HTTP/1.1 200 OK"   + An CustomResponse object with non-null and non-empty lists for:     - id     - name   + The full raw Response object |
| **6** | Delete valid user using session-based authentication via method:  **deleteValidUsers(String endpoint, String cookieValue, String requestBody)** | 1. Construct the complete URL by appending the endpoint to the base URL:   https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/users   1. Initialize a **RequestSpecification** using **RestAssured**:    * Add the session cookie "orangehrm" with the provided cookieValue.    * Set the header "Content-Type" to "application/json". 2. If a request body (body) is provided (not null), include it in the request using .body(). 3. Trigger a **DELETE** request to the constructed URL and extract the **Response**. 4. Create and return a **CustomResponse** object containing:    * The full Response    * The extracted statusCode and statusLine | * The method should return a CustomResponse object with:   + statusCode = 200   + A valid status string (e.g., "HTTP/1.1 200 OK")   + The complete raw Response object |
| **7** | Create new job titles using session-based authentication through method: **postNewJobTitles(String endpoint, String cookieValue, String requestBody)** | 1. Construct the final URL using the base URL and the language package ID as a path variable:   https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/job-titles   1. Initialize a **RequestSpecification** using **RestAssured**:    * Add the "orangehrm" cookie with the given cookieValue.    * Set the request header "Content-Type" to "application/json". 2. If the body is not null, include it using .body(). 3. Trigger a **POST** request to the endpoint and extract the **Response**. 4. Store these values in individual string variables:    * title 5. Return a **CustomResponse** object initialized with:    * The full Response    * statusCode and statusLine    * Extracted string values: title | * The method should return a valid CustomResponse containing:   + statusCode = 200   + A valid HTTP status (e.g., "HTTP/1.1 200 OK")   + Non-empty string values for:     - title   + The complete raw Response object |
| **8** | Update job title by ID using session-based authentication via method: **putJobTitleByID(String endpoint, String cookieValue, String requestBody)** | 1. Construct a request body in JSON format containing the retrieved ID: 2. Compose the full URL by appending the endpoint to the base URL:   https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/job-titles/{id}   1. Initialize a **RequestSpecification** using **RestAssured**:    * Add a session cookie named "orangehrm" with the provided cookieValue    * Set the "Content-Type" header to "application/json" 2. Attach the constructed requestBody to the request. 3. Send a **PUT** request to the full URL and extract the **Response** object. 4. Return a **CustomResponse** object with:    * The full Response object    * Extracted statusCode and statusLine    * The id, title, description, note list | * The method should return a CustomResponse containing:   + statusCode = 200   + A valid HTTP status line such as "HTTP/1.1 200 OK"   + A non-null and non-empty List<Object>     - id     - title     - description     - note   + The full raw Response object |
| **9** | Create admin pay-grades with session-based cookie authentication using method: **postUniquePaygrade(String endpoint, String cookieValue, String requestBody)** | 1. Construct the complete URL by appending the endpoint to the base URL:   https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/pay-grades   1. Initialize a **RequestSpecification** using **RestAssured**:    * Add the cookie "orangehrm" with the value provided by cookieValue.    * Set the header "Content-Type" to "application/json".    * Attach the JSON request body using .body(requestBody). 2. Trigger a **POST** request to the above endpoint and extract the **Response**. 3. Parse the response using **JsonPath** and retrieve the "data" object as a Map<String, Object> which holds the current module status. 4. Return a **CustomResponse** object initialized with:    * The full Response object    * statusCode and statusLine    * Extracted List of id, name | * The method must return a CustomResponse object that includes:   + statusCode = 200   + A valid status line (e.g., "HTTP/1.1 200 OK")   + A non-null and non-empty List<Object>     - id     - name   + The complete raw Response object |
| **10** | Delete paygrade by ID using a DELETE request with session-based authentication via method: **deletePaygradeById(String endpoint, String cookieValue, String requestBody)** | 1. Construct the complete URL by appending the endpoint to the base URL:   <https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/users>   1. Initialize a **RequestSpecification** using **RestAssured**:    * Add the session cookie "orangehrm" with the provided cookieValue.    * Set the header "Content-Type" to "application/json". 2. Include it in the request using .body(). Body should be like this: {"ids": [{id}]} 3. Trigger a **DELETE** request to the constructed URL and extract the **Response**. 4. Create and return a **CustomResponse** object containing:    * The full Response    * The extracted statusCode and statusLine | * The method should return a valid CustomResponse object containing:   + statusCode = 200   + A valid status string like "HTTP/1.1 200 OK"   + Non-null, correctly mapped values for:     - name     - url     - clientId     - clientSecret   + The full Response object |

NOTE: "Please do not delete any file in the src folder. But you are free to add any other file".

Expectations:

* + 1. **Learners should write automation scripts using Java and REST Assured to automate the API testing for all the provided methods (e.g., GET, POST, PUT, DELETE).** In other words, the automation script should perform all mentioned API interactions, including validation of responses.
    2. **Learners should not use any pre-built libraries or tools to validate API responses (e.g., JSON schema validation tools).** They should manually validate the response content (e.g., status codes, response body, etc.) by writing their own logic for assertion.



IMPLEMENTATION/FUNCTIONAL REQUIREMENT

**1.1 CODE QUALITY/OPTIMIZATIONS**

* + - 1. Associates should have written clean code that is readable.
      2. Associates need to follow SOLID programming principles.

EXECUTION STEPS TO FOLLOW

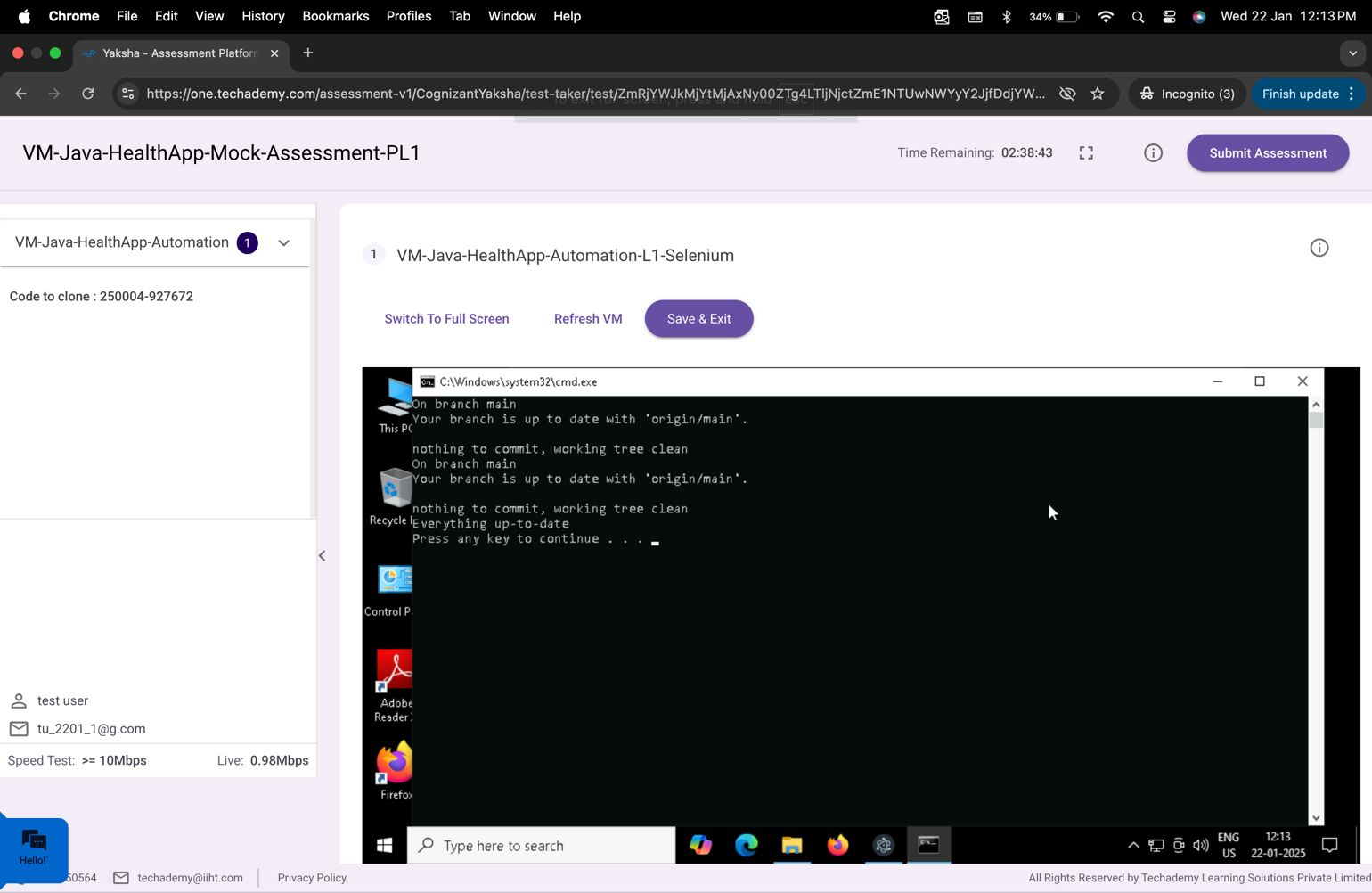


1. **You are mandatory required to run test cases for applications before final submission. Without this project evaluation will not happen.**
2. **You can launch test cases any time as follows: Right-click on testng.xml and run TestNGSuite.**

**A screenshot of a computer

Description automatically generated**

1. To do the final submission of the assessment :
   1. Press escape to come out of Fullscreen mode.
   2. Submit the assessment.



After the successful submission of the assessment, you will get a confirmation message displayed on your screen.

===============================================================================

All the Best