API Performance Testing

Approach

1. I have used a JMETER as an open source tool.

2. It's free and platform independent.

3. Using JMETER we can easily add different assertions/create virtual users/measure response time etc.

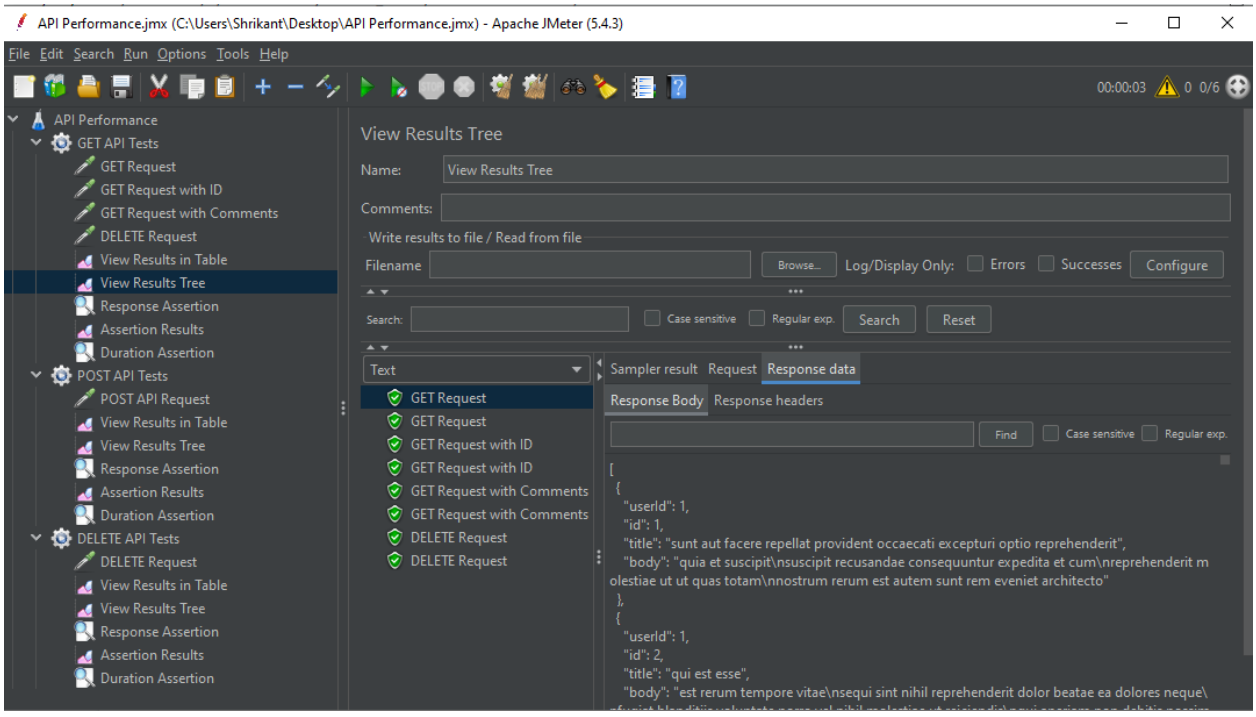
Programming language

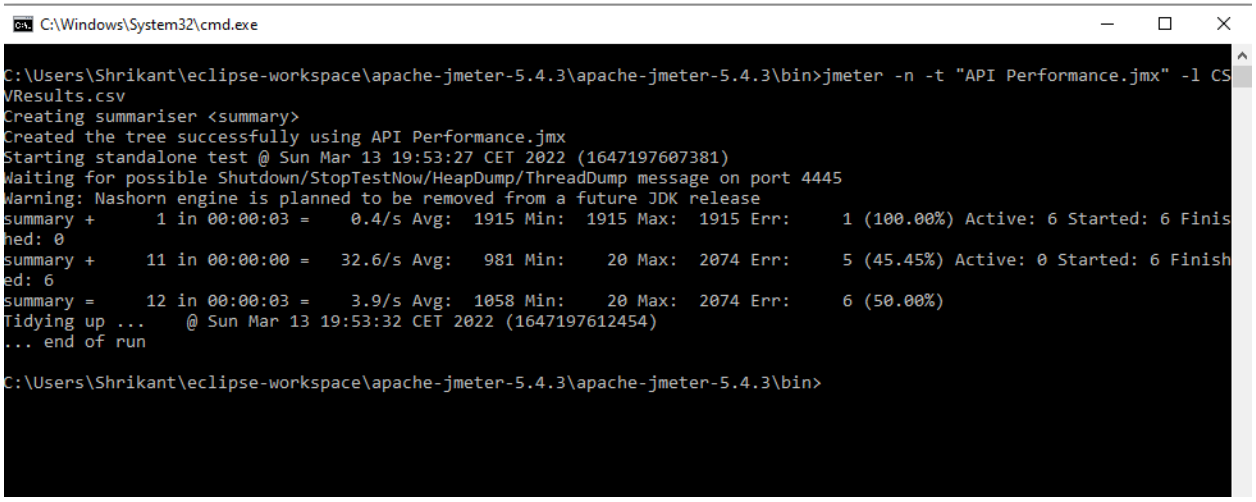
1. JMeter has their own UI + we can use bash scripting as well.

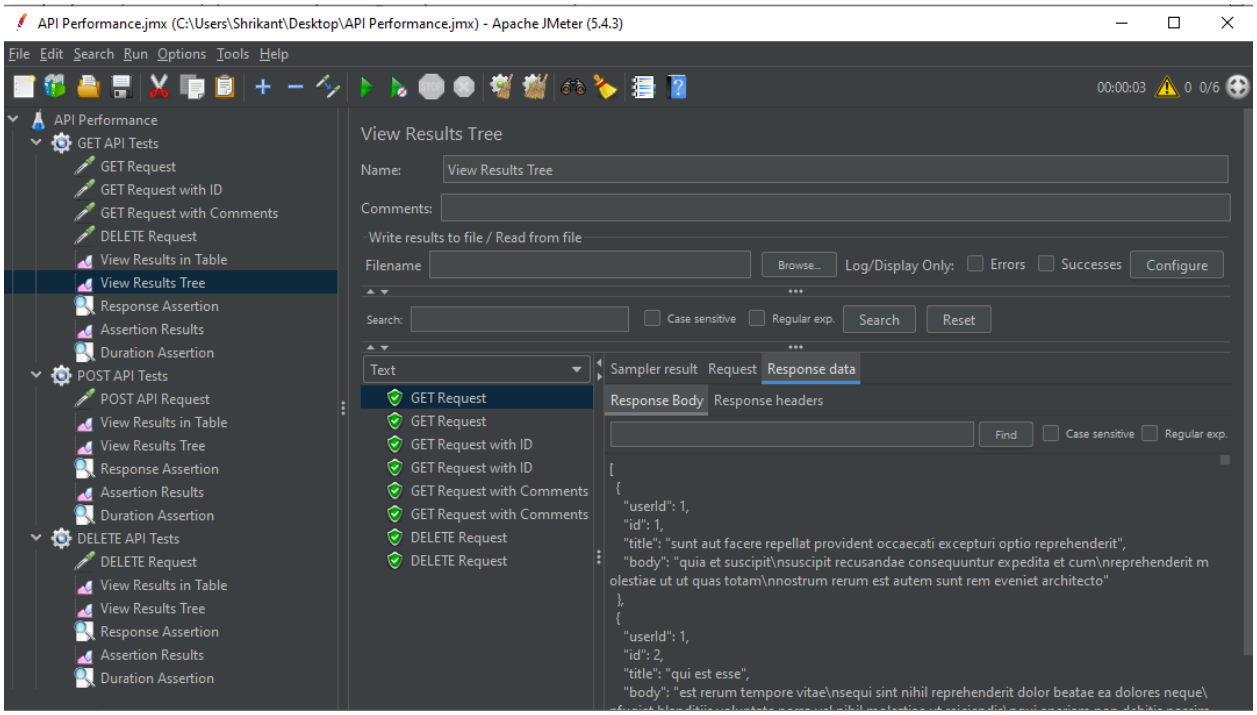
How to run

1. Clone the entire project on a local machine from GitHub.

2. Execute test from JMeter UI







	A	B	C	D	E	F	G	H	
1	timeStamp	elapsed	label	responseCode	responseMessage	threadName	dataType	success	failureMessage
2	1647197610180	1915	GET Request	200	OK	GET API Tests 1-1	text	false	The operation lasted too long: It took 1,915 milliseconds
3	1647197610180	1915	GET Request	200	OK	GET API Tests 1-2	text	false	The operation lasted too long: It took 1,915 milliseconds
4	1647197612112	25	GET Request with ID	200	OK	GET API Tests 1-2	text	true	
5	1647197612117	24	GET Request with ID	200	OK	GET API Tests 1-1	text	true	
6	1647197612138	20	GET Request with Comments	200	OK	GET API Tests 1-2	text	true	
7	1647197612142	31	GET Request with Comments	200	OK	GET API Tests 1-1	text	true	
8	1647197610180	2058	DELETE Request	200	OK	DELETE API Tests 3-2	text	false	The operation lasted too long: It took 2,058 milliseconds
9	1647197610180	2061	DELETE Request	200	OK	DELETE API Tests 3-1	text	false	The operation lasted too long: It took 2,061 milliseconds
10	1647197610180	2074	POST API Request	201	Created	POST API Tests 2-1	text	false	The operation lasted too long: It took 2,074 milliseconds
11	1647197610180	2074	POST API Request	201	Created	POST API Tests 2-2	text	false	The operation lasted too long: It took 2,074 milliseconds
12	1647197612174	226	DELETE Request	200	OK	GET API Tests 1-1	text	true	
13	1647197612158	283	DELETE Request	200	OK	GET API Tests 1-2	text	true	
14									

Current Automation Features -

1. Created Test Plan in JMeter for GET/POST/DELETE.

2. Added a different listeners to see details ex - response time/status codes/response body.

3. Added **assertions** on response time.

4. Able to create parallel virtual users to see performance in case of load.

What features we can add to framework in next phase

2. Integrate Automation with **Jenkins** for continuous integration.

3. Pull code at runtime from **GitHub** by Jenkins, execute, and send emailable reports to relevant audiences.

4. Create Automation dashboard to **monitor 24*7 Jenkins jobs** status.

5. Store automation reports on cloud ex – S3 bucket in AWS.

6. Manage most of Automation configuration as command-line arguments.

8. Execute same scripts on different environments ex – Integration/Staging/Preview/Production etc.

9. Integrate Automation with different Third party tools ex – Slack for notification purpose / Integrate with JIRA to update automation results.