**Testing the CRUD Operations and performance of List of Users**

1. CRUD operations using POSTMAN **:**

Created a collection named **Project collection** and added two requests

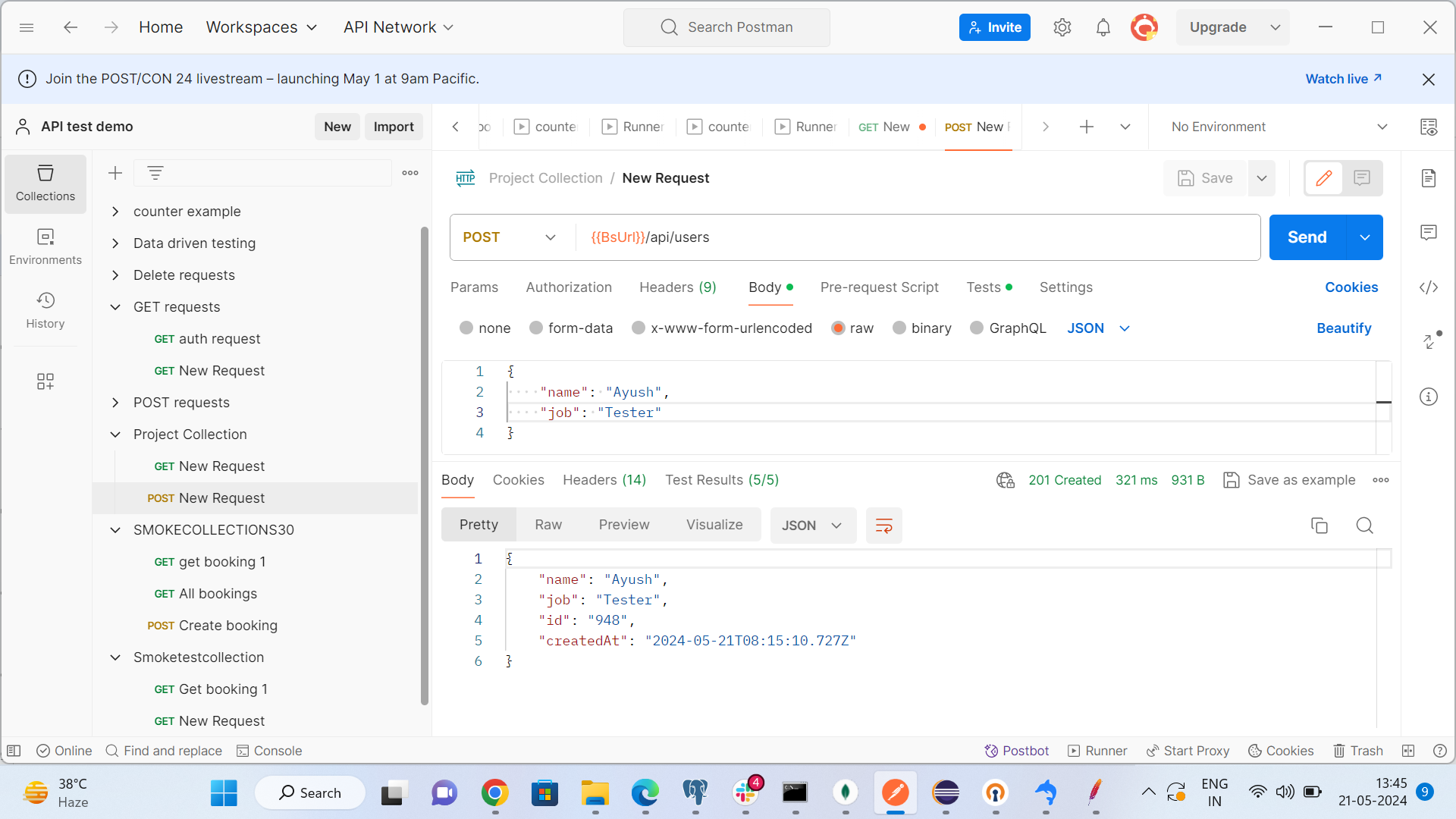
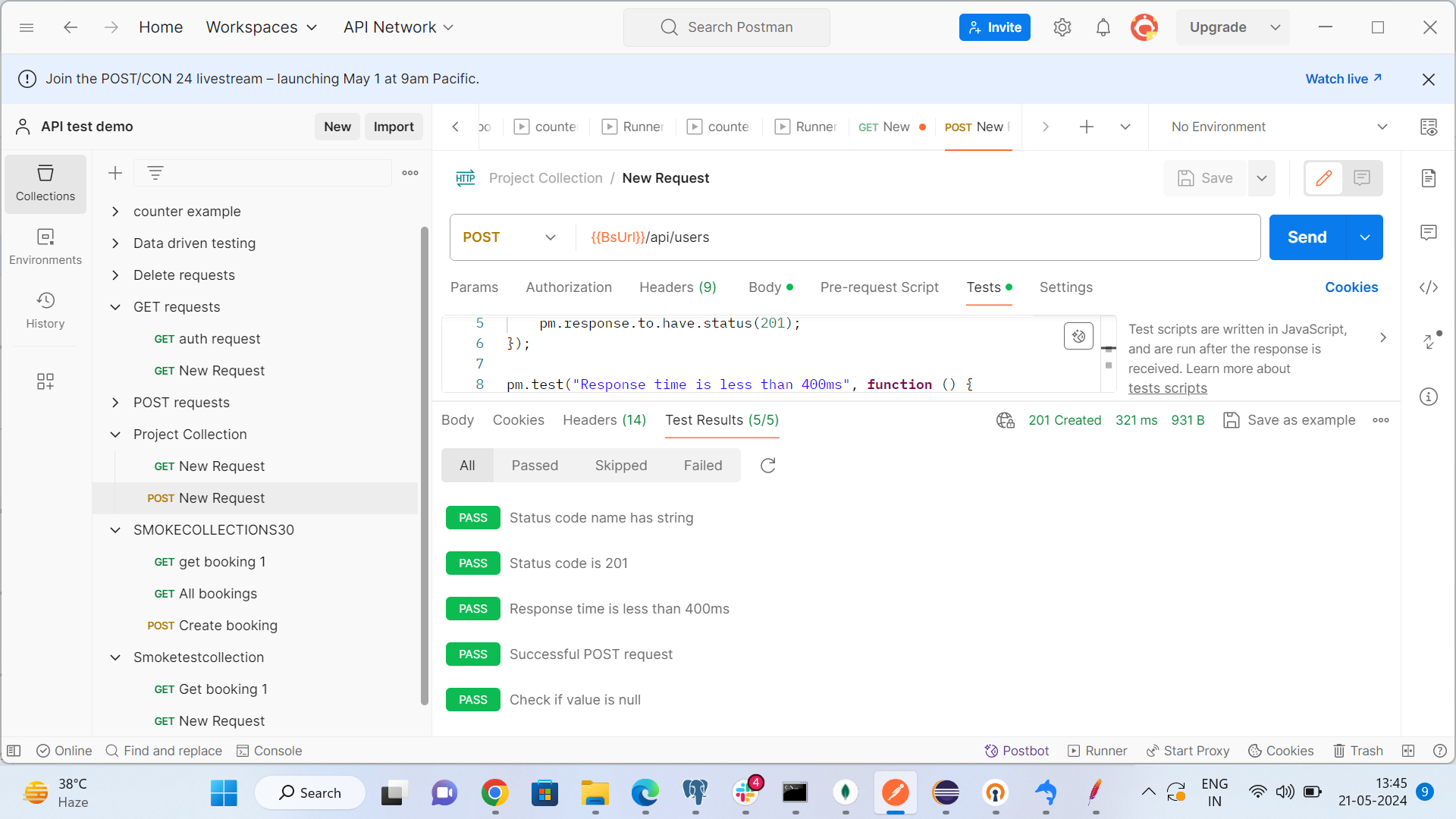
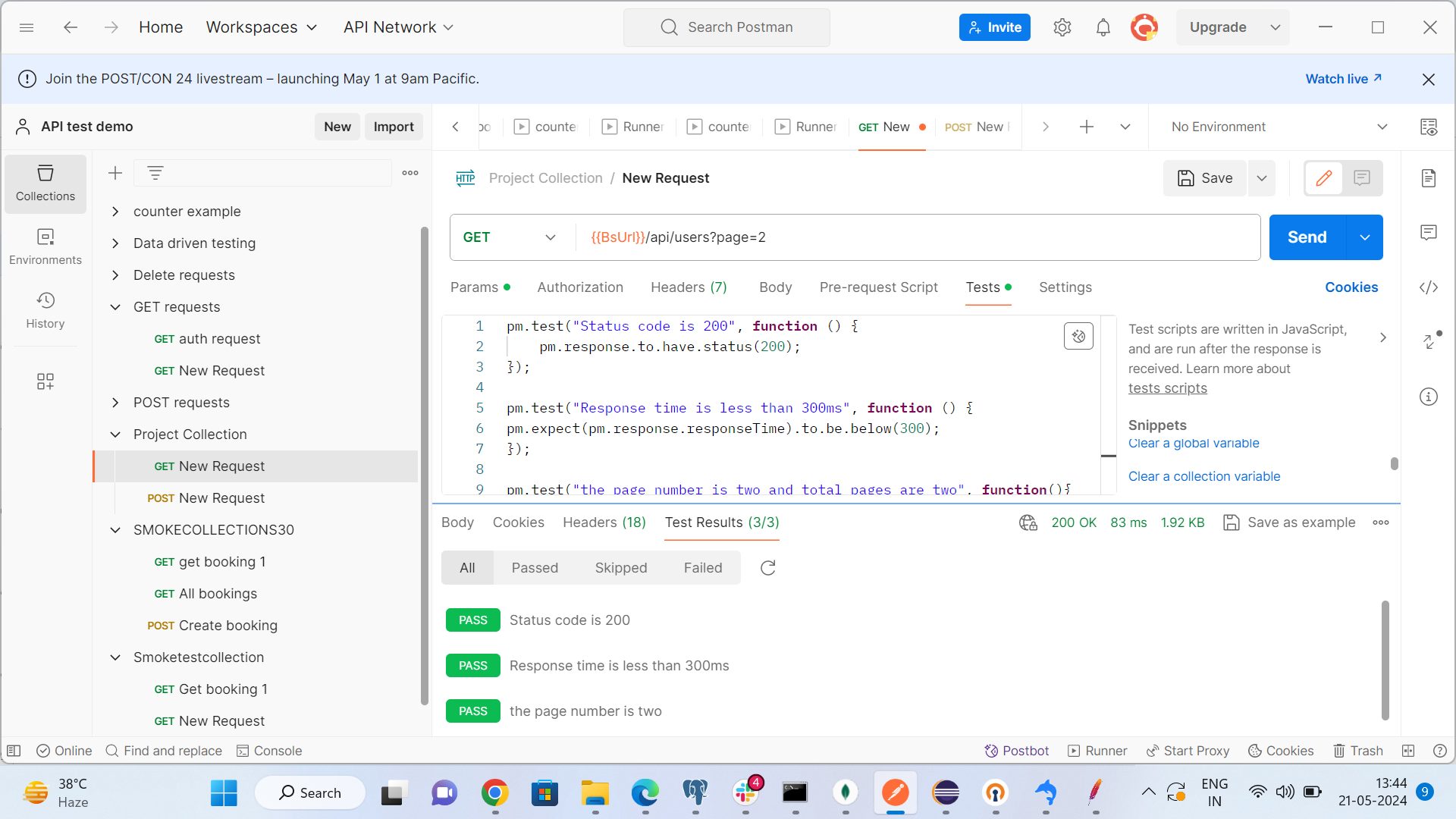
i) **GET request**, by setting the uri(<https://reqres.in>) as variable {{BsUrl}} and using api as {{BsUrl}}/api/users?page=2 .

Also validated the response with applying test script validations as:

| GET Request | {{BsUrl}}/api/users?page=2 . |
| --- | --- |
| Validations | 1. Response code = 200 2. Response time < 300 ms 3. responseJson.data validation for page count |

ii) **POST request**, for the url({{BsUrl}}/api/users)

| POST request | {{BsUrl}}/api/users |
| --- | --- |
| Request body | {  "name": "Ayush",  "job": "Tester"  } |
| Validations | 1. Response code = 201 2. Response time < 300ms 3. Response body contains string 4. Successful post request 5. Id data should not be NULL. |



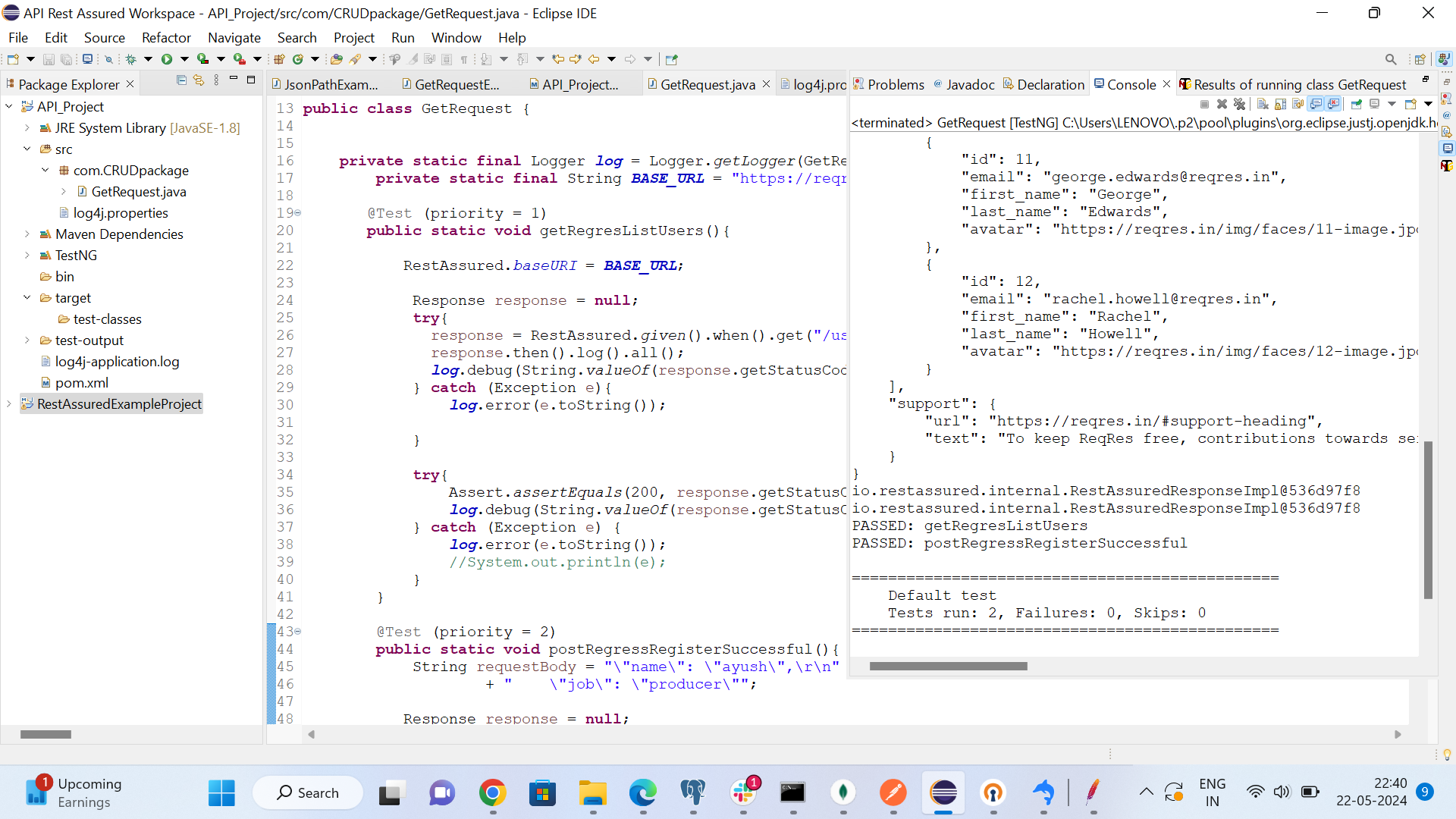
2. Eclipse Rest assured and log4j response **:**

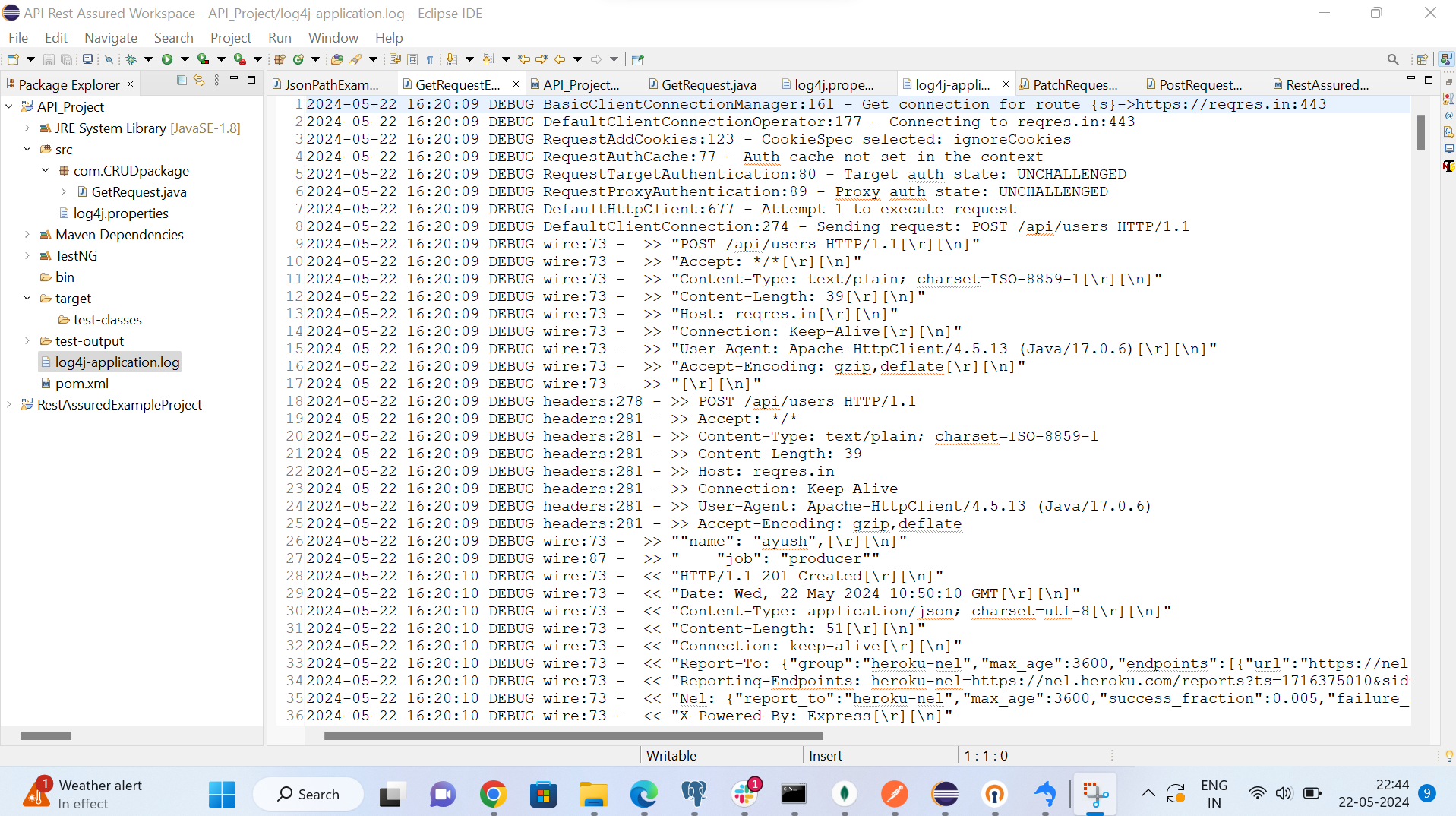
Step1 :- Created a maven project in eclipse named **API\_Project** and added required dependencies in that:

* Rest Assured
* Json schema validator
* testNG
* Log4j
* Hamcrest

Step2:- Created a class “**GetRequest**” in package “com.CRUDpackage” and wrote the required code for both get and post method in this with logger class instantiating for logging response in log file.

Step3:- Added a log4j property file name “log4j.properties” configuring console appender and file appender for logging in log4j-application.log.



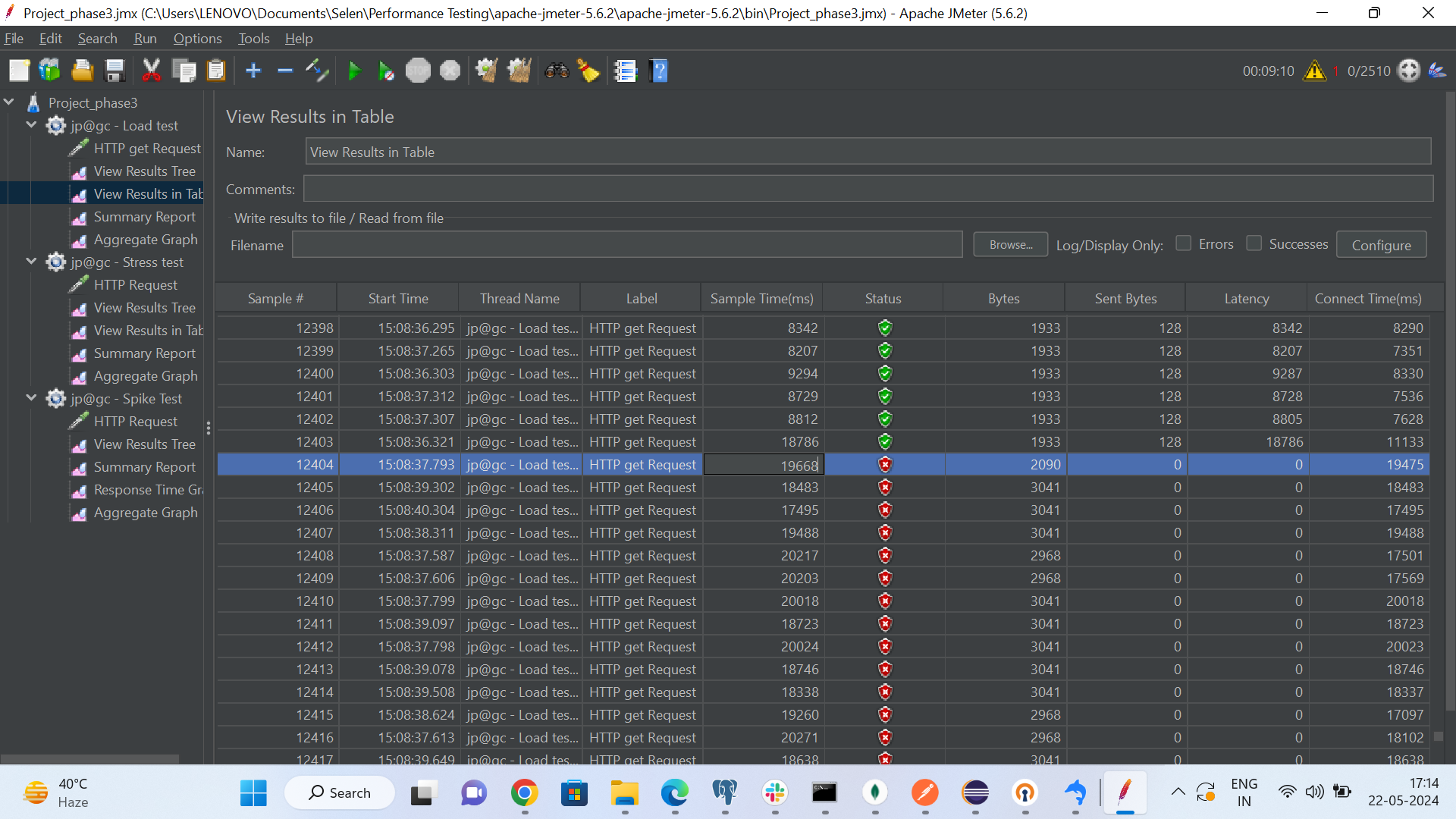
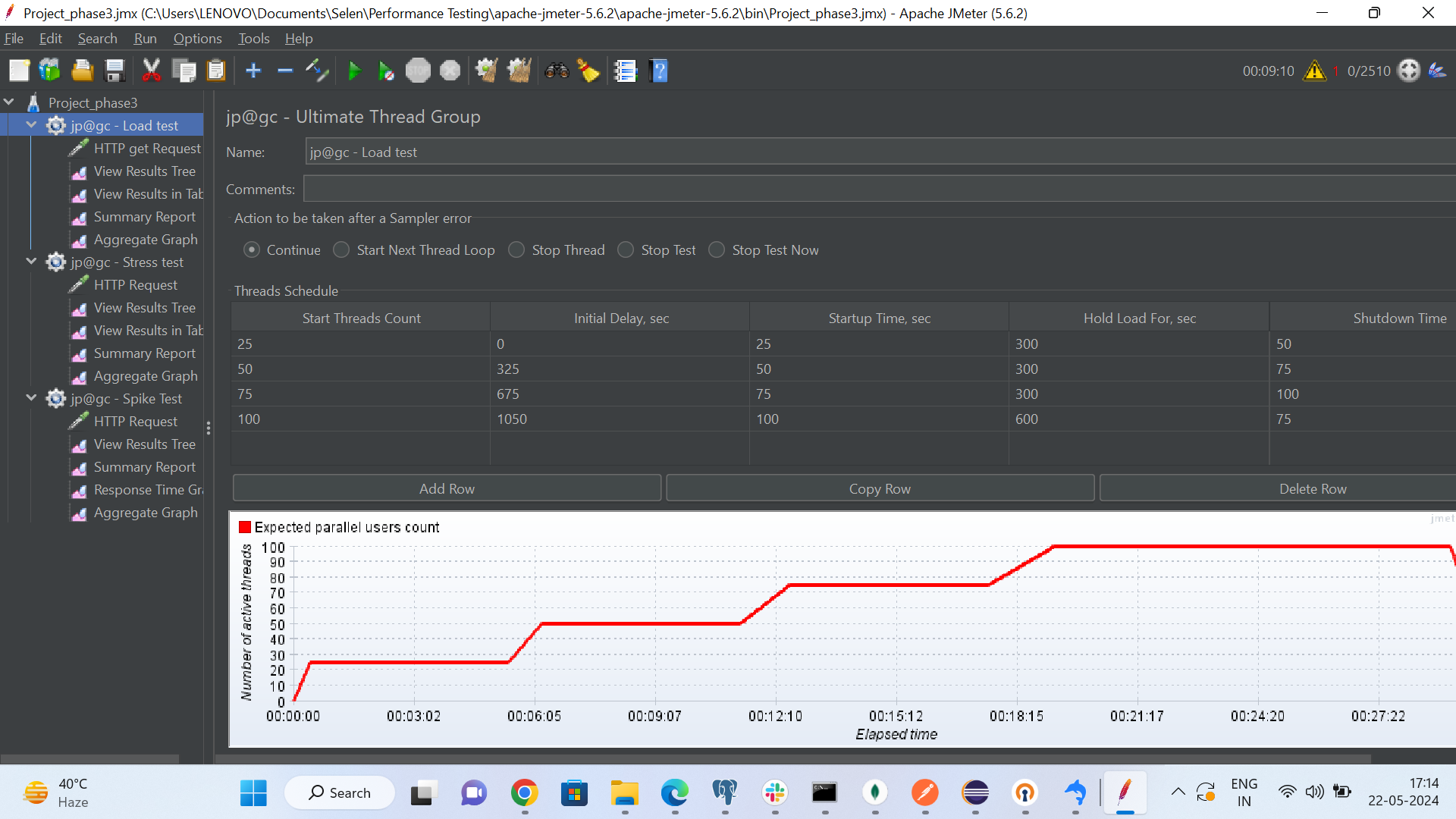


3. **JMeter** :

* Created a test plan named **Project\_phase3** and added an http request sampler created for “<https://reqres.in/api/users?page=2>”.
* Added jp@gc - Ultimate Thread Group for different type of performance tests as follow:

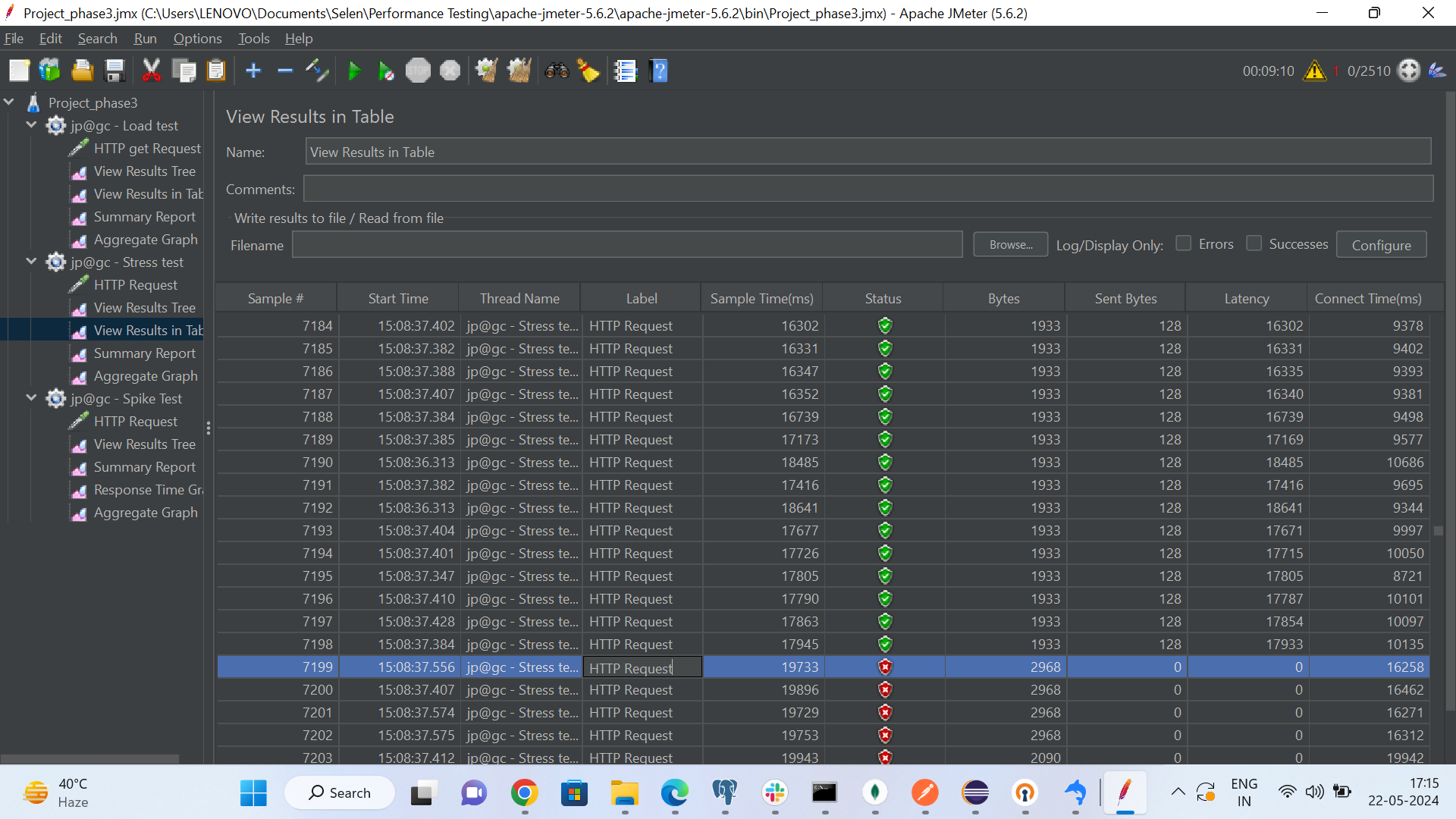
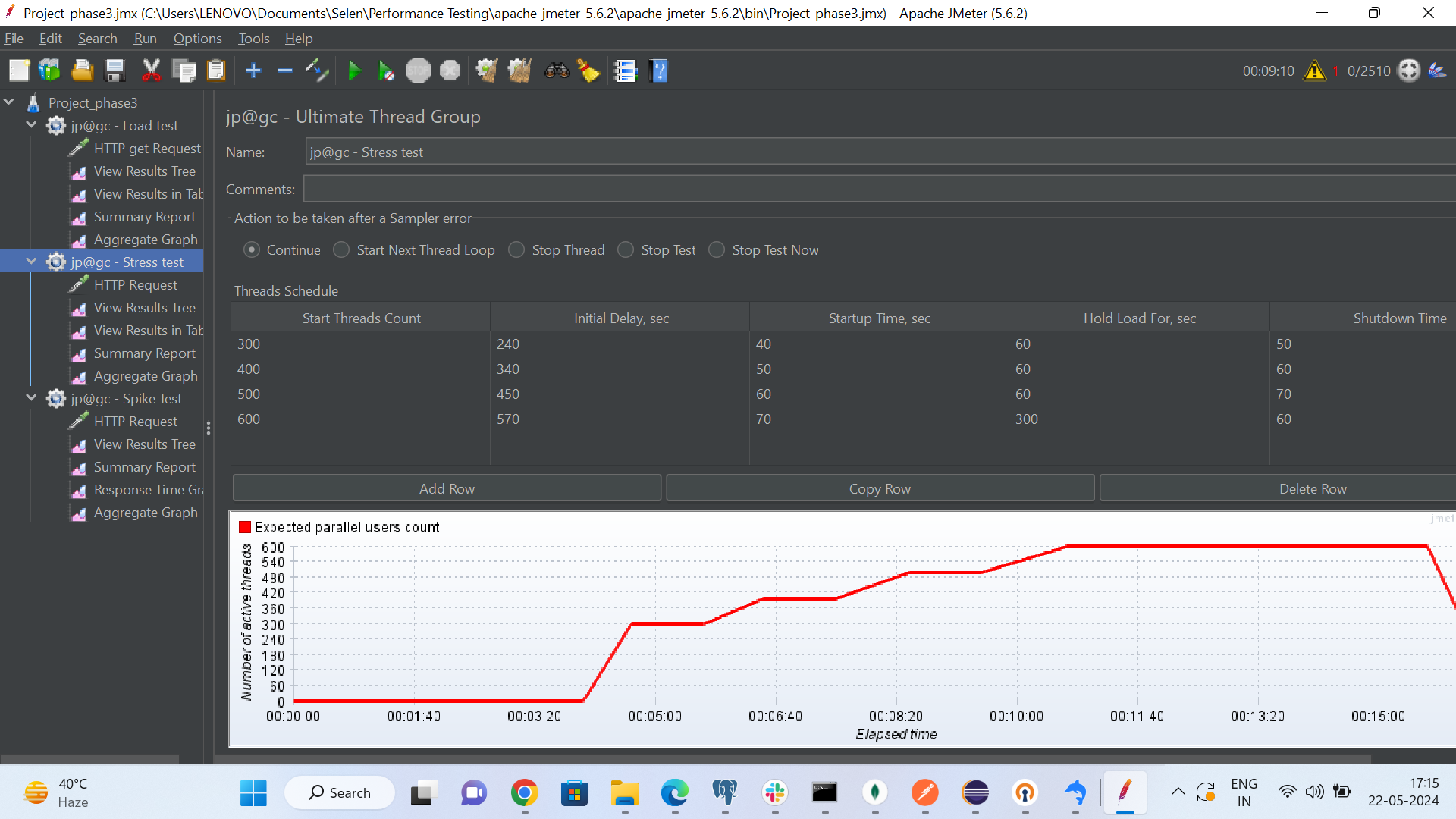
i) **Load Testing** : Thread group is added by scheduling load of maximum up to 100 thread count for hold time of 10 mins.

Added listeners for tracking performance and break point of response .



ii) **Stress testing** :

* The purpose of the stress testing is to discover the failing point of the system where the performance is seriously degraded.
* Thread schedule is configured with step up additions of thread counts towards a maximum value. The system might degrade seriously before reaching the maximum loads configured in the thread schedule



iii) **Spike testing:**

This testing is used to check the performance of api on sudden load or increase in users in single time.

Following thread group has three spike points, and the graph shows the breakpoint of this .

