

R

EST

A

SSURED

API

A

UTOMATION

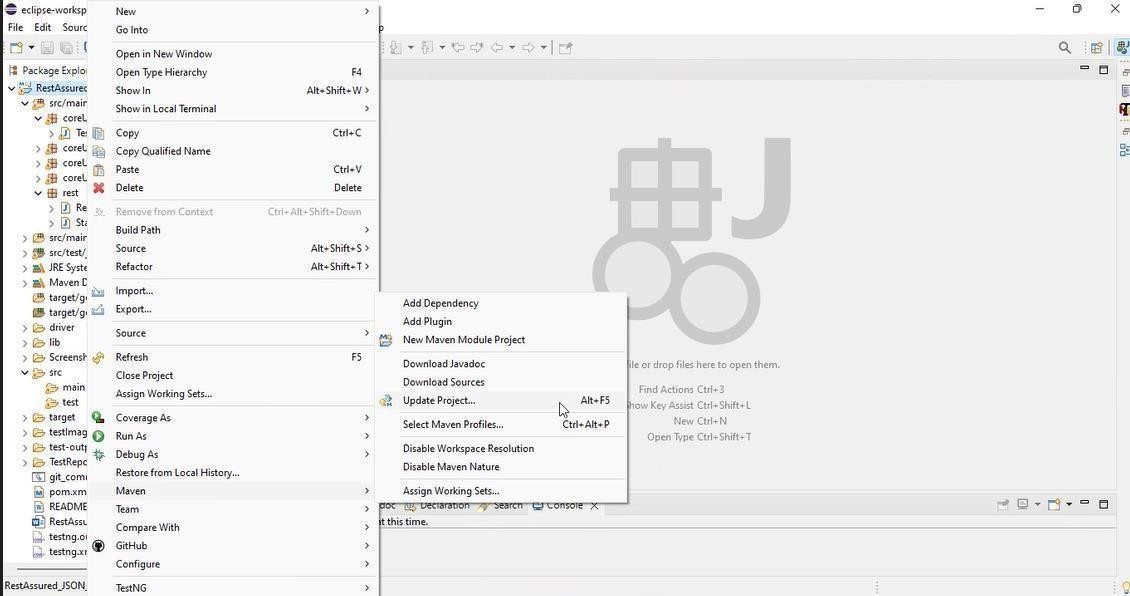
P

ROJECT

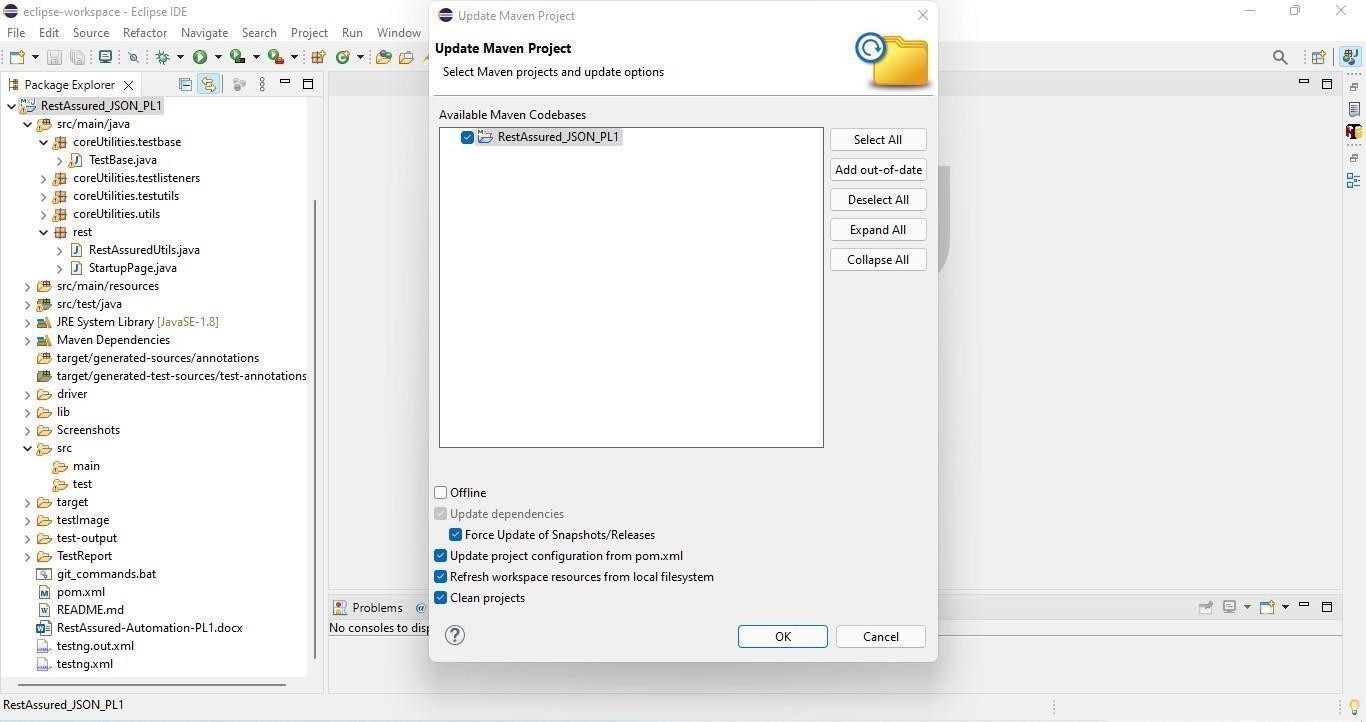
Pre-requisite:

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:

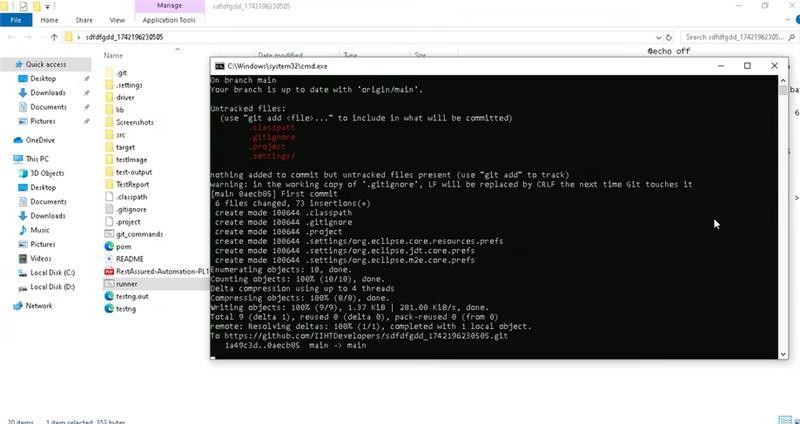
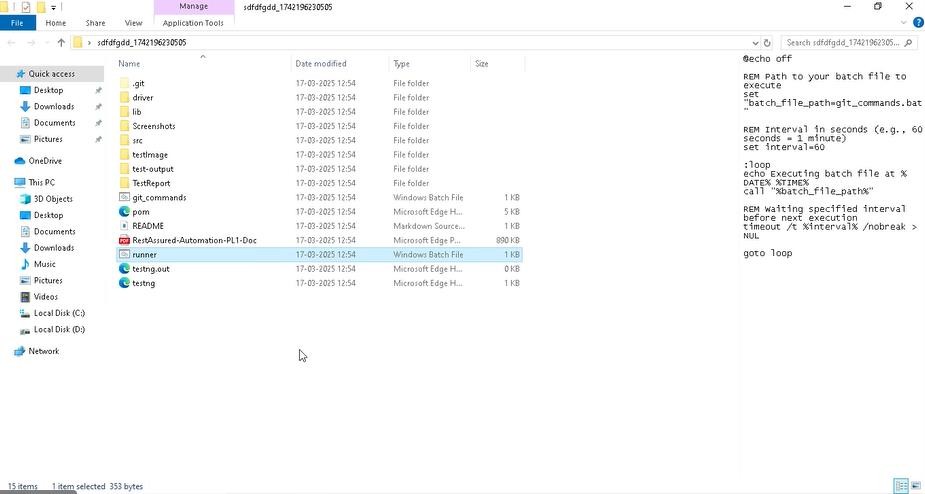
1. Right click on project : Go to “Maven” : Select “Update Project”



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



\*\* Before starting any implementation. Please execute this “runner” file present in the cloned folder. This will keep pushing the changes at regular intervals.



Template Code Structure:

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Package** | **Class/File** | **Description** | |
| /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  variable/object you need to share data/status between different methods. |
|  |  | 4. | Do not modify the signature of methods declared here. |
|  |  | 5. | You can create additional supportive common methods in  CommonEvents class. |
| /src/main/java/rest | AuthUtil.java | 1. | Class already defined to read and return bearer token from config.properties file. |
| /src/main/java/rest | CustomResponse | We have already created custom response object class. You need to use this and use it as return type from ApiUtil.java class methods. | |
| /src/main/resources/ | Config.xlsx | Data present to be used in Implementing functions. | |
| /src/main/java/coreUtilities/utils | CommonEvents.java | 1. Contains all common activities. 2. Certain templated common methods | |
|  |  | declared here.   1. You implement them as per your need. 2. You can add any additional method for common   activity here | |
|  | Testng.xml | Execution needs to kick started from TestNG xml | |

PROBLEM STATEMENT

Need to automate the following activities using RestAssured.

Key Activities to implement:

Below activities need to be implemented in the ApiUtil.java file present in src/main/java/rest package.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl**  **No.** | **Summary** | **Action** | **Expected Result** |
| **1** | **Retrieve the Localization Config using the method:**  getLocalizationValid(String endpoint, String cookievalue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/admin/localization ’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * Response * statusCode * statusLine * language(from json) * dateformat(from json)   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * Languages * DtaFormat.   -The complete raw response. |
| **2** | **Retrieve the Active Languages using the method:**  getActiveLanguages(String endpoint, String cookievalue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/admin/i18n/languages?limit=50&offset=0&sortOrder=ASC&activeOnly=true’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * statusCode * status * languageId * languageName * languageCode   5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * LanguageId * LanguageCode * LanguageName   -The complete raw response. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **3** | **Retrieve the Dashboard Shortcuts using the method:**  getDashboardShortcuts(String endpoint, String cookievalue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/dashboard/shortcuts ’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * Response * statusCode * status * leaveAssignLeave * leaveLeaveList * leaveApplyLeave * leaveMyLeave * timeEmployeeTimesheet * timeMyTimesheet   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * leave\_assign\_leave * leave\_leave\_list * leave\_apply\_leave * leave\_my\_leave * time\_employee\_timesheet * time\_my\_timesheet   -The complete raw response. |

|  |  |  |  |
| --- | --- | --- | --- |
| **4** | **Retrieve the Authentication Providers using the method:**    getAuthProviders(String endpoint, String cookievalue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/auth/openid-providers?limit=50&offset=0’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * statusCode, * status, * providerIds, * providerNames * providerUrls, * providerStatuses * clientIds   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * providerIds * providerNames * providerUrls * providerStatuses * clientIds   -The complete raw response. |
| **5** | **Retrieve the Admin OAuth Clients List using the method:**  getAdminOAuthList(String endpoint, String cookieValue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/admin/oauth-clients?limit=50&offset=0 ’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * Response * statusCode * status * clientInfo   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * ids * names * clientAdminIds * redirectUrls * enabledFlags * confidentialFlags   -The complete raw response. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **6** | Retrieve the LDAP Config using the method :  getLDAPConfig(String endpoint, String cookievalue,null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/admin/ldap-config 2. ’ 3. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 4. Trigger a GET call to the specified endpoint. 5. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode(), * StatusLine(), * enable * hostname * port * encryption * ldapImplementation * bindAnonymously * bindUserDN * hasBindUserPassword   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * HostName * Encryption * LdapImplementation   -The complete raw response. |
| **7.** | **Retrieve the Optional Field using the method:** getOptionalField(String endpoint, String cookievalue, null as body | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/pim/optional-field ’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * StatusLine * pimShowDeprecatedFields * showSIN * showSSN * showTaxExemptions   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * PimShowDeprecatedFields * ShowSIN * ShowSSN * ShowTaxExemptions   -The complete raw response. |
| **8.** | **Retrieve the Custom Fields using the method:** getCustomFields(String endpoint, String cookievalue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/pim/custom-fields?limit=50&offset=0 ’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * StatusLine * customFieldsList   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:   * statusCode=200 * status=OK.   -Implementation should be corrects  -Non null and non empty List for:   * CustomFieldsList   -The complete raw response. |
| **9.** | **Retrieve the Reporting Methods using the method:** getReportingMethods(String endpoint, String cookievalue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/pim/reporting-methods?limit=50&offset=0 ’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * statusLine * ids * names   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:  • statusCode=200  • status=OK.  -Implementation should be corrects  -Non null and non empty List for:  •Ids  • Names  -The complete raw response. |
| **10.** | **Retrieve the Termination Reasons using the method:**  getTerminationReasons(String endpoint, String cookievalue, null as body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/pim/termination-reasons?limit=50&offset=0 ’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json”. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * statusLine * ids * names   (extracted from Response object)  5. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:  • statusCode=200  • status=OK.  -Implementation should be corrects  -Non null and non empty List for:  •Ids  • Names  -The complete raw response |
| **11.** | **Update the OpenID Provider using the method:** updateOpenIdProvider(String endpoint, String cookievalue, String body) | 1. Create a URL using the endpoint ‘/web/index.php/api/v2/auth/openid-providers/{id}’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json” and id from the method getOpenIdProviderId 3. Create a Body String as {name:””,url:””,clientId:””, clientSecret:””,. 4. Trigger a PUT call to the specified endpoint. 5. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * statusLine * id * name * URL * Status * clientId * clientSecret * RedirectUris   (extracted from Response object)   1. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:  • statusCode=200  • status=OK.  -Implementation should be corrects  -Non null and non empty List for:  •Id  -The Name,ClientId,ClientSecret should match as passed in the body  -The complete raw response |
| **12.** | **Delete the OpenID Provider using the method:** deleteOpenIdProvider(String endpoint, String cookievalue, int id) | 1. Create a URL using the endpoint "/web/index.php/api/v2/auth/openid-providers/{id} 2. Fetch the Cookie for the session and set the Header Content Type to “application/json” and id from the method getOpenIdProviderId() 3. Create a Body as map using the id fetched 4. Trigger a DELETE call to the specified endpoint. 5. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * statusLine * DeletedId * deletedIds(list)   (extracted from Response object)  Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:  • statusCode=200  • status=OK.  -Implementation should be corrects  -The Id should match as passed in the request parameter  -The complete raw response |
| **13.** | **Create the OAuth Client using the method:**  createOAuthClient(String endpoint, String cookievalue, String body) | 1. Create a URL using the endpoint "/web/index.php/api/v2/admin/oauth-clients”  2. Fetch the Cookie for the session and set the Header Content Type to “application/json”  3. Create a Body {  "name": "random name",  "redirectUri": "random redirectUri",  "enabled": true,  "confidential": false  }  4. Trigger a POST call to the specified endpoint.  5. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  •response,  •StatusCode  •statusLine  •Id  •Name  •ClientId  •RedirectUri  •Enabled  •Confidential  (extracted from Response object)  Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:  • statusCode=200  • status=OK.  -Implementation should be corrects  -Non null and non empty List for:  •Id  -The Name,RedirectUri, should match as passed in the body  -The Redirect URI, Enabled  should match the body  -The complete raw response |
| **14.** | **Update the OAuth Client using the method:** updateOAuthClient(String endpoint, String cookievalue, String body) | 1. Create a URL using the endpoint "/web/index.php/api/v2/admin/oauth-clients/" + id’ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json” and id from the method getSecondOAuthClientId() 3. Create a Body String as {name:””, redirectUri:””,enabled:””, confidential:””,. 4. Trigger a PUT call to the specified endpoint. 5. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * statusLine * id * name * Enabled * Confidential * RedirectUris   (extracted from Response object)   1. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:  • statusCode=200  • status=OK.  -Implementation should be corrects  -The Name,RedirectUri, should match as passed in the body  -The Redirect URI, Enabled  should match the body  -The complete raw response |
| **15.** | **Delete the OAuth Client using the method:** deleteOAuthClient(String endpoint, String cookievalue, String body) | 1. Create a URL using the endpoint /web/index.php/api/v2/admin/oauth-clients‘ 2. Fetch the Cookie for the session and set the Header Content Type to “application/json” and id from the method getLastOAuthClientId() 3. Create a Body String as "{ \"ids\": [" + id + "] }" 4. Trigger a DELETE call to the specified endpoint. 5. Create an object of type CustomResponse and initialize it with values extracted from the Response object:  * response, * StatusCode * statusLine * deletedIds   (extracted from Response object)   1. Return the CustomResponse object from the method | -The method should Returns an object of type CustomResponse containing:  • statusCode=200  • status=OK.  -Implementation should be corrects  -The id passed in request should not be present in the retrieving method  -The complete raw response |

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.



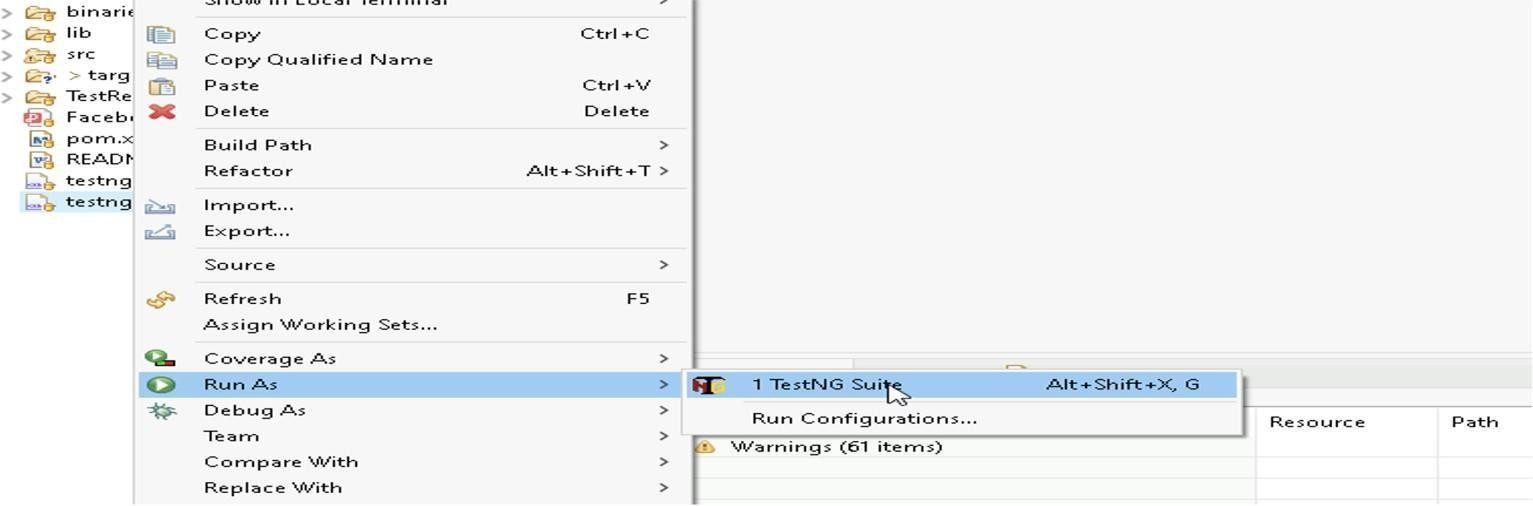
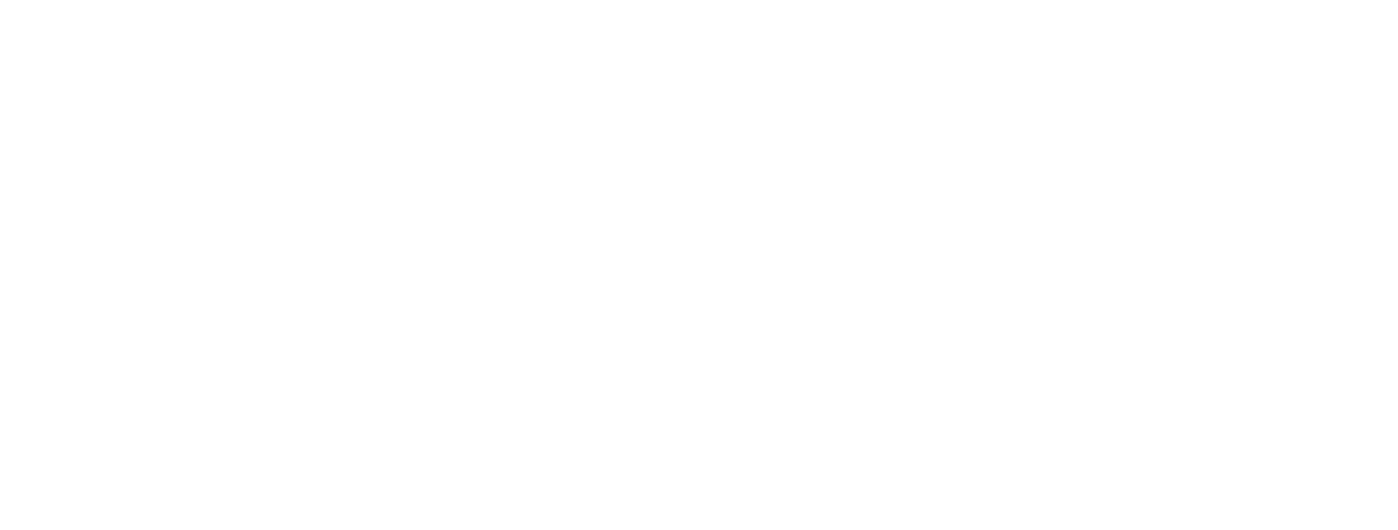
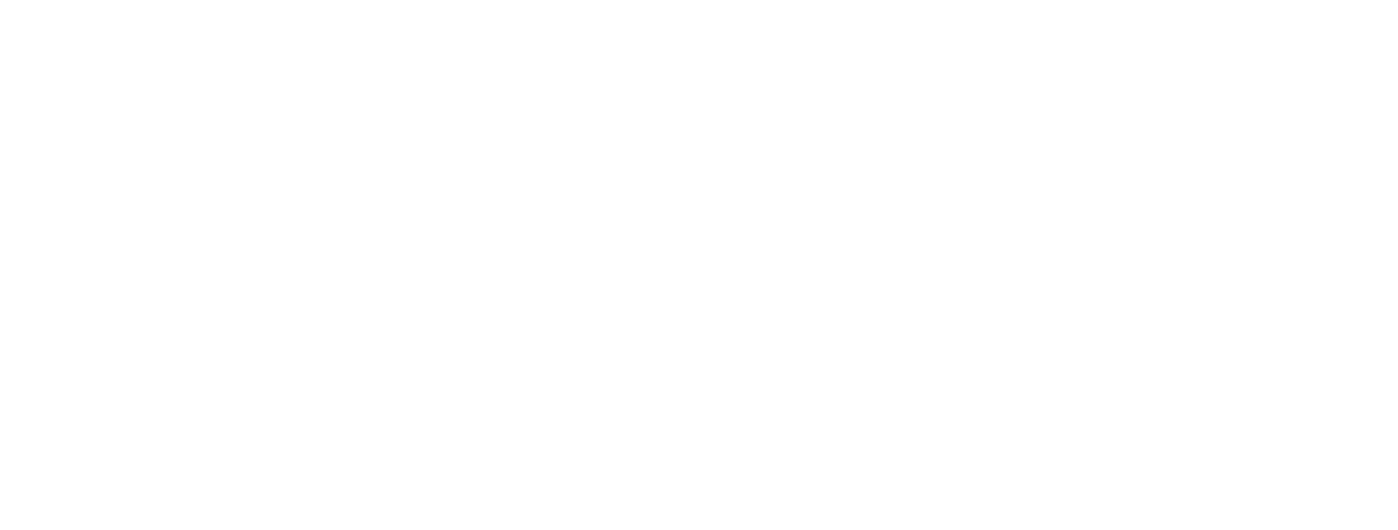
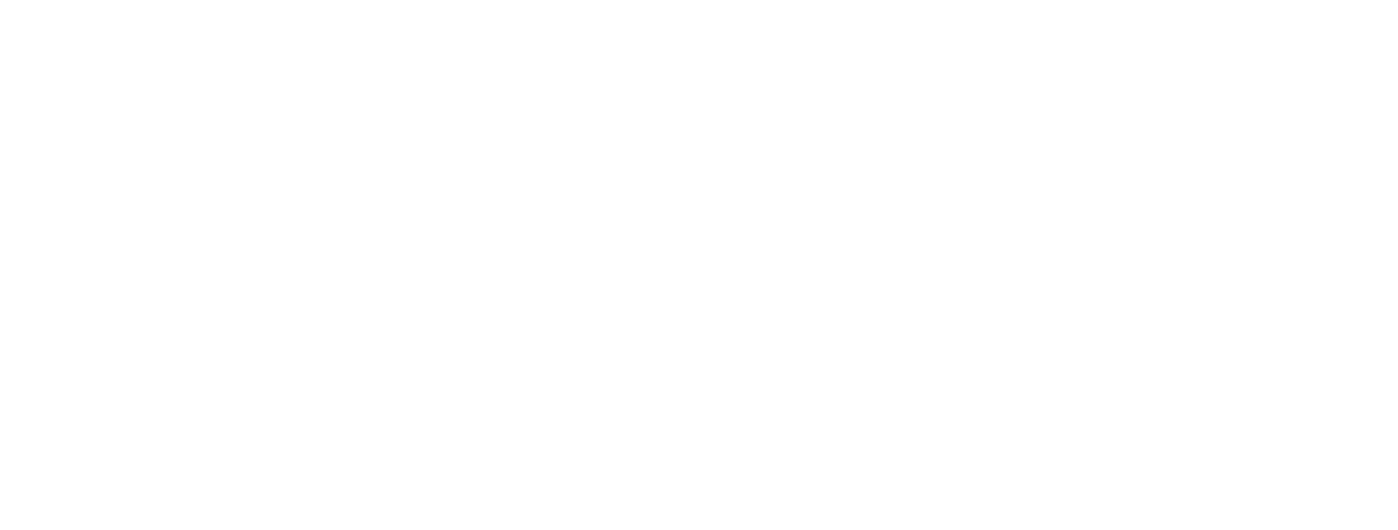
IMPLEMENTATION/FUNCTIONAL REQUIREMENT

**1.1 CODE QUALITY/OPTIMIZATIONS**

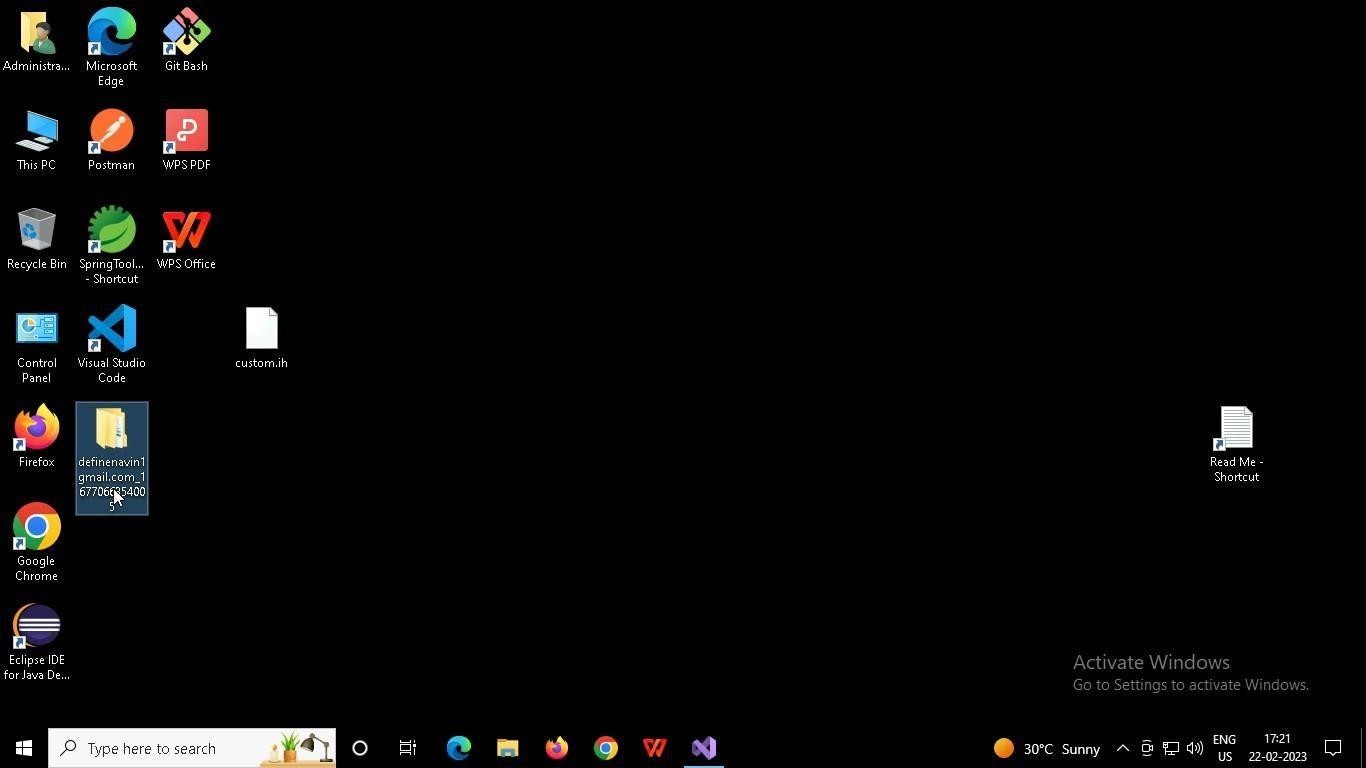
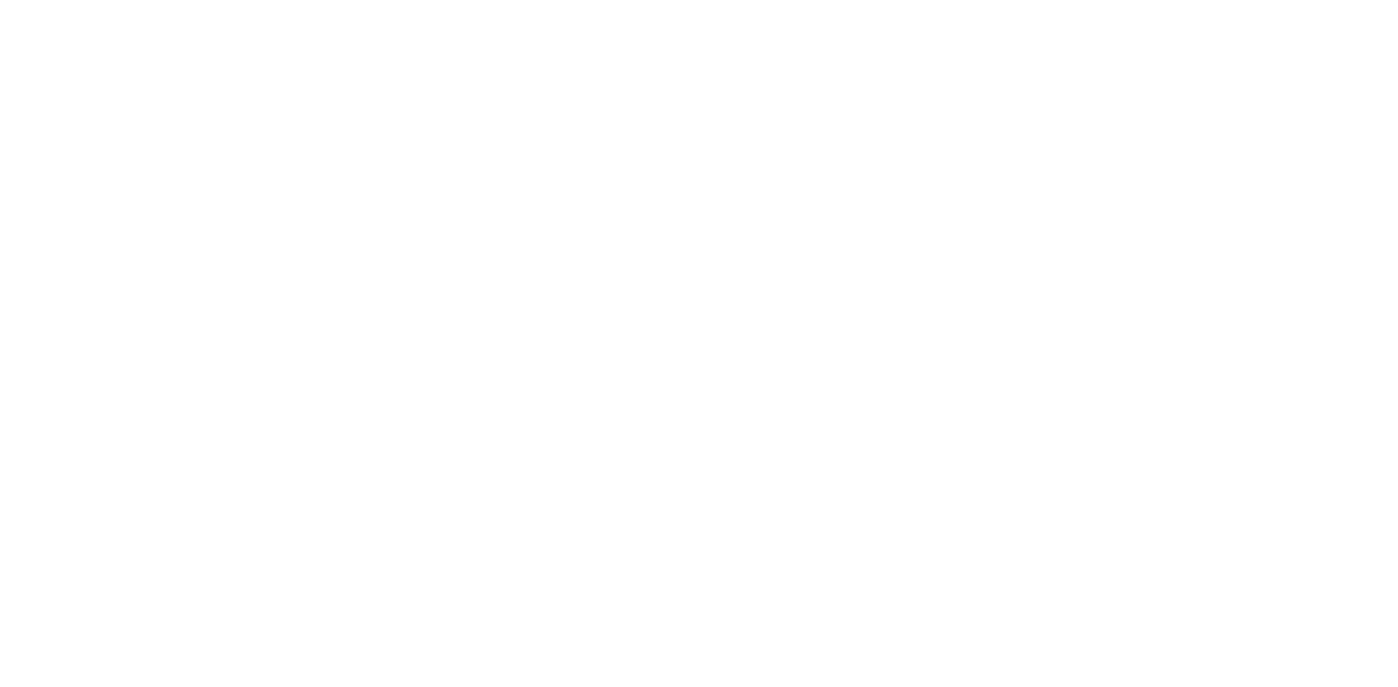
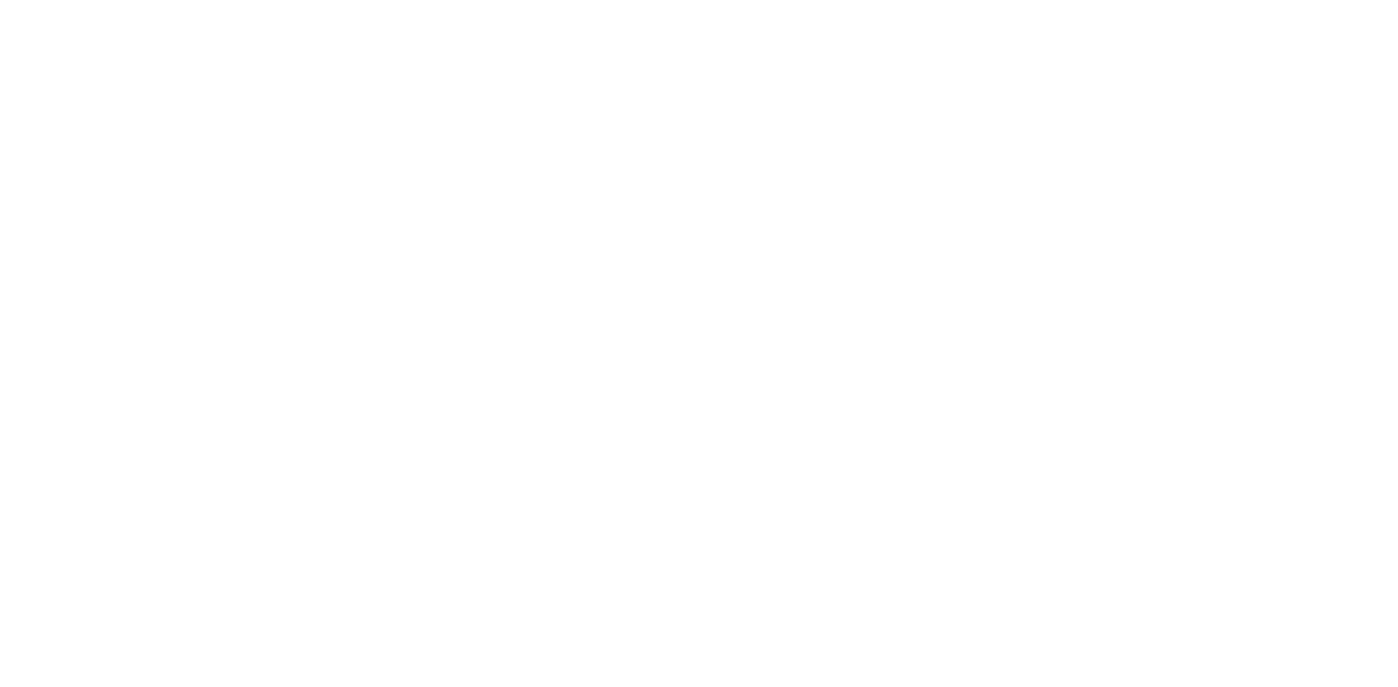
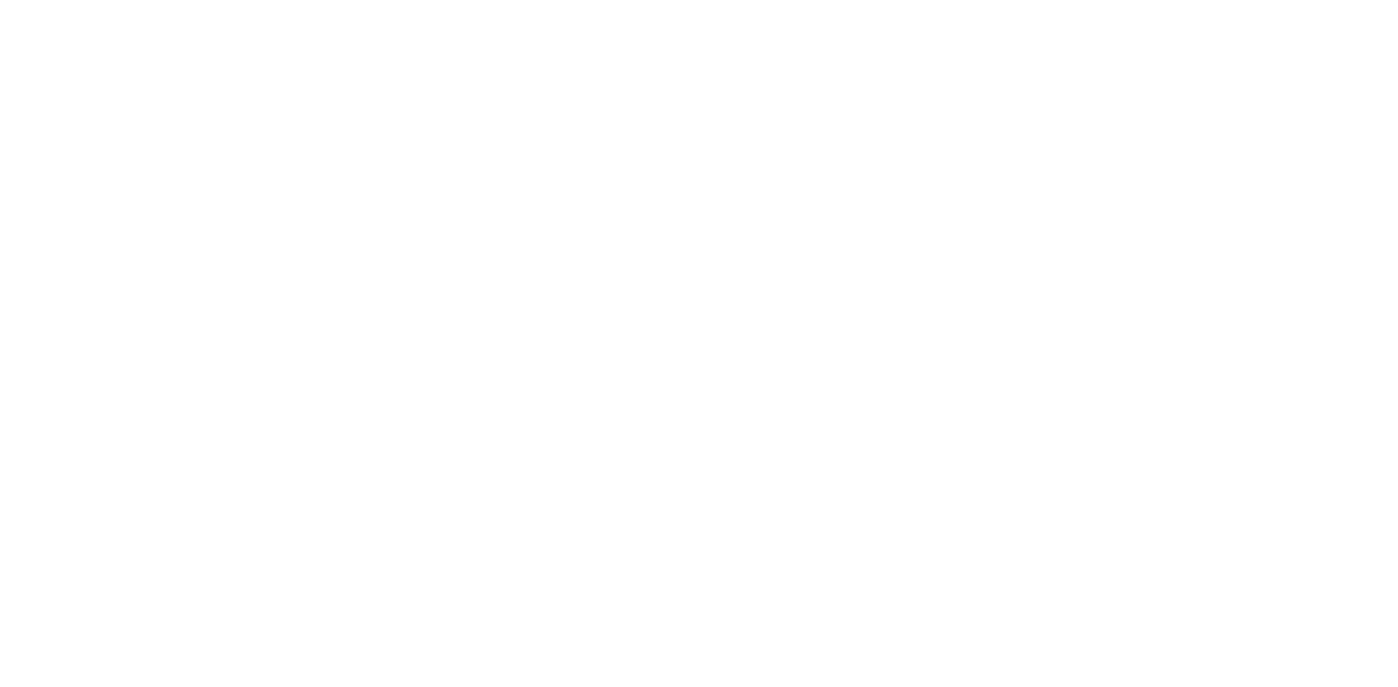
* + - 1. Associates should have written clean code that is readable.
      2. Associates need to follow SOLID programming principles.

# EXECUTION STEPS TO FOLLOW

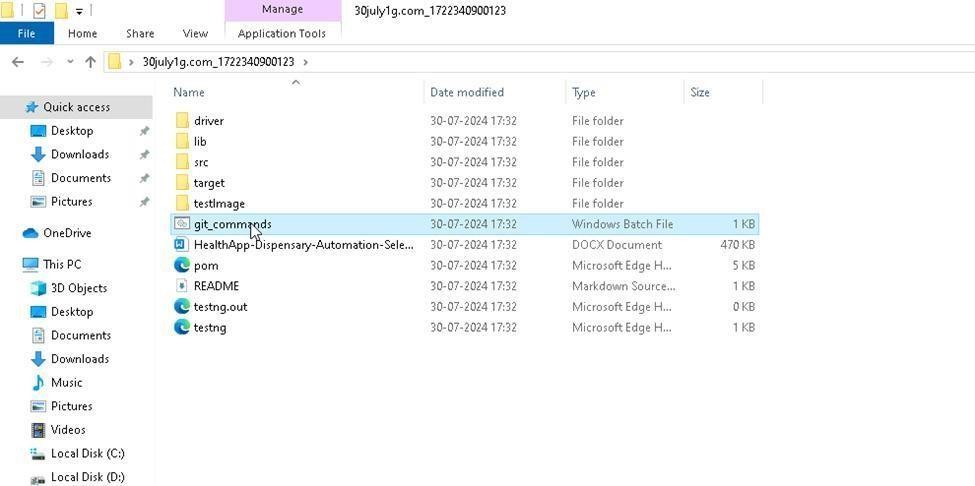
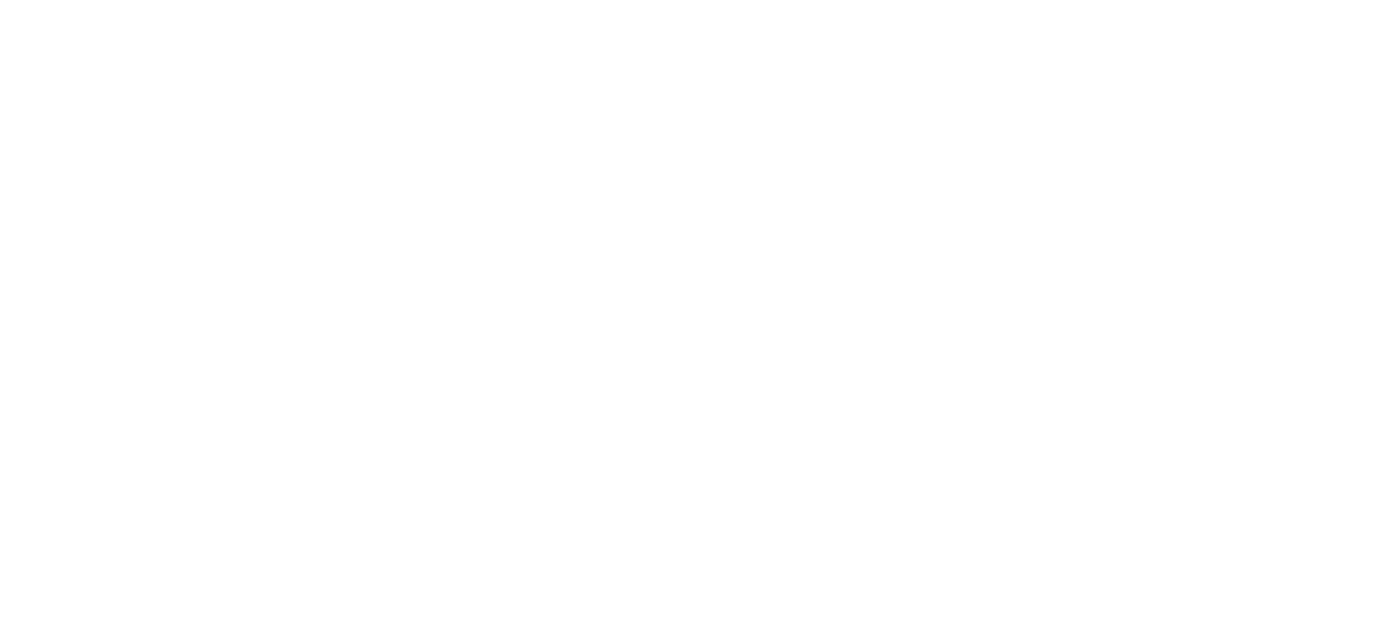
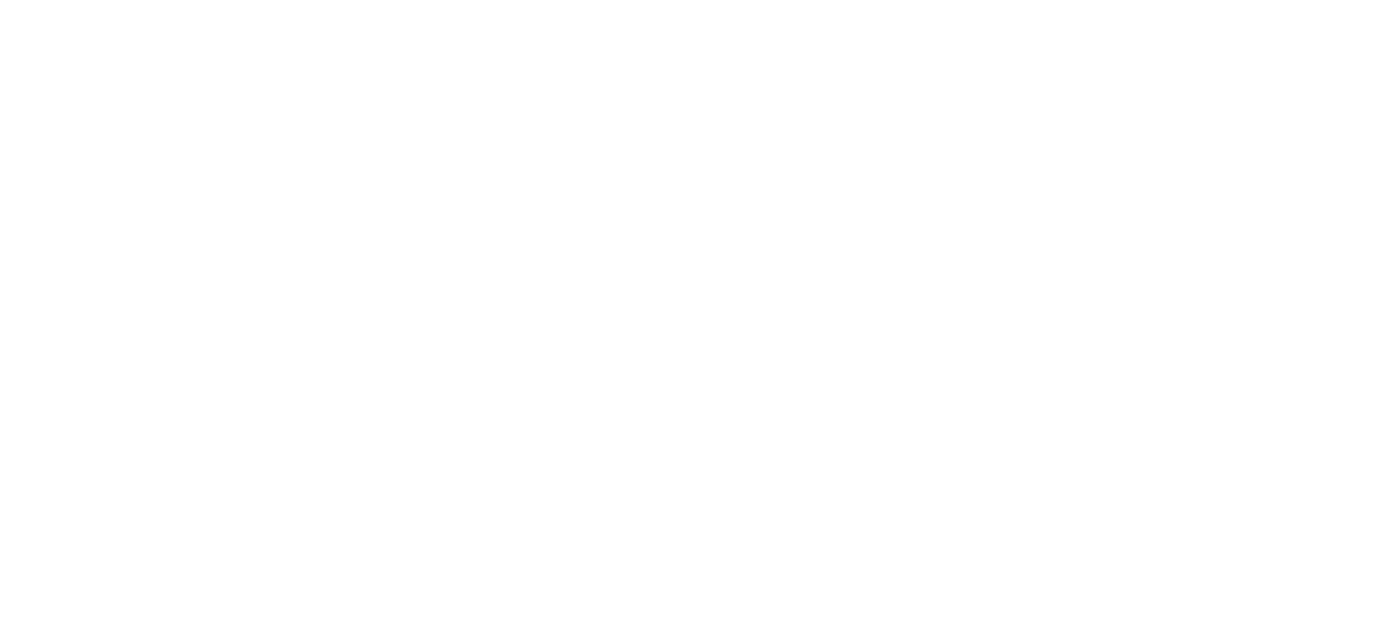
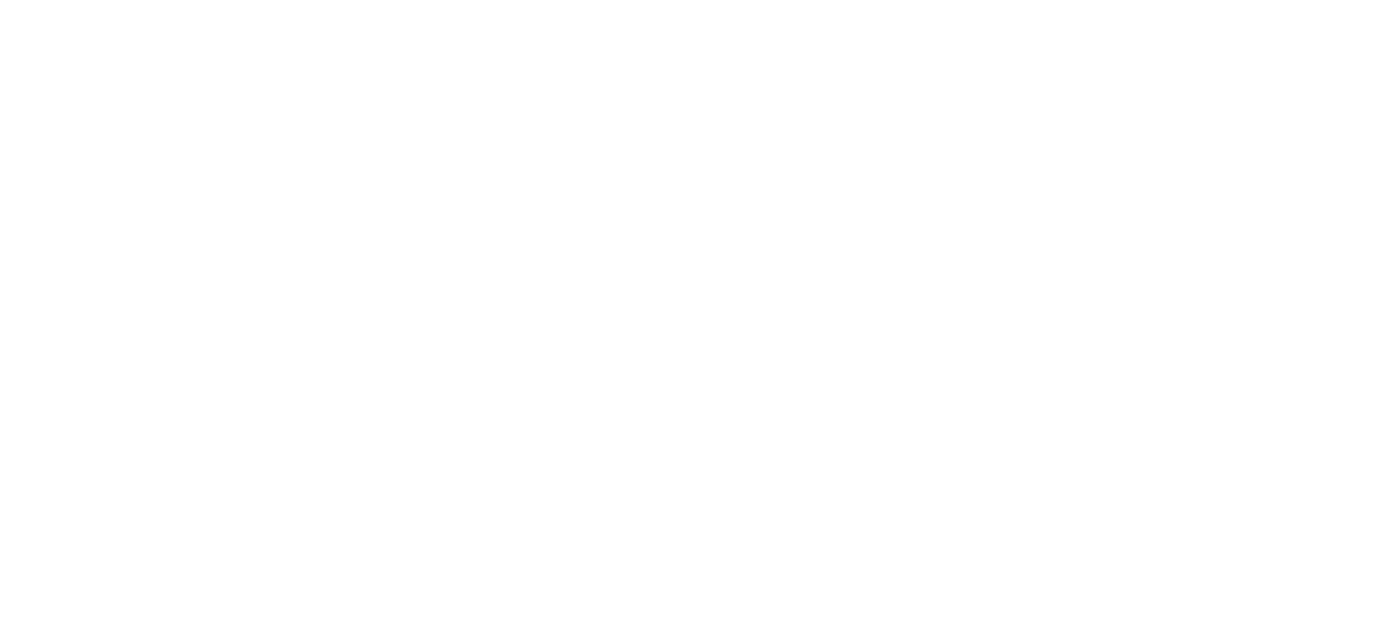
1. **You are mandatorily required to run test cases for applications before final submission. Without which project evaluation will not happen.**
2. **You can launch test cases any time as follows: Right click on testng.xml and run TestNGSuite**



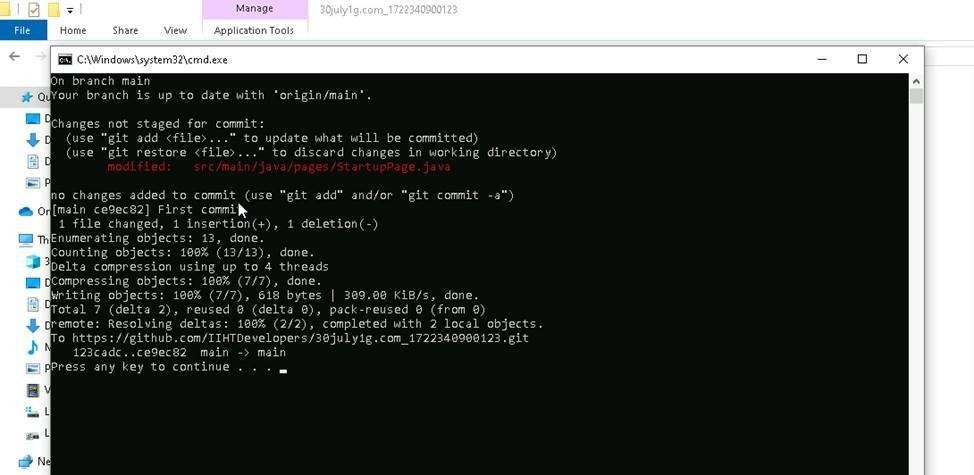
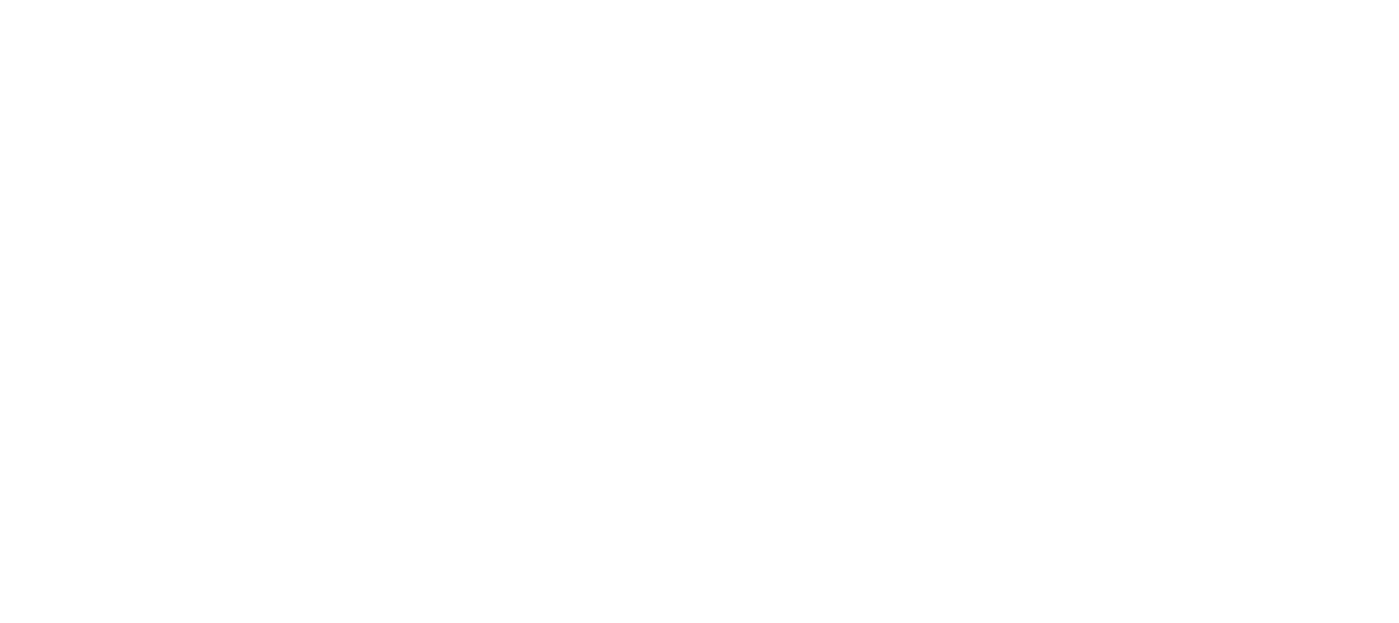
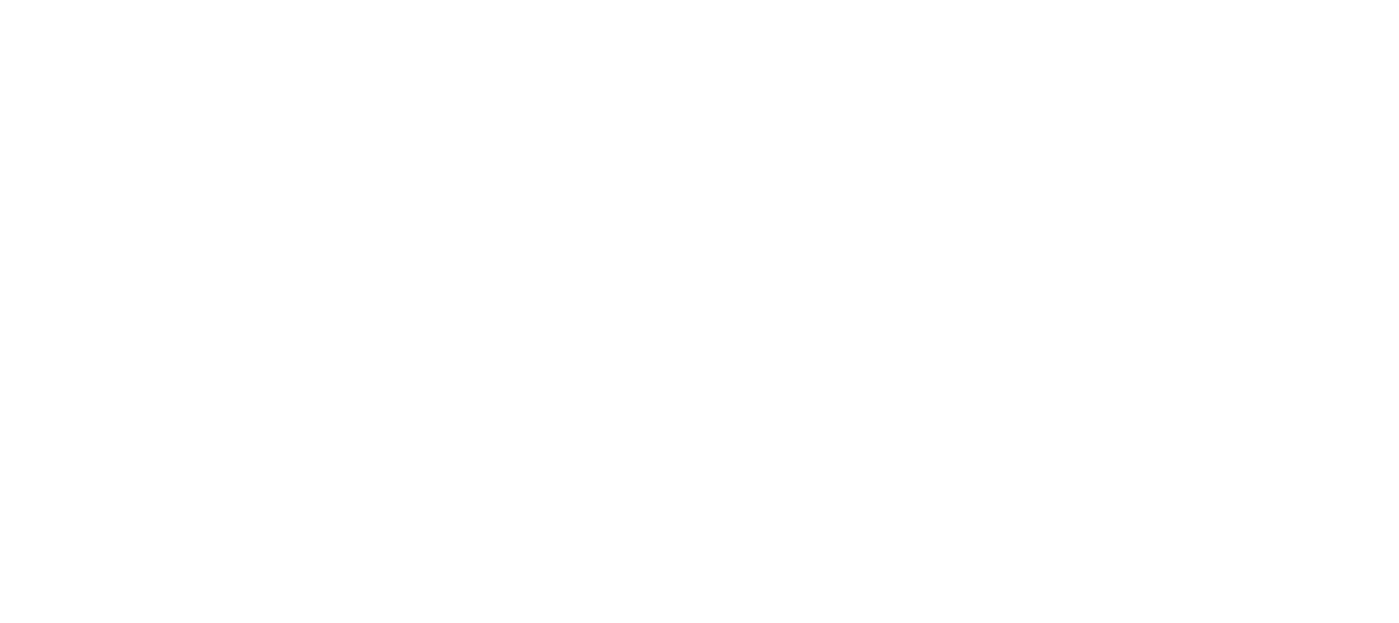
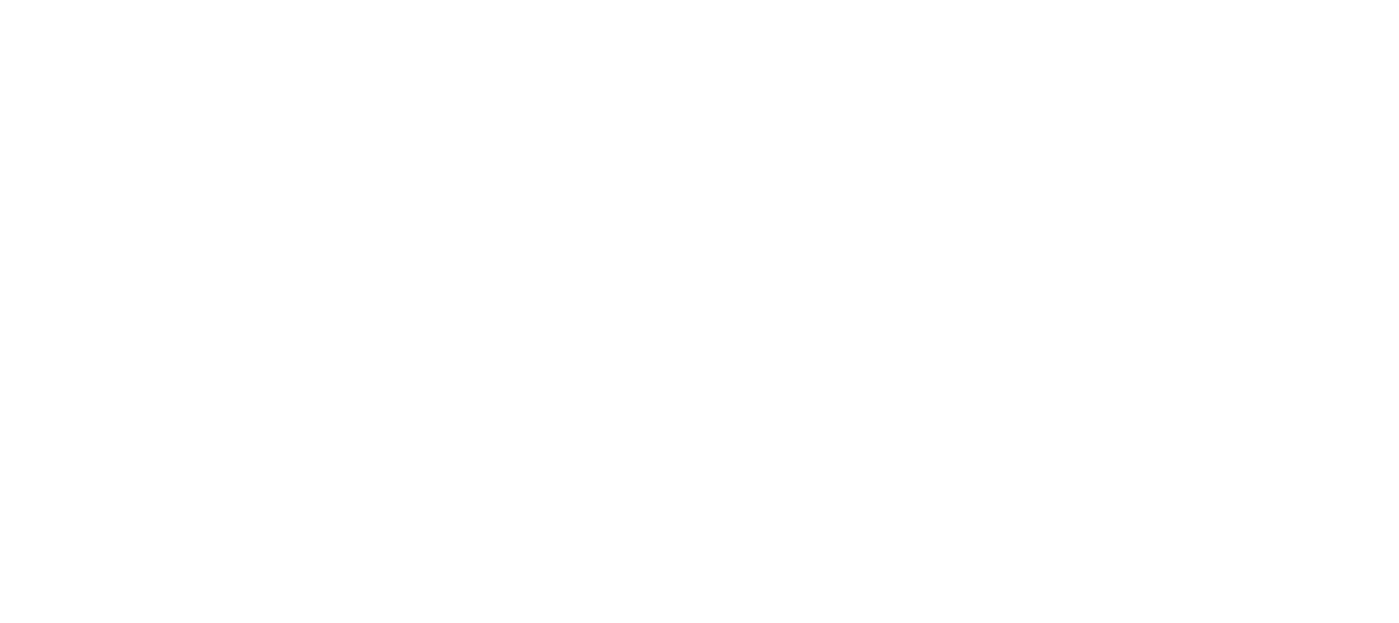
1. **Before final submission, you are also required to push your code to GIT. Following are the steps to follow:**



In your project folder, you will find a batch file named git\_commands



Double-click the batch file to run it. It will run the commands to push your code to GIT.



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

## All the Best