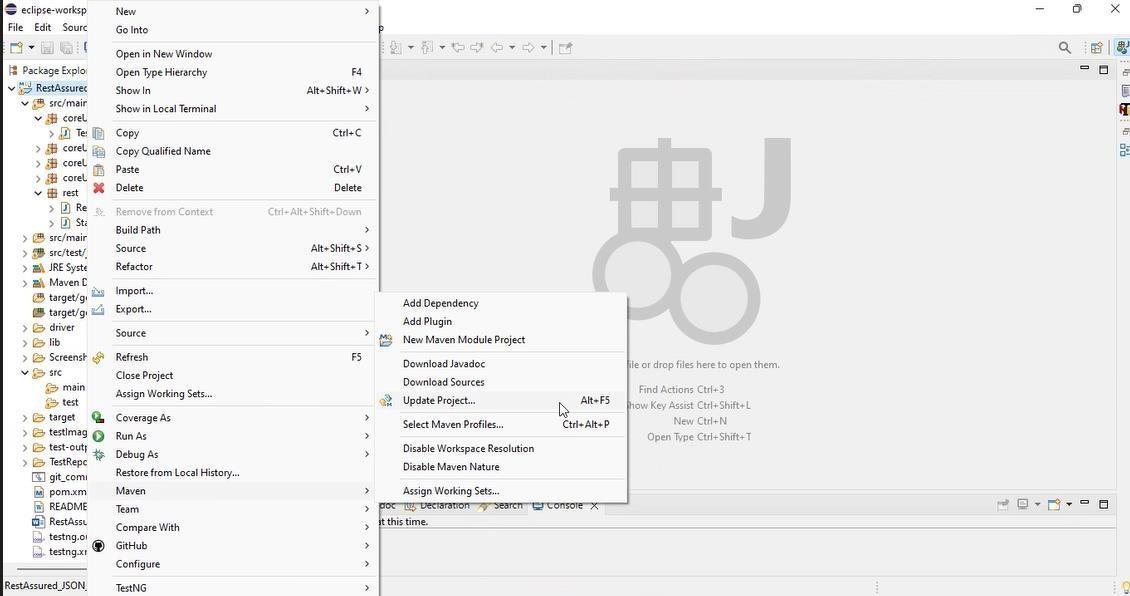
RestAssured API Automation Project

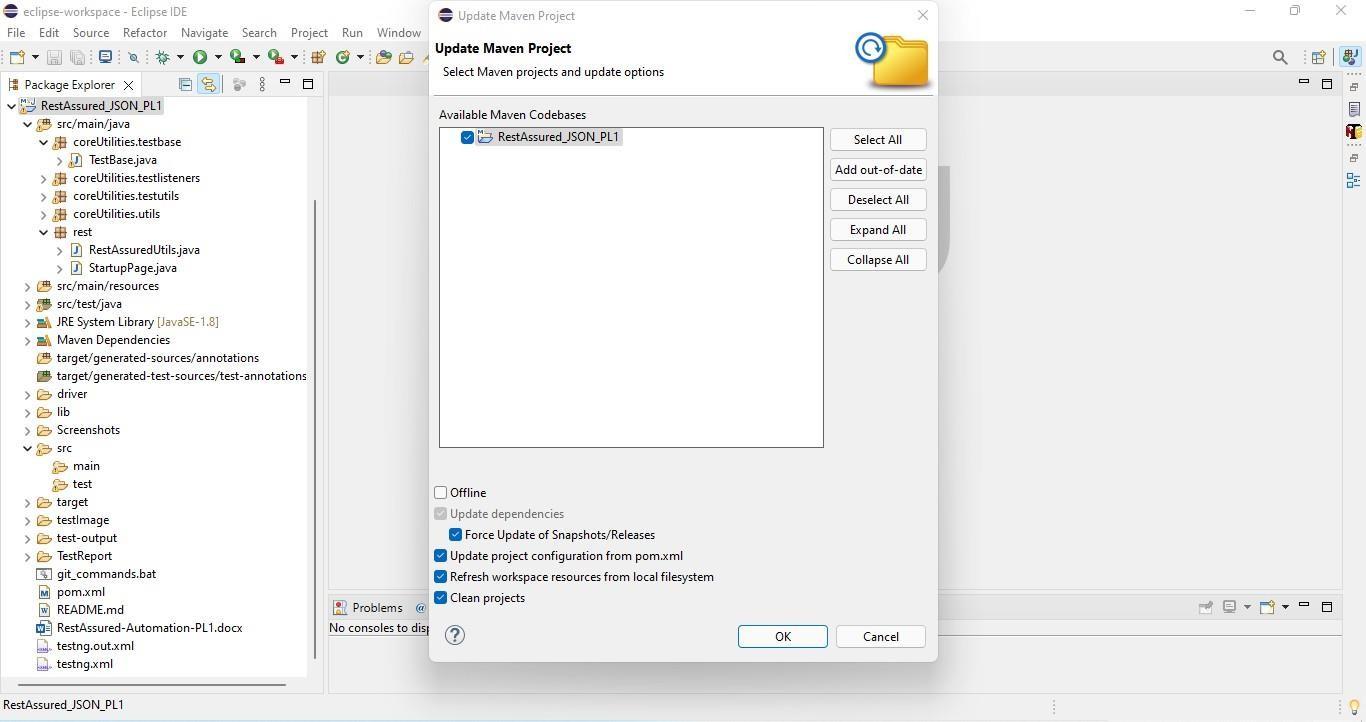
Pre-requisite:

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 “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/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. Call the POST on endpoint i.e **https://healthapp.yaksha.com/ api/Appointment/AddAppointment** 2. Provide the appointment details in the body parameter, including FirstName, LastName, Gender, Age, ContactNumber, AppointmentDate, AppointmentTime, PerformerName, AppointmentType, and DepartmentId. 3. Include an authorization header for the request. 4. Validate the response contains:   - AppointmentId   1. Ensure the AppointmentId is not null and matches the expected values. 2. Validate the response contains the appointment details in the response body. | Returns a 200 OK status with the created appointment details in the "Results" field, including AppointmentId and Status as 'OK'. |
| **2** | Cancel an Appointment with Authorization in Method:  cancelAppointmentWithAuth (String endpoint, Object body) | 1. Call the PUT on endpoint i.e **https://healthapp.yaksha.com/api/ Appointment/AppointmentStatus?appointmentId=+ appointmentId+ "&status=cancelled"** 2. If a body is required, pass it as the body parameter; otherwise, leave it null. 3. Provide the appointment ID and set the status as "cancelled". 4. Include an authorization header. 5. Validate the response contains:   - Results confirming the appointment cancellation.   1. Validate the response confirms the cancellation of the appointment. | Returns a 200 OK status if the appointment is successfully canceled.  or  Returns a 200 OK status with confirmation of appointment cancellation in the "Results" field. |
| **3** | Search for a Patient with Authorization in Method:  searchPatientWithAuth(String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api/**  **/Patient/SearchRegisteredPatient?search=Test**   1. Include the necessary query parameters in the body if required. 2. Add an authorization header to the request. 3. Validate the response contains:   - FirstName  - LastName  - ShortName   1. Validate the response contains patient details matching the search criteria, including fields like PatientId, ShortName, FirstName, LastName, Age, etc. | Returns a 200 OK status with a list of patients matching the search criteria in the response.  or  Returns a 200 OK status with the list of patients matching the search criteria in the 'Results' field. |
| **4** | Retrieve a List of Appointments for a Specified Performer in Method:  bookingListWithAuthInRang e(String endpoint, Object  body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api/**  **/Appointment/Appointments?From Date="+ dateFiveDaysBeforeStr + "&ToDate=" + currentDateStr + "&performerId=" + performerId + "&status=new"**   1. Include query parameters as FromDate, ToDate, and PerformerId. 2. Add the required authorization header. 3. Validate the response contains:   - AppointmentDate   1. Ensure 'AppointmentDate' is within the specified range. | Returns a 200 OK status with the list of appointments within the specified date range in the 'Results' field. |
| **5** | Retrieve Main Store Details in Method:  MainStoreDetailsWithAuth( String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api**  **/PharmacySettings/MainStore**   1. Add the necessary authorization header. 2. Validate the response contains:   - StoreId  - Name  - StoreDescription   1. Validate that:    1. The response status code is 200 (OK).    2. Fields Name, StoreDescription, and StoreId are not null.    3. The Status field in the response is “OK”. | Returns a 200 OK status with the main store details in the 'Results' field. |
| **6** | Retrieve a List of Pharmacy Stores in Method:  PharmacyStoresWithAuth(String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api**  **/Dispensary/PharmacyStores**   1. Add the necessary authorization header. 2. Validate the response contains:   - StoreId  - Name   1. Validate that:    1. The response status code is 200 (OK).    2. Each store in the results has non- null StoreId and Name fields.    3. The Status field in the response is “OK”. | Returns a 200 OK status with the list of pharmacy stores in the 'Results' field. |
| **7** | Activate a Pharmacy Counter Using Details in Method:  ActivatePharmCount(String endpoint, Object body) | 1. Call the PUT on endpoint i.e   **https://healthapp.yaksha.com/api**  **/Security/ActivatePharmacyCounter**  **?counterId=" + counterId + "&counterName=" + counterName**   1. Include counterId and counterName in the query parameters. 2. Add the required authorization header. 3. Validate the response contains:   - CounterId  - CounterName   1. Validate the response to ensure:    1. The status code is 200 (OK).    2. The Results field contains non- null CounterName and CounterId.    3. The Status field is "OK". | Returns a 200 OK status with details of the activated pharmacy counter in the Results field. |
| **8** | Deactivate a Pharmacy Counter in Method:  DeactivatePharmCount(String endpoint, Object body) | 1. Call the PUT on endpoint i.e   **https://healthapp.yaksha.com/api**  **/Security/DeactivatePharmacyCounter**   1. Include the necessary authorization header. 2. Validate the response to ensure:    1. The Results field contains a StatusCode of "200".    2. The Status field is "OK". | Returns a 200 OK status with details confirming the pharmacy counter has been deactivated. |
| **9** | Retrieve Appointment Applicable Departments in Method:  AppointApplicDept(String endpoint, Object body) | 1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api**/ **Master/AppointmentApplicableDepartments** 2. Add the required authorization header. 3. Validate the response contains the list of applicable departments, including:   - DepartmentId  - DepartmentName   1. Validate the response to ensure:    1. The status code is 200 (OK).    2. Each item in the Results list contains non-null DepartmentId and DepartmentName.    3. The Status field is "OK". | Returns a 200 OK status with the list of applicable departments in the 'Results' field. |
| **10** | Retrieve Admitted Patients Data in Method:  admittedPatientData(String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api**/  **/Admission/AdmittedPatientsData? admissionStatus=admitted**   1. Add the necessary authorization header. 2. Validate the response contains the list of admitted patients, including:   - PatientId  - AdmittedDate  - DischargedDate   1. Validate the response to ensure:    1. The status code is 200 (OK).    2. Each item in the Results list contains non-null PatientId and AdmittedDate, and DischargedDate is null.    3. The Status field is "OK". | Returns a 200 OK status with the list of admitted patients in the 'Results' field. |
| **11** | Retrieve Profile Details of Admitted Patients in Method:  getProfileDetails(String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api**  **/Employee/Profile?empId=" + expectedEmployeeId**   1. If a request body is needed, include it in the body parameter. 2. Ensure an authorization header is included. 3. Validate the response contains the profile details:   - EmployeeId  - FirstName  - LastName  - DateOfBirth  - Email  - UserName   1. Ensure the EmployeeId is not null. 2. Validate the response contains the profile details of admitted patients in the "Results" field. | Returns a 200 OK status with profile details of admitted patients in the "Results" field. |
| **12** | Add a New Department in Method:  addDepartment(String endpoint, Map<String, String> body) | 1. Call the POST on the endpoint i.e., **https://healthapp.yaksha.com/api**  **/Settings/Department**  2. Provide DepartmentCode and DepartmentName in the request body.  3. Include the authorization header.  4. Validate the response contains:  - DepartmentId  - DepartmentCode  - DepartmentName  5. Ensure the DepartmentId is not null and matches the expected values.  6. Validate the response to ensure  the department is added successfully. | Returns a 200 OK status with the created department details in the "Results" field. |
| **13** | Retrieve the List of Departments in Method:  getDepartmentsList(String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api**  **/Settings/Department**   1. If a request body is needed, include it in the body parameter. 2. Ensure an authorization header is included. 3. Validate the response contains the list of departments, including:   - DepartmentId  - DepartmentName.   1. Ensure the DepartmentId and DepartmentName are not null. 2. Validate the response contains the list of departments in the "Results" field. | Returns a 200 OK status with the list of departments in the "Results" field. |
| **14** | Edit Department Details in Method:  editDepartmentDetails(String endpoint, Map<String, String> body) | 1. Call the PUT on endpoint i.e   **https://healthapp.yaksha.com/api**  **/Settings/Department**   1. Include DepartmentCode, DepartmentName, and other to be updated fields in the body. 2. Add the required authorization header. 3. Validate the response to ensure:    1. The status code is 200 (OK).    2. The Results field should contain the DepartmentCode and DepartmentName as non- null details.    3. The Status field is "OK". | Returns a 200 OK status with the updated department details in the response. |
| **15** | Retrieve Imaging Data Response in Method:  getImagingDataResponse(String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api**  **/RadiologySettings/ImagingTypes**   1. If a request body is needed, include it in the body parameter. 2. Ensure an authorization header is included. 3. Validate the response contains a list of imaging data:   - ImagingTypeId  - ImagingTypeName   1. Ensure ImagingTypeId and ImagingTypeName are not null. 2. Validate the response contains imaging data in the “Results” field. | Returns a 200 OK status with imaging data in the "Results" field. |
| **16** | Retrieve Signatories Details by Department in Method:  getsignatoriesDetails(String endpoint, Object body) | 1. Call the GET on endpoint i.e   **https://healthapp.yaksha.com/api**  **/Master/Signatories?departmentName=radiology**   1. If a request body is needed, include it in the body parameter. 2. Ensure an authorization header is included. 3. Validate the response contains:   - EmployeeId  - FirstName  - LastName   1. Ensure EmployeeId, FirstName, and LastName are not null. 2. Validate the response contains a list of signatories in the "Results" field. | Returns a 200 OK status with signatories details in the "Results" field. |

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