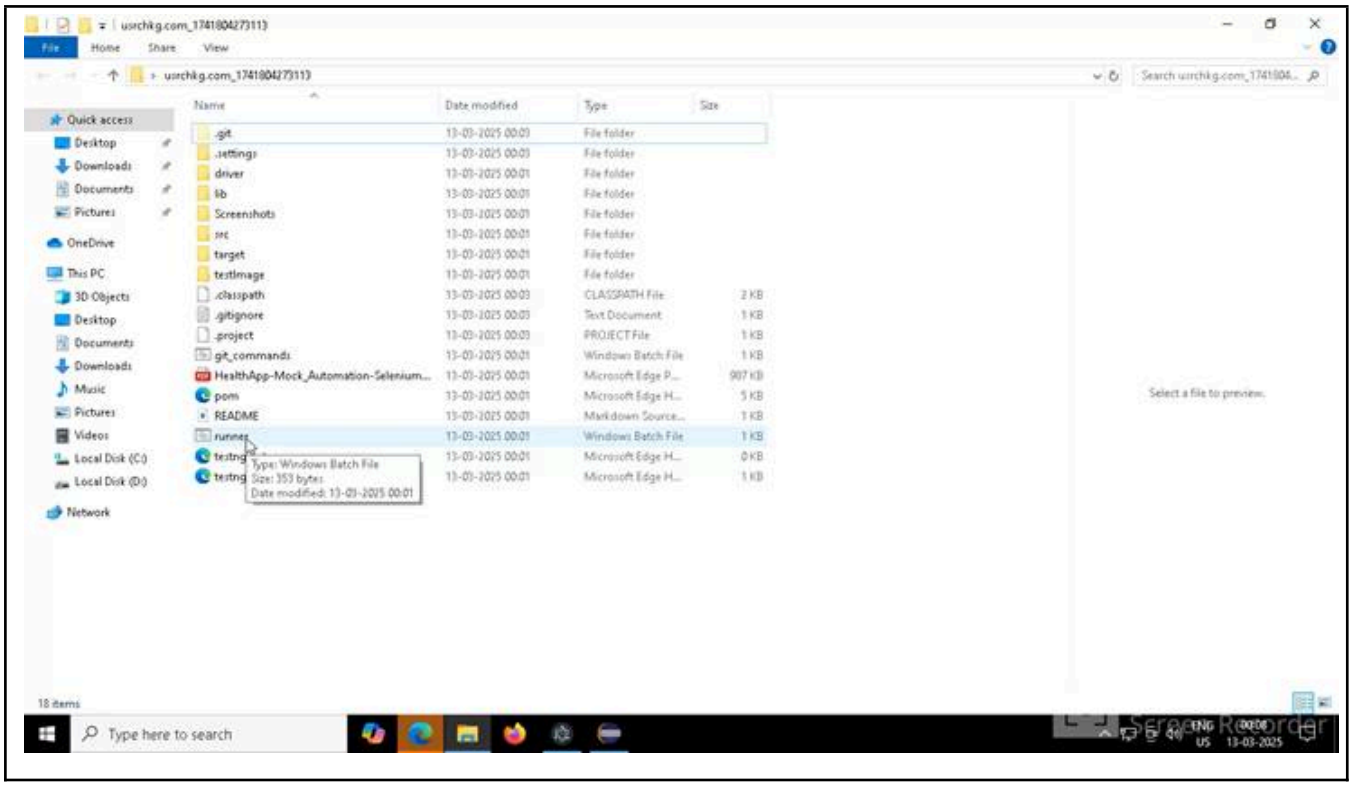
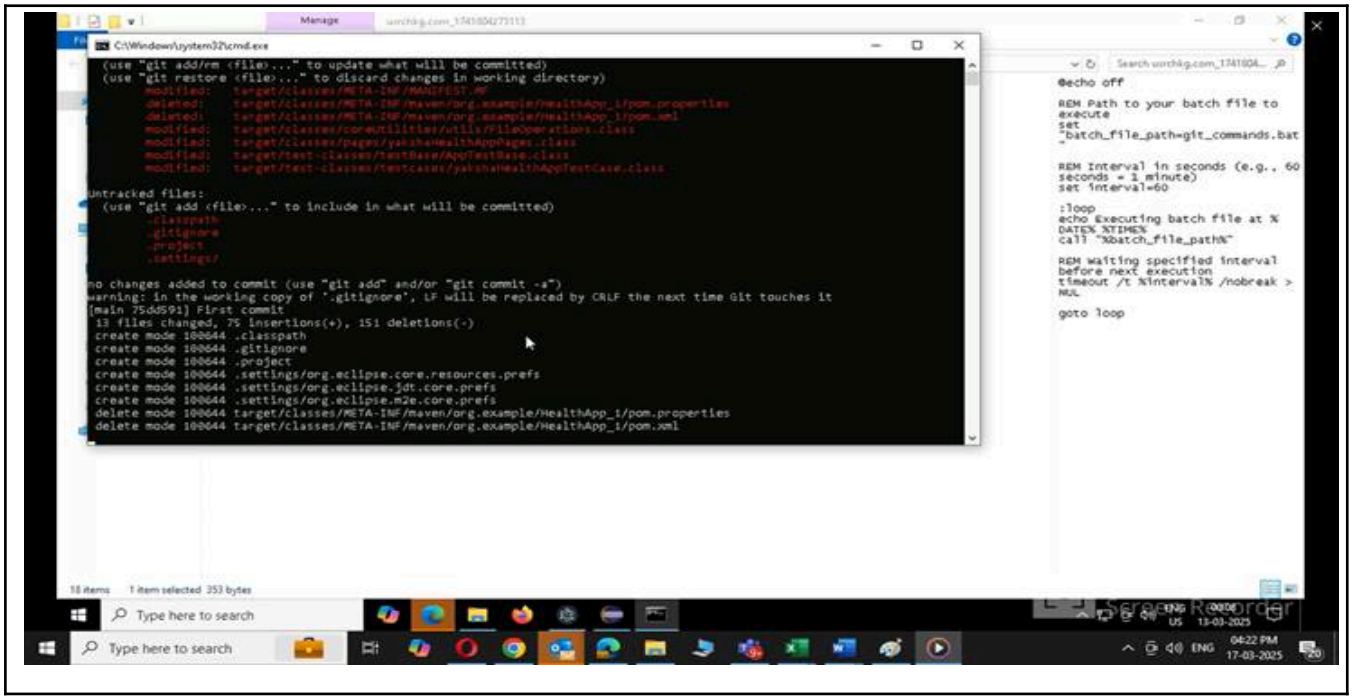
REST-ASSURED
(API - AUTOMATION PROJECT – PL2_8)

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.

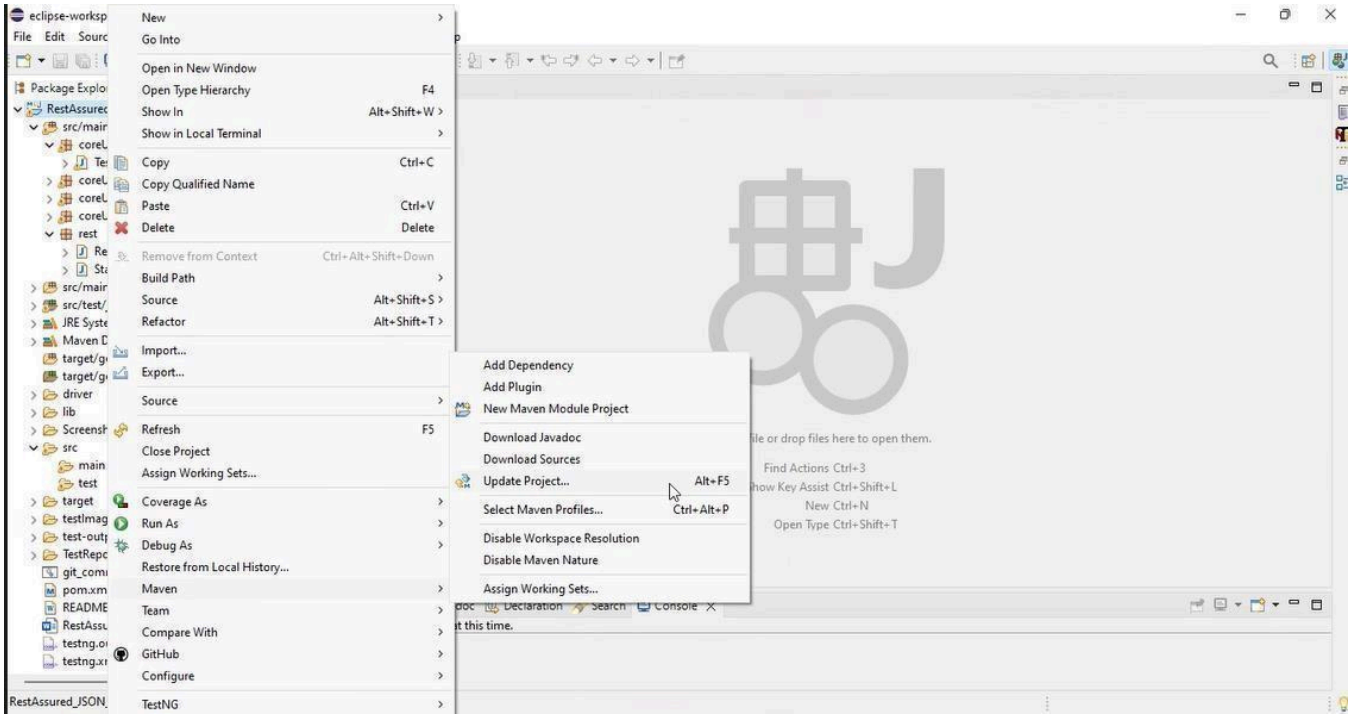


This will launch a command terminal for you where it will keep on pushing your updated code to GIT on regular intervals. Keep that command terminal open at backend and you can continue working on your project.

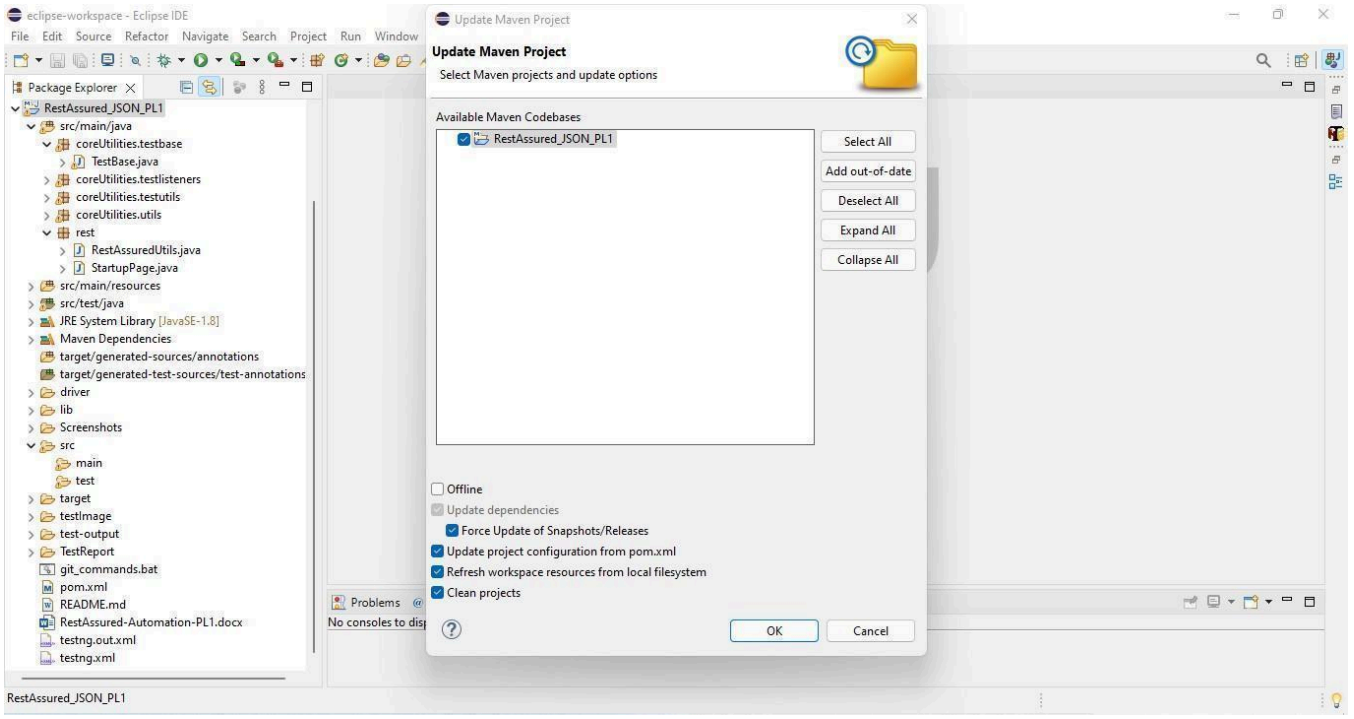


As soon as you import project in eclipse, update the project using maven update option as below. This is to resolve issue if any maven dependency not downloaded properly:

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



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 “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/rest	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.properties	1. Data present to be used in creating a new Post.
	TestData.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	Perform login with Cookie Authentication using method: **GetHolidayData (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://yakshahrm.makemylabs.in/orangehrm-5.7 /web/index.php/api/v2/leave/holidays ?fromDate=2025-01-01&toDate=2025-12-31. 2. **Prepare Request** • Build request with RestAssured. • Include `cookieValue` (String) as `"orangehrm"` cookie. • Set header `"Content-Type": "application/json"`. • Optionally attach `body` (Map<String, String>) if provided. 3. **Send GET Request** • Request URL: `BASE_URL + endpoint`. • Capture `response` (Response). 4. **Extract HTTP Metadata** • `statusCode` (int) → HTTP status code. • `status` (String) → full status line. 5. **Initialize Lists** • `ids`, `names`, `dates`, `recurrings`, `lengths`, `lengthNames` → List<Object> for each field. 6. **Parse JSON Response** • Use `JsonPath` to get `data` → List<Map<String, Object>>. • If `data` is not null: o Loop through each holiday and extract fields: `id`, `name`, `date`,	1. Method should return a fully initialized CustomResponse object with: a. statusCode = 200 b. status = OK (or valid status line like "HTTP/1.0 200 OK") c. The complete raw Response object d. List<Object> ids e. List<Object> names f. List<Object> dates g. List<Object> recurrings h. List<Object> lengths i. List<Object> lengthNames 2. An `Data` object with non-null and non-empty lists for: a. `ids` b. `names` c. `dates`

		`recurring`, `length`, `lengthName`. o Add each to its corresponding list. 7. **Handle Null Data** • Log a warning if `data` is null along with `statusCode`. 8. **Wrap in CustomResponse** • Return `CustomResponse` containing: o `response` (Response), `statusCode` (int), `status` (String) o All extracted lists: `ids`, `names`, `dates`, `recurrings`, `lengths`, `lengthNames`.	
2	Retrieve leave data using session cookie authentication through method: **GetLeaveData(String endpoint, String cookieValue, Map<String, String> body)**	1. Construct the complete URL by concatenating the BASE_URL and the provided endpoint: a. https://yakshahrm.makemylabs.in/orangehrm-5.7 /web/index.php/api/v2/leave/holidays ?fromDate=2025-01-01&toDate=2025-12-31. 2. **Prepare Request**: • Add session cookie (`orangehrm`) • Set header `Content-Type: application/json` • Include optional `body` if provided 3. **Send GET Request** and capture the response in JsonPath jsonPath. 4. **Capture Response Details**: • `int statusCode` → HTTP status code • `String status` → status line 5. **Parse JSON Data**: • `List<Map<String, Object>> data` → extract from response • Initialize lists: o `List<Object> ids` o `List<Object> names` o `List<Object> dates` o `List<Object> recurrings`	1. The method should return a CustomResponse object with: a. statusCode = 200 b. status equal to a valid HTTP status line like "HTTP/1.0 200 OK" c. The complete raw Response object d. List<Object> ids e. List<Object> names f. List<Object> dates g. List<Object> recurrings h. List<Object> lengths i. List<Object> lengthNames 2. An `Data` object with non-null and non-empty lists for: a. `ids`

		○ List<Object> lengths ○ List<Object> lengthNames 6. **Populate Lists**: For each item in data, add values to ids, names, dates, recurrings, lengths, lengthNames. 7. Log a warning if data is null along with statusCode. 8. **Wrap in CustomResponse**: Return CustomResponse containing response, statusCode, status, ids, names, dates, recurrings, lengths, lengthNames.	b. names c. dates
3	Retrieve Emp count using session-based authentication through the method: **GetEmpCount(String endpoint, String cookieValue, Map<String, String> body)**	1. Construct the final URL by combining the BASE_URL and the provided endpoint: a. https://yakshahrm.makemylabs.in/orangehrm-5.7 /web/index.php/api/v2/pim/employees/count. 2. **Prepare Request**: • Add session cookie (orangehrm) • Set header Content-Type: application/json • Include optional body if provided 3. **Send GET Request**: Execute .get() and capture Response. 4. **Capture Response Details**: • int statusCode → HTTP status code • String status → status line • Debug print the raw response 5. **Convert Response to JSONPath**: Transform the response into a JsonPath object for easy data extraction. 6. **Parse JSON Data**: • Map<String, Object> data → extract data object • int count → extract "count" from data 7. **Handle Missing Key**: Log if count key is missing.	1. The method should return a CustomResponse object containing: • ids • statusCode = 200 • A valid status line such as "HTTP/1.0 200 OK" • The complete Response object from the API call • int count 2. Emlpoyee count should not be null

		8. **Wrap in CustomResponse**: Return `CustomResponse` containing `response`, `statusCode`, `status`, `count`.	
4	Retrieve employee leaves information using session-based authentication through method: **GetLeaveType(String endpoint, String cookieValue, Map<String, String> body)**	1. Construct the final URL by combining the BASE_URL and the provided endpoint: o https://yakshahrm.makemylabs.in/orangehrm-5.7 /web/index.php/api/v2/leave/leave-types?limit=0 2. **Prepare the GET request** using RestAssured: • Include the `orangehrm` cookie for authentication. • Set the header `Content-Type` to `application/json`. • Attach the `body` only if it is not null. 3. **Send the GET request** to `BASE_URL + endpoint` and capture the response. 4. **Convert the response to JsonPath** for easy field extraction. 5. **Parse JSON Data**: • `List<Map<String, Object>> data` → extract from response • Initialize lists: o `List<Object> ids` o `List<Object> names` o `List<Object> situationals` o `List<Object> Deletes` 6. **Extract fields** from the `data` array in the JSON: • `ids` → from `data.id` • `names` → from `data.name` • `situationals` → from `data.situational` • `Deletes` → from `data.deleted` 7. **Check for null**: If `data` is null, log a warning that the `data` field is empty with status code.	• Method must return a valid CustomResponse object containing: • statusCode = 200 • status like "HTTP/1.0 200 OK" o The full raw Response object • An `Data` object with non-null and non-empty lists for: o List<Object> ids o List<Object> names o List<Object> situationals o List<Object> Deletes

		8. **Return a `CustomResponse` object** containing: • The raw `response` • `statusCode` and `status` • Lists: `ids`, `names`, `situationals`, `Deletes`	
5	Retrieves a usage report by sending a GET request and extracts header-related information.: **GetUsageReport (String endpoint, String cookieValue, Map<String, String> body)**	1. Construct the final URL by combining the BASE_URL and the provided endpoint: /web/index.php/api/v2/leave/reports?name=my_leave_entitlements_and_usage. 2. **Prepare the GET request** with RestAssured: • Include the `orangehrm` cookie for authentication. • Set the `Content-Type` header to `application/json`. • Attach the `body` only if it is not null. 3. **Send the GET request** to `BASE_URL + endpoint` and capture the response. 4. **Convert the response to JsonPath** for structured extraction. 5. **Parse JSON Data**: • `List<Map<String, Object>> headers` → extract from response • Initialize lists: o `List<Object> names` o `List<Object> props` o `List<Object> sizes` o `List<Object> pins` o `List<Object> cellProperties` 6. **Extract fields from `data.headers` array**: • `names` → from `header.name` • `props` → from `header.prop` • `sizes` → from `header.size` • `pins` → from `header.pin` • `cellProperties` → from `header.cellProperties` (keep as `Map` or `null`) 7. **Handle missing or unexpected data**:	• The method must return a `CustomResponse` containing: • `statusCode = 200` • A valid `status` line like `"HTTP/1.0 200 OK"` • An `Data` object with non-null and non-empty lists for: ▪ `int subUnitId` ▪ `String subUnitName` ▪ `int subUnitCount` • The full raw `Response` object • List<Object> props • List<Object> names • List<Object> sizes • List<Object> pins • List<Object> cellProperties

		• If `data.headers` is null, log a warning along with status code. • If `cellProperties` is not a `Map` or null, log a warning and set it to null. 8. **Return a `CustomResponse`** containing: • The raw `response` • `statusCode` and `status` • Lists: `props`, `names`, `sizes`, `pins`, `cellProperties`	

6	Send aGET request to retrieve vacancy details and constructs a CustomResponse object containing parsed data from the response. **GetVacancies (String endpoint, String cookieValue, Map<String, Object> queryParams)**	1. Construct the complete URL by appending the endpoint to the base URL: `/web/index.php/api/v2/recruitment/vacancies?limit=50&offset=0&sortField=vacancy.name&sortOrder=ASC&model=detailed.` 2. **Build the GET request** using RestAssured: • Add the authentication cookie `"orangehrm"`. • Set `Content-Type` to `"application/json"`. • Attach `queryParams` if not null. 3. **Send the GET request** to `BASE_URL + endpoint` and extract the `response`. 4. **Convert the response to JSONPath** for structured parsing. 5. **Extract response details**: • `statusCode` → HTTP status code. • `statusLine` → HTTP response status text. 6. **Parse JSON Data**: • `List<Object> dataList` → extract from response 7. **Extract vacancy fields** from the JSON '`data`' array: • `id` → Vacancy ID • `name` → Vacancy name • `description` → Job description • `numOfPositions` → Number of available positions • `statusList` → Vacancy status (active/inactive) • `isPublished` → Whether the vacancy is published • `jobTitle` → Related job title 8. **Return the parsed result** in a `CustomResponse` containing: • Raw `response`, • `statusCode`, `statusLine`,	• The method should return a `CustomResponse` object with: • `statusCode = 200` • A valid `status` string (e.g., `"HTTP/1.0 200 OK"`) • List<Object> id • List<Object> name • List<Object> description • List<Object> numOfPositions • List<Object> statusList • List<Object> isPublished • List<Object> jobTitle

		• Lists: `id`, `name`, `description`, `numOfPositions`, `statusList`, `isPublished`, `jobTitle`.	
7	Sends a GET request to retrieve job titles and constructs a CustomResponse object containing the parsed data from the response. **GetJobTitles (String endpoint, String cookieValue, Map<String, Object> queryParams)**	1. Construct the complete URL by appending the endpoint to the base URL:`/web/index.php/api/v2/admin/job-titles?limit=0`. 2. **Get Request** • Use `GET` on the `endpoint` with `orangehrm` as `cookieValue` and optional `queryParams`. • Request header: `"Content-Type: application/json"`. 3. **Capture Response** • `Response response` → holds the raw HTTP response. • `int statusCode` → HTTP status code • `String statusLine` → status line 4. **Parse Response JSON** • Convert `response` into `JsonPath jsonPath` for extraction. 5. **Extract Keys and Store Variables (as lists in `CustomResponse`)** • `List<Object> idsAsObjects` → stores `data.id` from JSON. • `List<Object> titlesAsObjects` → stores `data.title` from JSON. 6. **Return Result** • Wrap `response`, `statusCode`, `statusLine`, `idsAsObjects`, `titlesAsObjects` in `CustomResponse`.	• The method should return a valid `CustomResponse` containing: • `statusCode = 200` • A valid HTTP `status` (e.g., `"HTTP/1.0 200 OK"`) • List<Object>idsAsObjects • List<Object>titlesAsObjects
8	Retrieve Personal details using session-based cookie authentication via method: **GetEmpPersonalDetails (String endpoint, String cookieValue, Map<String, Object> queryParams)**	1. Construct the complete URL by appending the endpoint to the base URL:`"/web/index.php/api/v2/pim/employees/" + empNumber + "/personal-details` 2. **Prepare Request** • Create a `RequestSpecification` using RestAssured.	• The method should return a valid `CustomResponse` containing: • `statusCode = 200` • A valid HTTP `status` (

		• Add a session cookie named "orangehrm" with the provided cookieValue • Set header Content-Type: application/json. • If query parameters are provided, attach them to the request. 3. **Send GET Request** • Call .get(BASE_URL + endpoint) to send the GET request. • Extract the Response object. 4. **Extract Response Metadata** • Get statusCode from response. • Get String status → statusLine from response. 5. **Extract Required Keys into Variables from responseBody:** • Integer empNumber ← data.empNumber or data[0].empNumber • String firstName ← data.firstName or data[0].firstName • String lastName ← data.lastName or data[0].lastName • String nationalityName ← data.nationality.name or data[0].nationality.name 6. **Return Wrapped Response** • Pass all variables (response, statusCode, status, empNumber, firstName, lastName, nationalityName) into CustomResponse.	e.g., "HTTP/1.0 200 OK") • List<Object> empNumber • List<Object> firstName • List<Object> lastName • List<Object> nationalityName
9	Retrieve Employee data using session-based authentication through method: **GetEmpData (String endpoint, String cookieValue, String body)**	1. Compose the full URL by appending the endpoint to the base URL: /web/index.php/api/v2/pim/employees/" + empNumber 2. Initialize a **RequestSpecification** using **RestAssured**:	• The method should return a valid CustomResponse containing: • statusCode = 200 • A valid HTTP status (

		• Add a session cookie named `"orangehrm"` with the provided `cookieValue` • Set the `"Content-Type"` header to `"application/json"` 2. **Prepare Request** • Use RestAssured to create a GET request. • Set `Content-Type: application/json`. • Add `cookie` for authentication. 3. **Send GET Request** • Send request to `BASE_URL + endpoint`. • Extract the `Response` object. 4. **Extract Response Metadata** • Get `statusCode` from response. • Get `statusLine` from response. 5. **Convert Response to JSON** • Parse the response body into `JsonPath` for key extraction. 6. **Initialize Lists** • `List<Integer> empNumbers` • `List<String> firstNames` • `List<String> lastNames` • `List<String> employeeNumbers` 7. **Extract Keys from JSON** • Check if `data` is an array or single object. • If `data` is **array**: o Extract `data.empNumber` → `empNumbers` o Extract `data.firstName` → `firstNames` o Extract `data.lastName` → `lastNames` o Extract `data.employeeId` → `employeeNumbers` • If `data` is **single object**: o Add `data.empNumber` to `empNumbers`	e.g., `"HTTP/1.0 200 OK"`) • List<Object> empNumbers • List<Object> firstNames • List<Object> lastNames • List<Object> employeeNumbers

		○ Add `data.firstName` to `firstNames` ○ Add `data.lastName` to `lastNames` ○ Add `data.employeeId` to `employeeNumbers` 8. **Return Wrapped Response** • Pass `response`, `statusCode`, `statusLine`, `empNumbers`, `firstNames`, `lastNames`, `employeeNumbers` to `CustomResponse`.	
10	Retrieve leave work week using session-based cookie authentication via method: **GetLeaveWorkWeek (String endpoint, String cookieValue)**	1. Construct the complete URL by appending the endpoint to the base URL: `/web/index.php/api/v2/leave/workweek?model=indexed` 2. **Prepare Request** • Use RestAssured to create a GET request. • Add `cookie` for authentication. 3. **Send GET Request** • Call `BASE_URL + endpoint`. • Extract the `Response` object. 4. **Extract Response Metadata** • Get `statusCode` from response. • Get `statusLine` from response. 5. **Convert Response to JSON** • Parse the response body into `JsonPath`. 6. **Extract Keys from JSON** • Extract `data` as a `Map<String, Integer>` where keys represent day numbers and values represent hours in Map<String, Integer> workweekData 7. **Return Wrapped Response** • Pass `response`, `statusCode`, `statusLine`, and `workweekData` to `CustomResponse`.	• The method should return a valid `CustomResponse` object containing: • `statusCode = 200` • A valid `status` string like `"HTTP/1.0 200 OK"` • List<Object> workweekData

11	Post employee using session-based cookie authentication via method: **PostEmployee (String endpoint, String cookieValue, String body)**	1. Construct the complete URL by appending the endpoint to the base URL: /web/index.php/api/v2/pim/employees 2. **Prepare POST Request** • Use RestAssured to create a POST request. • Add `cookie` for authentication. • Set `Content-Type` header as `application/json`. • Include the request `body` containing employee data. • Enable `relaxedHTTPSValidation()` for SSL. 3. **Send POST Request** • Call `BASE_URL + endpoint`. • Extract the `Response` object. 4. **Extract Response Metadata** • Get `statusCode` from response. • Get `statusLine` from response. 5. **Return Wrapped Response** • Wrap `response`, `statusCode`, and `statusLine` into `CustomResponse`.	• The method should return a valid `CustomResponse` object containing: • `statusCode = 200` • A valid `status` string like `"HTTP/1.0 200 OK"` • The full `Response` object • The response body should contain the added UniqueName
12	Put Emp details using session-based cookie authentication via method: **putEmployeeDeatils (String endpoint, String cookieValue, String requestBody)**	1. Construct the complete URL by appending the endpoint to the base /web/index.php/api/v2/pim/employees/" + empNumber + "/personal-details. 2. **Prepare PUT Request** a. Use RestAssured to create a PUT request. b. Add `cookie` for authentication. c. Set `Content-Type` header as `application/json`. d. Include the request body containing employee details using .body. 3. **Send PUT Request** a. Call `BASE_URL + endpoint`. b. Extract the `Response` object. 4. **Extract Response Metadata** a. Get `statusCode` from response.	• The method should return a valid `CustomResponse` object containing: • `statusCode = 200` • A valid `status` string like `"HTTP/1.0 200 OK"` • List<Object> empNumbers • List<Object> firstNames • List<Object> lastNames • List<Object> employeeIds

		b. Get `statusLine` from response. 5. **Parse Response JSON** a. Convert response to `JsonPath`. b. Extract the `"data"` node → Object dataNode. c. Handle case when `data` is a single object (Map). 6. **Initialize Lists for Employee Fields** a. `List<Integer> empNumbers` → `"empNumber"` b. `List<String> firstNames` → `"firstName"` c. `List<String> lastNames` → `"lastName"` d. `List<String> employeeIds` → `"employeeId"` 7. **Populate Lists Safely** a. Check for `null` or empty values before adding to lists. 8. **Return Wrapped Response** a. Wrap `response`, `statusCode`, `statusLine`, and all the lists: `empNumbers, firstNames, lastNames, employeeIds` in `CustomResponse`.	
13	Post job-categories using session-based cookie authentication via method: **PostJobCategoriesTest(String endpoint, String cookieValue, String requestBody)**	1. Construct the complete URL by appending the endpoint to the base URL: /web/index.php/api/v2/pim/reports/defined 2. **Prepare POST Request** • Use RestAssured to create a POST request. • Add the `cookie` for authentication. • Set `Content-Type` header as `application/json`. • Include the `requestBody` containing job category details. 3. **Send POST Request** • Call `BASE_URL + endpoint`. • Extract the `Response` object.	• The method should return a valid `CustomResponse` object containing: • `statusCode = 200` • A valid `status` string like `"HTTP/1.0 200 OK"` • List<Object> empStatusIdList • List<Object> empStatusNameList

		4. **Parse Response JSON** • Convert the response into `JsonPath`. • Extract the `"data"` node → Object dataObj. • Handle the case when `data` is a single object (`Map`). 5. **Initialize Lists** • `List<Integer> empStatusIdList` → to store `"id"` • `List<String> empStatusNameList` → to store `"name"` 6. **Populate Lists** • If `data` is a `Map`, extract `"id"` and `"name"` and add them to respective lists. 7. **Extract Response Metadata** • Get `statusCode` from the response. • Get `statusLine` from the response. 8. **Wrap in CustomResponse** • Pass `response`, `statusCode`, `statusLine`, `empStatusIdList`, `empStatusNameList`, and `null` to `CustomResponse`.	
14	Post candidate using session-based authentication through method: **PostCandidate (String endpoint, String cookieValue, String requestBody)**	1. Compose the full URL by appending the endpoint to the base URL: /web/index.php/api/v2/recruitment/candidates 2. **Prepare POST Request** • Use RestAssured to create a POST request. • Apply `relaxedHTTPSValidation()` (to skip SSL checks). • Add authentication via `cookie`. • Set `Content-Type` header to `application/json`. • Attach the `requestBody` containing candidate details. 3. **Send POST Request** • Call `BASE_URL + endpoint`. • Extract the `Response` object. 4. **Parse Response JSON**	• The method should return a valid `CustomResponse` containing: • `statusCode = 200` • A valid HTTP `status` (e.g., `"HTTP/1.0 200 OK"`) • List<Object> empStatusIdList • List<Object> empStatusNameList

		• Convert the response to `JsonPath`. • Extract the `"data"` node → Object dataObj. • Handle it if it's a `Map` (single object). 5. **Initialize Lists** • `List<Integer> empStatusIdList` → to store `"id"` • `List<String> empStatusNameList` → to store `"name"` 6. **Populate Lists** • If `data` is a `Map`, extract `"id"` and `"name"` and add them to respective lists. 7. **Extract Response Metadata** • Get `statusCode` from the response. • Get `statusLine` from the response. 8. **Wrap in CustomResponse and return** • Pass `response`, `statusCode`, `statusLine`, `empStatusIdList`, `empStatusNameList`, and `null` to `CustomResponse`.	
15	Delete recruitment candidate using session-based cookie authentication via method: **DeleterecruitmentCad (String endpoint, String cookieValue, String requestBody)**	1. Construct the complete URL by appending the endpoint to the base /web/index.php/api/v2/recruitment/candidates 2. **Prepare DELETE Request** • Use RestAssured to create a DELETE request. • Apply `relaxedHTTPSValidation()` to skip SSL checks. • Add authentication via `cookie`. • Set `Content-Type` header to `application/json`. • Attach `requestBody` containing IDs for deletion. 3. **Send DELETE Request** • Call `BASE_URL + endpoint`. • Extract the `Response` object. 4. **Parse Response JSON** • Convert the response to `JsonPath`.	• The method should return a valid `CustomResponse` object containing: • `statusCode = 200` • A valid `status` string like `"HTTP/1.0 200 OK"` • List<Object> empStatusIdList • List<Object> empStatusNameList

		• Extract `"data"` array which contains IDs of deleted records. 5. **Initialize Lists** • `List<Integer> empStatusIdList` → store deleted IDs. • `List<String> empStatusNameList` → empty list, since names aren't returned in DELETE. 6. **Extract Response Metadata** • Get `statusCode` from the response. • Get `statusLine` from the response. 7. **Wrap in CustomResponse** • Pass `response`, `statusCode`, `statusLine`, `empStatusIdList`, `empStatusNameList`, and `null` to `CustomResponse`.	

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

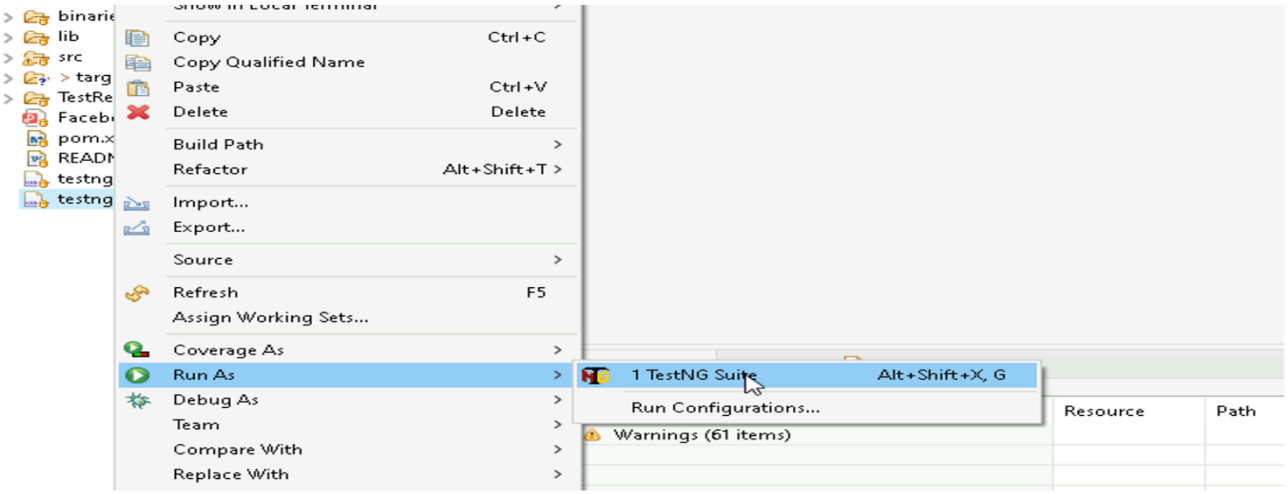
1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

EXECUTION STEPS TO FOLLOW

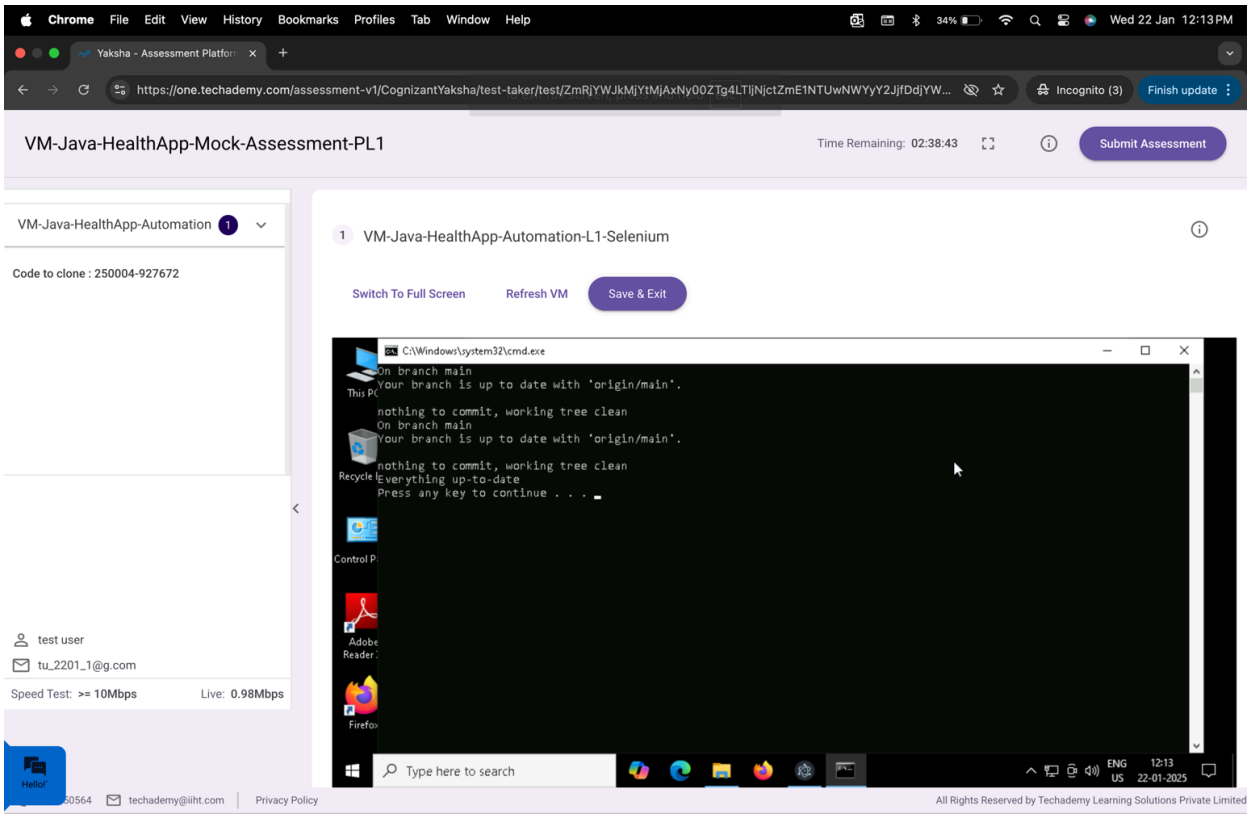
---

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.**



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