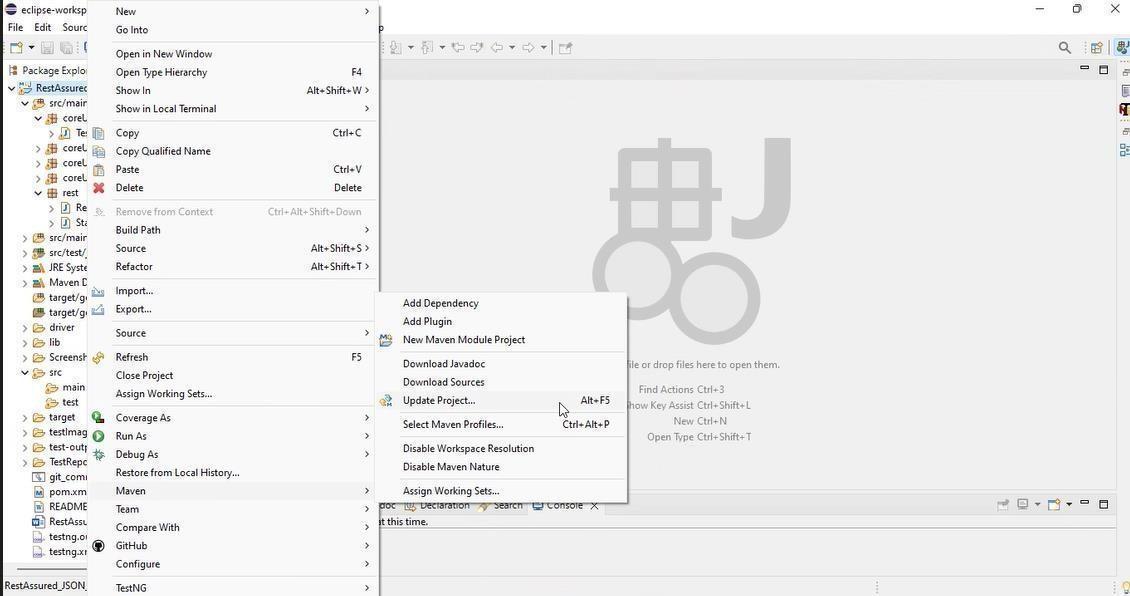
RestAssured API Automation Project

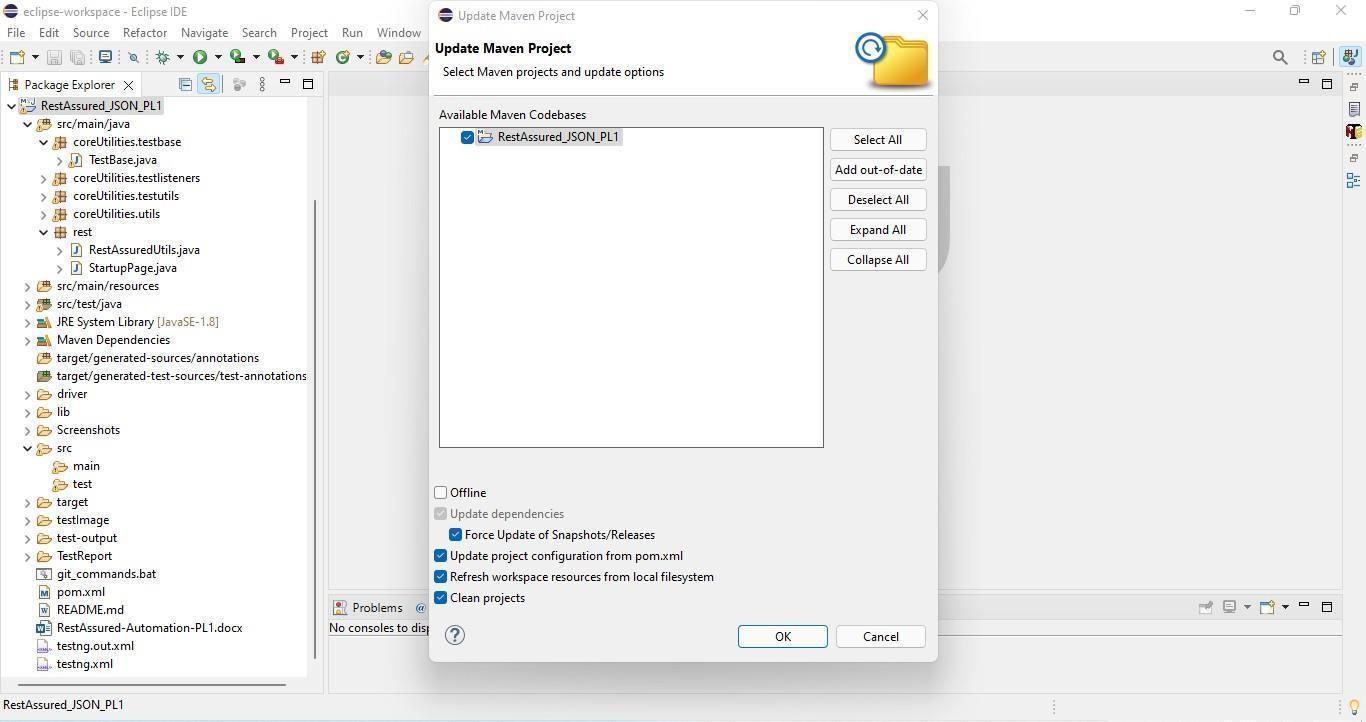
Pre-requisite:

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



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 the “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/coreUtilities/utils/ | FileOperations.java | 1. Contains methods to read from excel file. 2. Method is in templated form. 3. **You will be required to implement these methods as very first activity, because for creating post the data**   **should be read from excel.** |
| /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 ApiUti.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. 3. You implement them as per your need. 4. 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** | Create an Appointment with Authorization in Method:  createAppointmentWithAuth( String endpoint, Map<String, String> body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/ api/Appointment/AddAppointment** 2. The body required as the part of request is passed in requestBody variable in method argument. 3. Include a bearer token for authentication in authorization header. 4. Trigger a POST call. 5. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results.AppointmentId (extracted from Response object) * Status (extracted from Response object)  1. Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results.AppointmentId and complete Response object.  StatusCode should be 200.  Status should be OK.  Results.AppointmentId should not be null. |
| **2** | Get All Applicable Doctors with Authorization in Method:  getAllApplicableDoctorsWithAuth(String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Visit/AppointmentApplicable Doctors** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)  1. Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should be empty.  Results list must have unique performarId. |
| **3** | Cancels an existing appointment with Authorization in Method:  cancelAppointmentWithAuth( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Appointment/AppointmentStatus?appointmentId=** appointmentId**&status=cancelled** 2. Include a bearer token for authentication in authorization header. 3. Trigger a PUT call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * ResultMessage (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty. |
| **4** | Finds if there is any clashing appointment with Authorization in Method:  clashAppointmentWithAuth( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Appointment/CheckClashingAppointment?patientId=**patientId**&requestDate=**requestDate**&performerId=**performerId 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * ResultMessage (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty. |
| **5** | Searches for a patient using specified query parameters with Authorization in Method:  searchPatientWithAuth(String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Patient/SearchRegisteredPatient?search=Test** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have FirstName and ShortName. |
| **6** | Retrieves a list of appointments for a specified performer within a given date range with Authorization in Method:  bookingListWithAuthInRange( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Appointment/Appointments?FromDate=**dateFiveDaysBeforeStr**&ToDate=**currentDateStr**&performerId=**performerId**&status=new** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have AppointmentDate. |
| **7** | Retrieves the complete list of stock details from the pharmacy with Authorization in Method:  AllStockDetailsWithAuth(String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/PharmacyStock/AllStockDetails** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have ItemId. |
| **8** | Retrieves details of the main store in the pharmacy settings  with Authorization in Method:  MainStoreDetailsWithAuth( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/PharmacySettings/MainStore** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should be empty.  Results list must have Name, StoreDescription and StoreId. |
| **9** | Retrieves a list of pharmacy stores and verifies the details of each store with Authorization in Method:  PharmacyStoresWithAuth( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Dispensary/PharmacyStores** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have StoreId and Name. |
| **10** | Retrieves and verifies patient consumption details with Authorization in Method:  PatientConsumption( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/PatientConsumption/PatientConsumptions** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have unique PatientId and PatientName. |
| **11** | Activates a pharmacy counter using counter details with Authorization in Method:  ActivatePharmCount(String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Security/ActivatePharmacyCounter?counterId=**counterId**&counterName=**counterName 2. Include a bearer token for authentication in authorization header. 3. Trigger a PUT call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have CounterName and CounterId. |
| **12** | Deactivates a pharmacy counter with Authorization in Method:  DeactivatePharmCount(String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Security/DeactivatePharmacyCounter** 2. Include a bearer token for authentication in authorization header. 3. Trigger a PUT call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty. |
| **13** | Retrieves and verifies the list of appointment applicable departments with Authorization in Method:  AppointApplicDept( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Master/AppointmentApplicableDepartments** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have unique DepartmentId and DepartmentName. |
| **14** | Retrieves and verifies the list of currently admitted patients with Authorization in Method:  admittedPatientData  ( String endpoint, Object body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Admission/AdmittedPatientsData?admissionStatus=admitted** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have unique PatientId, AdmittedDate and DischargedDate. |
| **15** | Adds a new currency using the provided data with Authorization in Method:  addCurrencyWithAuth( String endpoint, Map<String, String> body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/InventorySettings/Currency** 2. Include a bearer token for authentication in authorization header. 3. Trigger a POST call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty. |
| **16** | Finds if a patient with the requested phone number already exists with Authorization in Method:  findMatchingPatientWithAuth  ( String endpoint, Map<String, String> body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Patient/MatchingPatients?FirstName=**firstName**&LastName=**lastName**&PhoneNumber=**phoneNumber**&Age=**age**&Gender=**gender**&IsInsurance=**isInsurance**&IMISCode=**imisCode 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty. |
| **17** | Lists all registered patients and checks if their patient IDs are unique with Authorization in Method:  getRegisteredPatientsWithAuth( String endpoint, Map<String, String> body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/Patient/SearchRegisteredPatient?search=** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have unique PatientId. |
| **18** | Retrieves and verifies the list of Billing Counters with Authorization in Method:  getBillingCountersWithAuth( String endpoint, Map<String, String> body) | 1. Create an URL by combining the BASE\_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/a pi/billing/BillingCounters** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call. 4. Create an object of type CustomResponse and initialize it with following values:  * Response * statusCode (extracted from Response object) * Results (extracted from Response object) * Status (extracted from Response object)   Return the CustomResponse object from the method. | Returns an object of type CustomResponse containing statusCode, status, Results list and complete Response object.  StatusCode should be 200.  Status should be OK.  Results should not be null and should not be empty.  Results list must have unique CounterId. |

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. **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