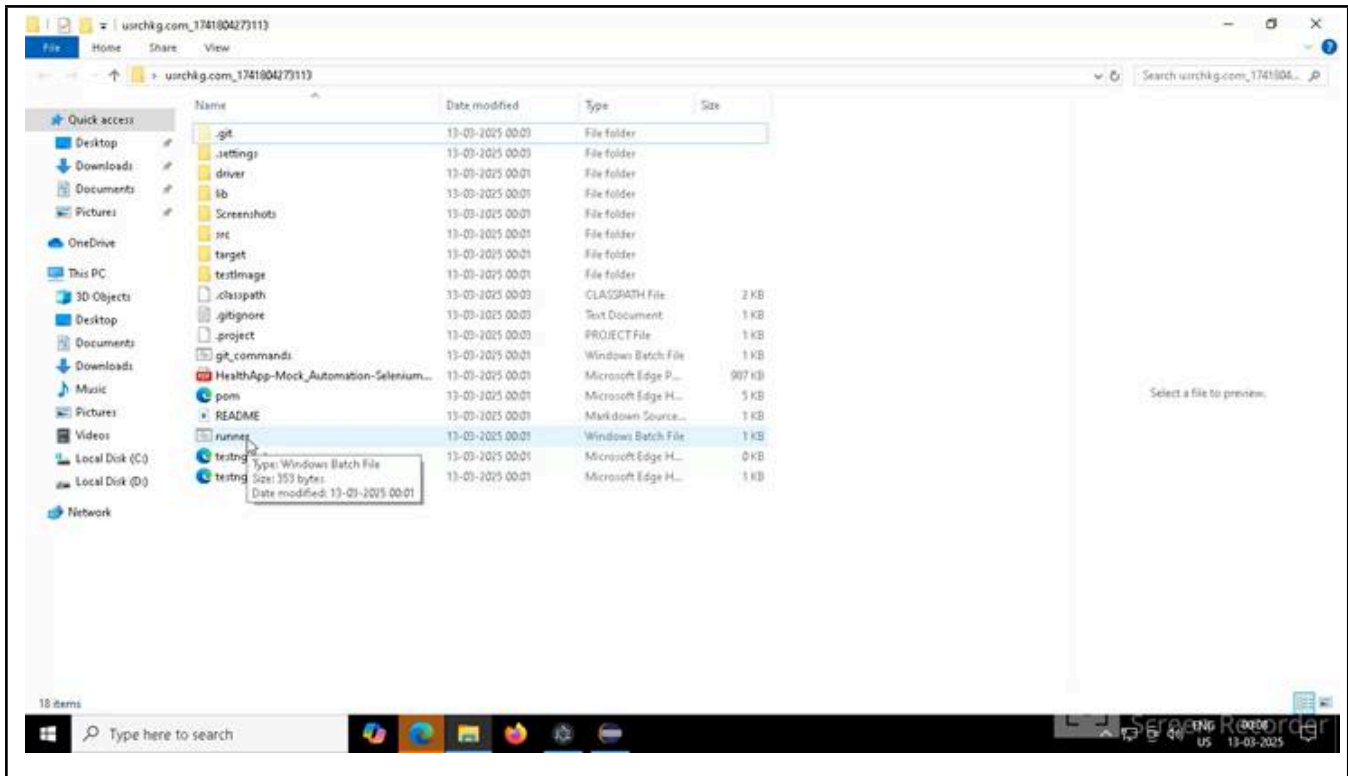# REST-ASSURED
(API-AUTOMATION PROJECT – PL1_5)

Pre-requisite:

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


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 the backend and you can continue working on your project.
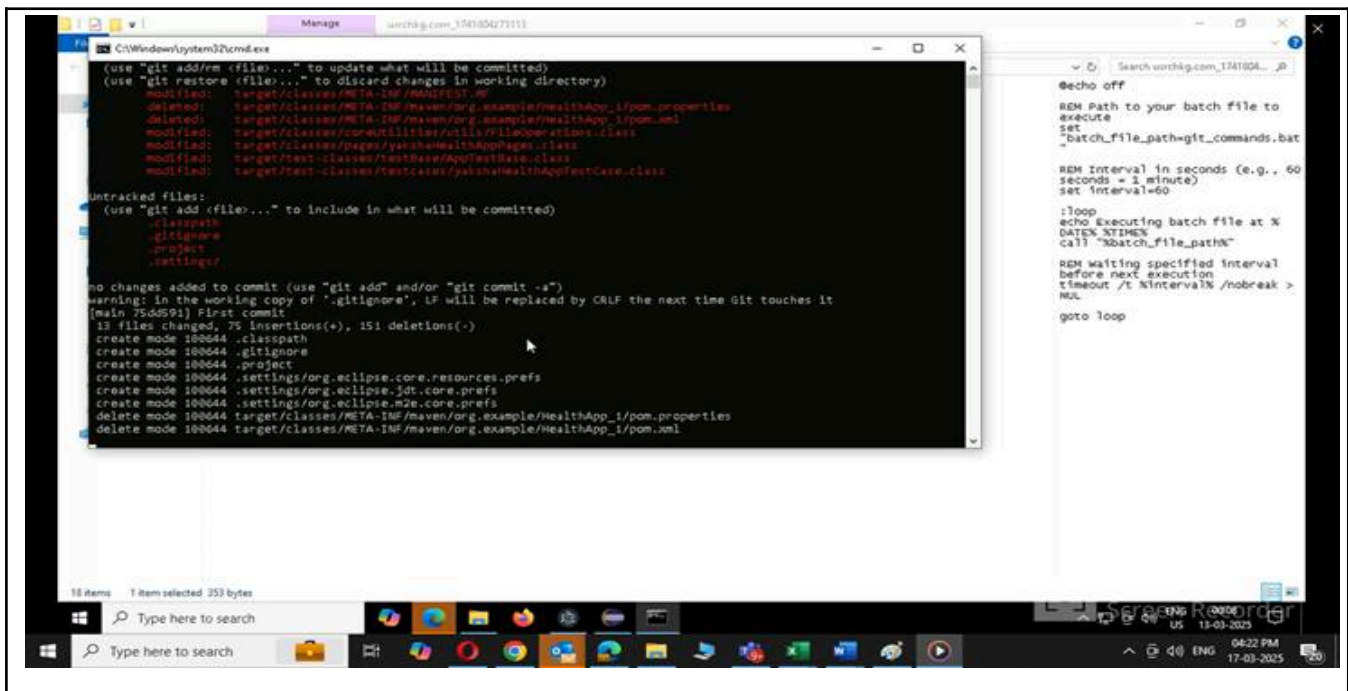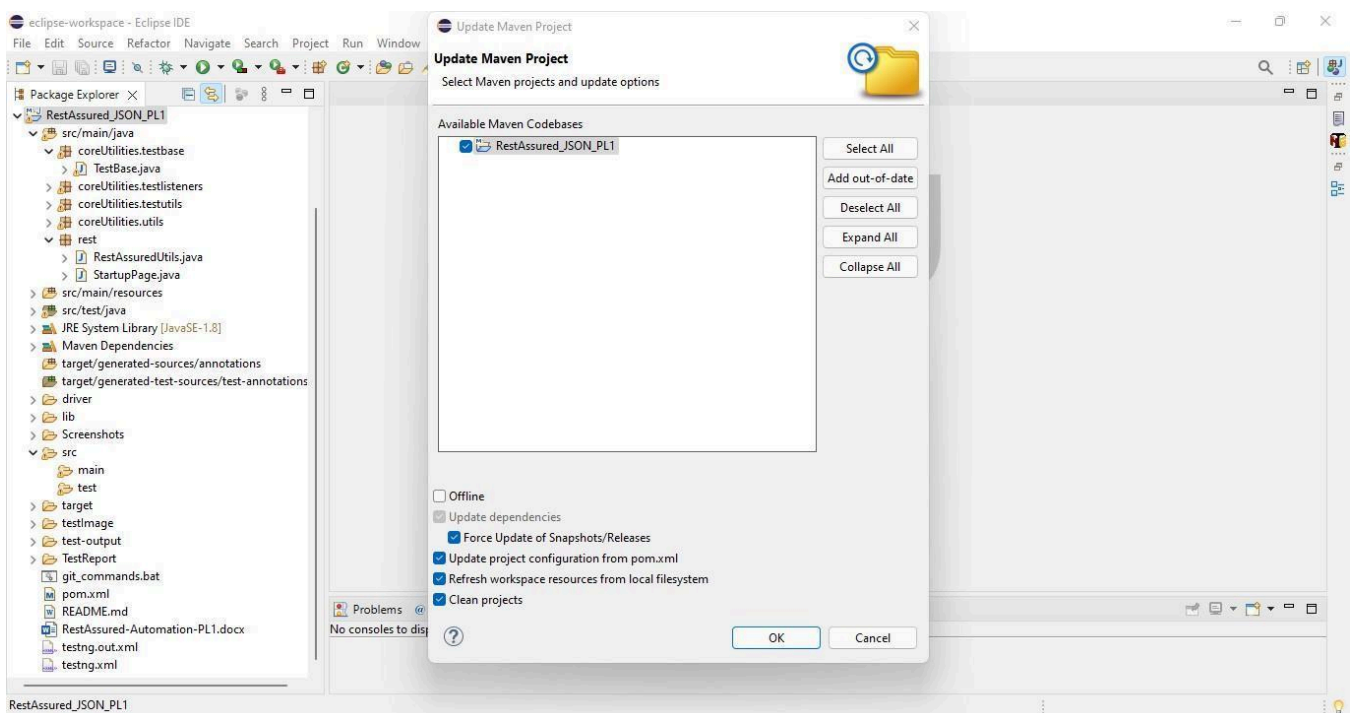
As soon as you import a project in eclipse, update the project using the maven update option as below. This is to resolve issue if any maven dependency not downloaded properly:

1. Right click on project : Go to "Maven" : Select "Update Project"
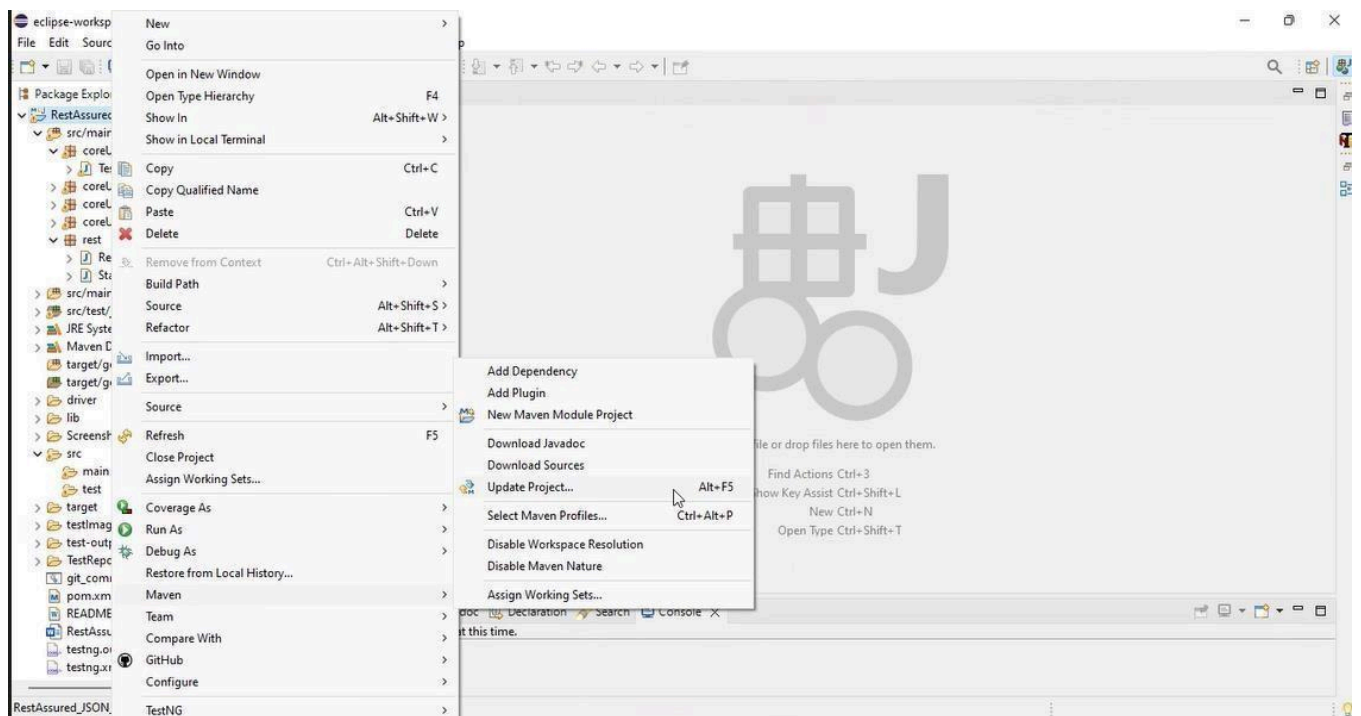
2. In Update Maven Project Box Select "Force Update of Snapshots/Releases" and click OK

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 the "Test" Folder. Files there are noneditable. Editing those files and trying to save them will throw errors 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 already implemented.
/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. Signature means method name, method return type, number of arguments, any exception thrown (if present, if not then please don't add).** 6. Do not modify the signature of methods declared here. 7. You can create additional supportive common methods in 8. CommonEvents class.
/src/main/resources/	config.properties	1. Data present to be used in creating a new Post.
	testData.xlsx	1. Data is present for creating post and put requests
/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: a. https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/subunits?mode=tree 2. Initialize a RequestSpecification using RestAssured: a. Add a cookie with name "orangehrm" and the value passed as cookieValue. b. Set the "Content-Type" header to "application/json". 3. Trigger a GET request to the above URL. 4. Parse the response using `JsonPath` and retrieve the `data` field as a `List<Map<String, Object>>`. 5. From each map, extract and wrap values into separate lists as follows: ● `id` to be extracted and saved in **ids** list. ● `unitId` to be extracted and saved in **unitIds** list. ● `name` to be extracted and saved in **names** list. ● `level` to be extracted and saved in **levels** list. ● `children` to be extracted and saved in **children** list. 6. Create and return a **CustomResponse** object containing: ● The full `Response`	1. Method should return a fully initialized CustomResponse object with: a. statusCode = 200 b. status = OK (or valid status line like "HTTP/1.1 200 OK") c. A non-null and non-empty ids list d. A non-null and non-empty unitIds list e. A non-null and non-empty names list f. A non-null and non-empty levels list g. A non-null and non-empty childrens list h. The complete raw Response object

		● The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`. ● All extracted lists (**ids, unitIds, names, levels, children**)	

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:
		   a. https://opensource-demo.orangehrmlive.com/web/index.php/api/v2/admin/skills?limit=50&offset=0
		2. Create a RequestSpecification using RestAssured:
		   a. Add a session cookie named "orangehrm" with the provided cookieValue.
		   b. Add a header "Content-Type" with the value "application/json".
		3. Send a GET request to the constructed URL.
		
		4. Parse the response using `JsonPath` and retrieve the `data` field as a `List<Map<String, Object>>`.
		
		5. From each map (representing a skill), extract and wrap values into separate lists as follows:
		   ● `id` to be extracted and saved in **ids** list.
		   ● `name` to be extracted and saved in **names** list.
		   ● `description` to be extracted and saved in **descriptions** variable.
		
		6. Create and return a CustomResponse object initialized with:
		   a. The complete Response
		   b. The extracted `statusCode` in variable `statusCode`
			1. The method should return a CustomResponse object with:
			   a. statusCode = 200
			   b. status equal to a valid HTTP status line like "HTTP/1.1 200 OK"
			   c. Non-null and non-empty lists for:
			      i. ids
			      ii. names
			      iii. descriptions
			   d. The complete raw Response object

		and `statusLine` in variable `status`. c. All extracted lists (**ids**, **names**, **descriptions**)	

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. Parse the response using `JsonPath` and retrieve the `data` field as a `List<Map<String, Object>>`.
		
		4. From each map (representing an education entry), extract and wrap values into separate lists as follows:
		   ● `id` to be extracted and saved in **ids** list.
		   ● `name` to be extracted and saved in **names** list.
		
		5. Create a CustomResponse object using:
		   ● The complete Response object
		   ● The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`.
		
		   ● Both extracted lists (**ids** and **names**)
		6. 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:
			     ➔ ids and names
			   ● 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 2. Create a **RequestSpecification** using **RestAssured**:
 ● Include the session cookie "orangehrm" using the provided cookieValue
 ● Add the header "Content-Type" with value "application/json"
 3. Trigger a **GET** request to the above URL.
 4. Parse the response using `JsonPath` and retrieve the `data` field as a `List<Map<String, Object>>`.
 5. From each map (representing a license entry), extract and wrap values into separate lists as follows:
 ● `id` to be extracted and saved in **ids** list.
 ● `name` to be extracted and saved in **names** list.
 6. Return a CustomResponse object populated with:
 ● Full Response object
 ● The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`.
 ● Both extracted lists (**ids** and **names**) | ● Method must return a valid CustomResponse object containing:
 ● statusCode = 200
 ● status like "HTTP/1.1 200 OK"
 ● Non-null and non-empty lists for:
 ▪ ids
 ▪ names <br> o 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`

2. Use **RestAssured** to initialize a `RequestSpecification`:

    ● Add the session cookie `"orangehrm"` using the provided `cookieValue`

    ● Set the request header `"Content-Type"` to `"application/json"`

3. If a request body (`body`) is not null, include it in the request.

4. Parse the response using `JsonPath` and retrieve the `data` field as a `List<Map<String, Object>>`.

5. From each map (representing a language entry), extract and wrap values into separate lists as follows:
    ● `id` to be extracted and saved in **ids** list.
    ● `name` to be extracted and saved in **names** list.

6. Return a `CustomResponse` object initialized with:

    ● The full `Response`

    ● The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`.

    ● Both extracted lists (**ids** and **names**) | ● 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:
        ▪ `ids`
        ▪ `names`

● 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` 2. Initialize a **RequestSpecification** using **RestAssured**: - Add the session cookie `"orangehrm"` with the provided `cookieValue`. - Set the header `"Content-Type"` to `"application/json"`. 3. body must be included in request object using `.body()`. 4. Trigger a **DELETE** request to the constructed URL and extract the **Response**. 5. Create and return a **CustomResponse** object containing: - The full `Response` - The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`.	- 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` 2. Initialize a **RequestSpecification** using **RestAssured**: Add the `"orangehrm"` cookie with the given `cookieValue`.	- The method should return a valid `CustomResponse` containing: `statusCode = 200` - A valid HTTP `status` (e.g., `"HTTP/1.1 200 OK"`) - Non-empty object values for: `title2` - The complete raw `Response` object

		● Set the request header "Content-Type" to "application/json".	
		3. body must be included in request object using `.body()`..	
		4. Trigger a **POST** request to the endpoint and extract the **Response**.	
		5. Parse the JSON response using **JsonPath** and retrieve the "data" object as a `Map<String, Object>`.	
		6. From this map: ● **title** to be extracted and saved in **title2** variable.	
		7. Return a **CustomResponse** object initialized with: ● The full `Response` ● The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`. ● The extracted **title2**	
8	Update job title by ID using session-based authentication via method: **putJobTitleByID(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/job-titles/{id}` 2. Initialize a **RequestSpecification** using **RestAssured**: ● Add a session cookie named "orangehrm" with the provided `cookieValue` ● Set the "Content-Type" header to "application/json"	● 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>` ▪ ids ▪ titles ● The full raw `Response` object

3. body must be included in request object using `.body()`.

4. Send a **PUT** request to the full URL and extract the **Response** object.

5. Parse the JSON response using **JsonPath** and retrieve the `"data"` object as a `Map<String, Object>`.

6. From this map:
   - **id** to be extracted and saved into the `ids` list
   - **title** to be extracted and saved into the `titles` list
   - **description** to be extracted and saved into the `description` variable
   - **note** to be extracted and saved into the `note` variable
   - **jobSpecification** to be extracted and saved into the `jobSpecification` map

7. Return a **CustomResponse** object with:
   - The full `Response` object
   - The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`.
   - The List<Object> of `ids`
   - The List<Object> of `titles`
   - The Object type `description`
   - The Object type `note`
   - The `Map<String, Object>` `jobSpecification`

| 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`

2. Initialize a **RequestSpecification** using **RestAssured**:

    ● Add the cookie `"orangehrm"` with the value provided by `cookieValue`.

    ● Set the header `"Content-Type"` to `"application/json"`.

    ● body must be included in request object using `.body()`.

3. Trigger a **POST** request to the above endpoint and extract the **Response**.

4. Parse the JSON response using **JsonPath** and retrieve the `"data"` object as a `Map<String, Object>`.

5. From this map:
    ● **id** to be extracted and saved into the variable `id2`
    ● **name** to be extracted and saved into the variable `name`

6. Return a **CustomResponse** object initialized with:

    ● The full `Response` object

    ● The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`.

    ● The extracted `id2` and `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 Object

        ▪ id2

        ▪ name

    ● The complete raw `Response` object |

| 1 0 | 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/pay-grades

 2. Initialize a **RequestSpecification** using **RestAssured**:
 • Add the session cookie `"orangehrm"` with the provided `cookieValue`.
 • Set the header `"Content-Type"` to `"application/json"`.

 3. body must be included in request object using `.body()`.

 4. Trigger a **DELETE** request to the constructed URL and extract the **Response**.
 5. Create and return a **CustomResponse** object containing:
 • The full `Response`
 • The extracted `statusCode` in variable `statusCode` and `statusLine` in variable `status`. | ● The method should return a valid `CustomResponse` object containing:

 ● `statusCode = 200`

 ● A valid `status` string like `"HTTP/1.1 200 OK"`

 ● 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.**
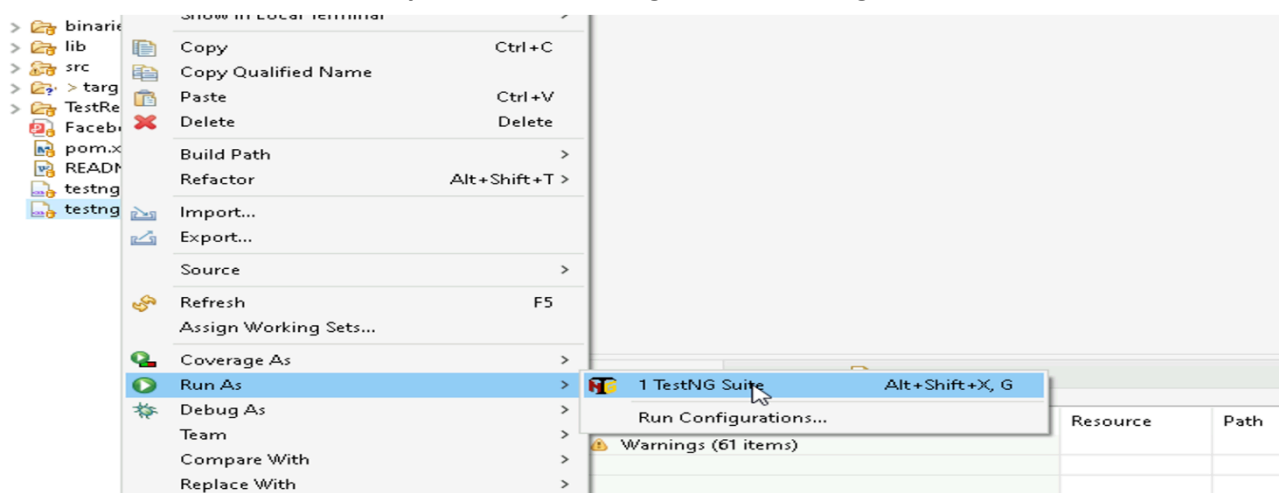
NOTE:
- **When executing test cases via TestNG.xml, it launches the application UI using Selenium WebDriver. This is intentional behavior to facilitate cookie extraction for authentication.**

3. To do the final submission of the assessment :
   a. Press escape to come out of Fullscreen mode.
   b. Submit the assessment.
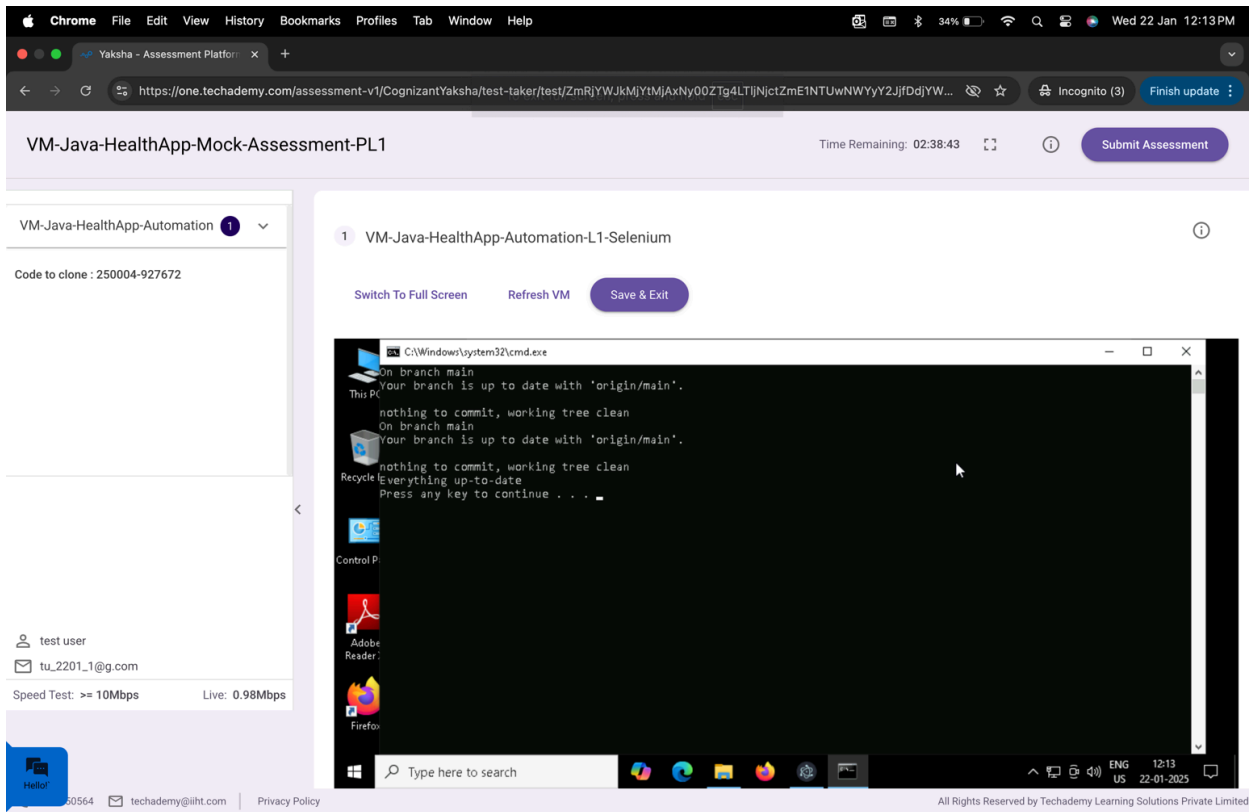
After the successful submission of the assessment, you will get a confirmation message displayed on your screen.

==

All the Best