Table of Contents

1. Introduction

The application testing applied to our system (Bid Buy Sell) for concurrent programs which is main architectural challenges that will be addressed in handling multiple concurrent transactions and real-time information updates. Considering the relevance of concurrent programs testing, several research have been conducted in this area, especially involving adaptation of the techniques and criteria applied in sequential programs. This document will present a systematic view, which is a systemic procedures and experiments in testing that will be applied to our application.

The objective of the concurrent programming is to increase application performance, allowing better use of available resources. A concurrent application consists of several processes that interact to solve a problem, reducing the computational time in several application domains.

For this testing, we are concern on putting our software architecture into extreme testing, so to prove that the desired qualities are satisfied.

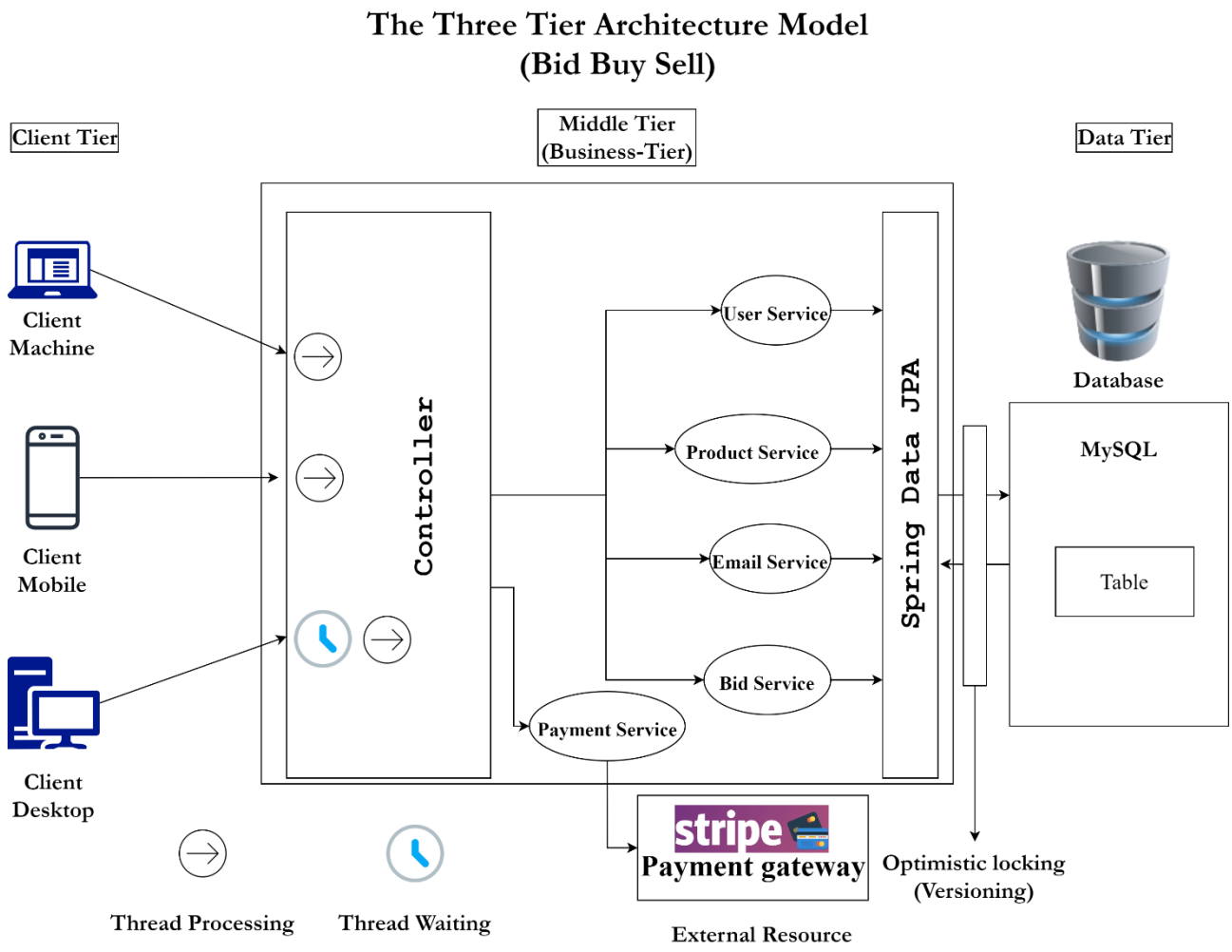


Figure 1. Architectural Design

2. Background on Testing

Concurrent user load testing sends traffic to a web application to stress the infrastructure. Specific metrics are observed and recorded during the test, and system response times during periods of sustained heavy load are measured. With JMeter, we can increase the number of concurrent users slowly or quickly throughout our test to record how performance is affected under sustained load.

The idea behind concurrent user testing is to identify the response time of a web app for a specified number of concurrent users making requests to our website. Concurrent user testing measures how long it takes the server to respond to a specified number of concurrent requests. A concurrent user test is often used to identify bottlenecks in the performance of a website – *basically to find out how many concurrent users can make requests of a website until the performance of the site is significantly degraded.*

Send 10 to 10,000+ simultaneous users to our web application to test the performance. We know there are limits to how much traffic our website can handle, but do you know what those limits are? There are several layers supporting our website that could be a potential bottleneck including web servers, file servers, routers, firewalls, and more. Once we have identified the breaking point, we can then strengthen the weak spots in our system.

Simultaneous user testing is sometimes mistakenly referred to as called concurrent user testing, however, there is a difference, even if the words themselves indicate that something is happening or occurring at the same time. While concurrent users refer to the number of users using or landing on our website or application at any given time, simultaneous users are users, or visitors, that are carrying out a specific transaction at a specific point in time.

For example, you may have 100 different visitors on a specific page, how does performance compare when 40 users log into our portal at the same time? This is important factors to understand, as it directly affects the user experience.

3. Scope

Handling multiple concurrent transactions is one of the factors to be checked where there will be many users active on the system and they will be able to bid on the products. But multiple users may place bids on the same product at the same time. Another concern is updating the product price in real time for ongoing bidding processes. If the product is being bided by someone, there will be multiple users who will be on the same page waiting or trying to bid. If one bid is updated, other users must instantly see the updated bid amounts on their systems. This must be nearly real-time.

4. Testing tools

Test management and automation tools are required for test execution. For performance, load and security testing the test approach and tools are required. Therefore, we are using JMeter and Robot Framework.

4.1 JMeter

JMeter for the purpose of load testing is considered as the one of the best tools. Apache JMeter is a Java open-source GUI-based software that is used as a performance testing tool for analyzing and measuring the performance of our application.

Performance testing can be widely categorized as

1. **load testing** - focuses on testing whether the system can handle *concurrent* user accesses without breaking.
2. **stress testing** focuses on how our system copes with high load and limited resources under constrained conditions.

JMeter is extensible, meaning that we can write custom script, using the pre-built interface of JMeter, making it even more powerful.

Requirements:

- Make sure you got java installed (Requires Java 8+)
- Install JMeter via http://jmeter.apache.org
- Download the Plugins Manager JAR file and put it into JMeter's lib/ext directory. Then start JMeter and go to "Options" menu to access the Plugins Manager.

4.2 Robot Framework

Robot Framework is a test automation framework that is Python-based. This framework supports writing an object-page model in keyword driven methodology. Robot Framework allows users to write their own test-cases without programming knowledge. In addition to this, it has very descriptive logs including complete debugging capabilities through readable logs.

To use Robot Framework, we must make sure our system has following prerequisites:

1. JDK 1.8-1
2. Python
3. Robot Framework Library
4. Robot Framework Selenium Library
5. Chrome Driver

Before having the script, we need to install VS Code for Developing Test Script and configure or add few extensions which are related to robot framework. For more detail, please see in appendix with setup procedure.

5. Test Approach

5.1 Use Case Scenario – Concurrent Load Testing

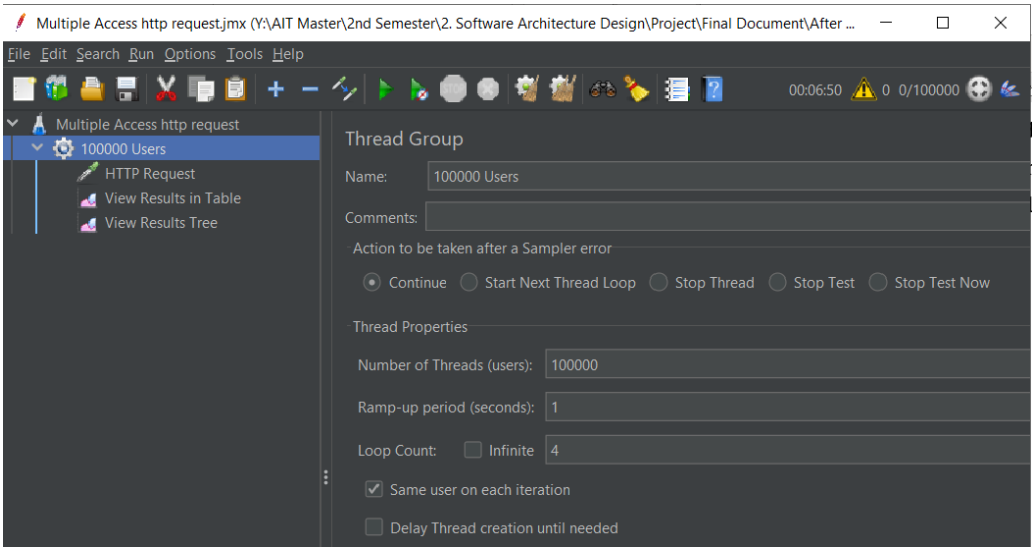
With JMeter, there are a variety of ways we can conduct a concurrent user test. For example, we can begin load testing with as few as 10 concurrent users and run these users for 5 minutes to establish our baseline performance metrics. After establishing a baseline, we can increase the number of concurrent users by 10 users a minute until we reach 100 concurrent users. We may choose to follow that up with a test run for another 5 minutes for every 100 additional concurrent users to be sure that the results level out.

Some factors that may cause drops in web page response time while adding concurrent users include additional allocation of memory on the webs server or additional concurrent database connections on the backend. These could easily cause a drop in the average page load speed while waiting for the system resources to become free only to drop back to normal levels once the resources have been allocated.

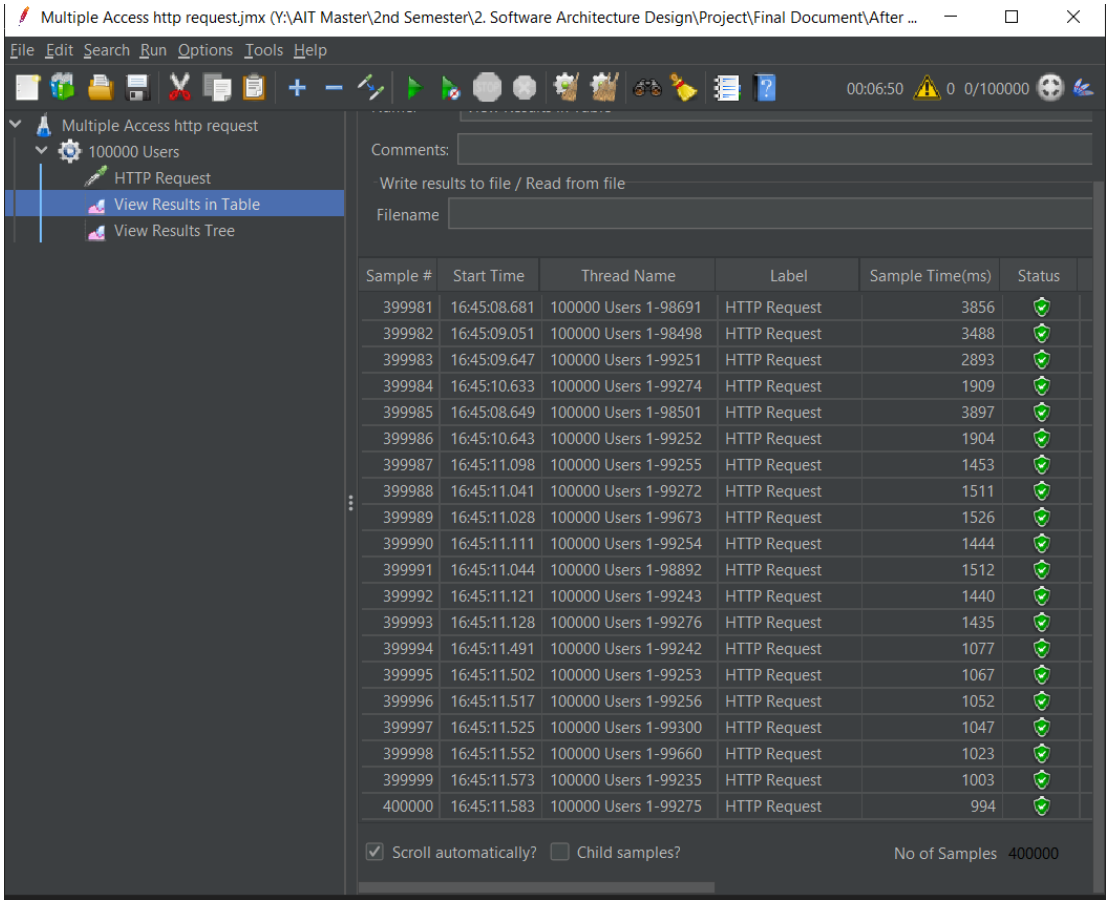
To test this, we may choose to run a test of 500 to 1000 concurrent users, or until we feel we have adequately proven that our web app is capable of handling peak user numbers. These tests can be used to identify both the volume of users that causes unacceptable page load speeds as well as the number of concurrent page requests that causes the web app to crash. This may be done by running additional load tests that start at a higher volume of users in order to push the system to its limits.

5.1.1 Test multiple access

We configure 100000 threads in 4 loops, making it 400000 threads or users access the localhost.



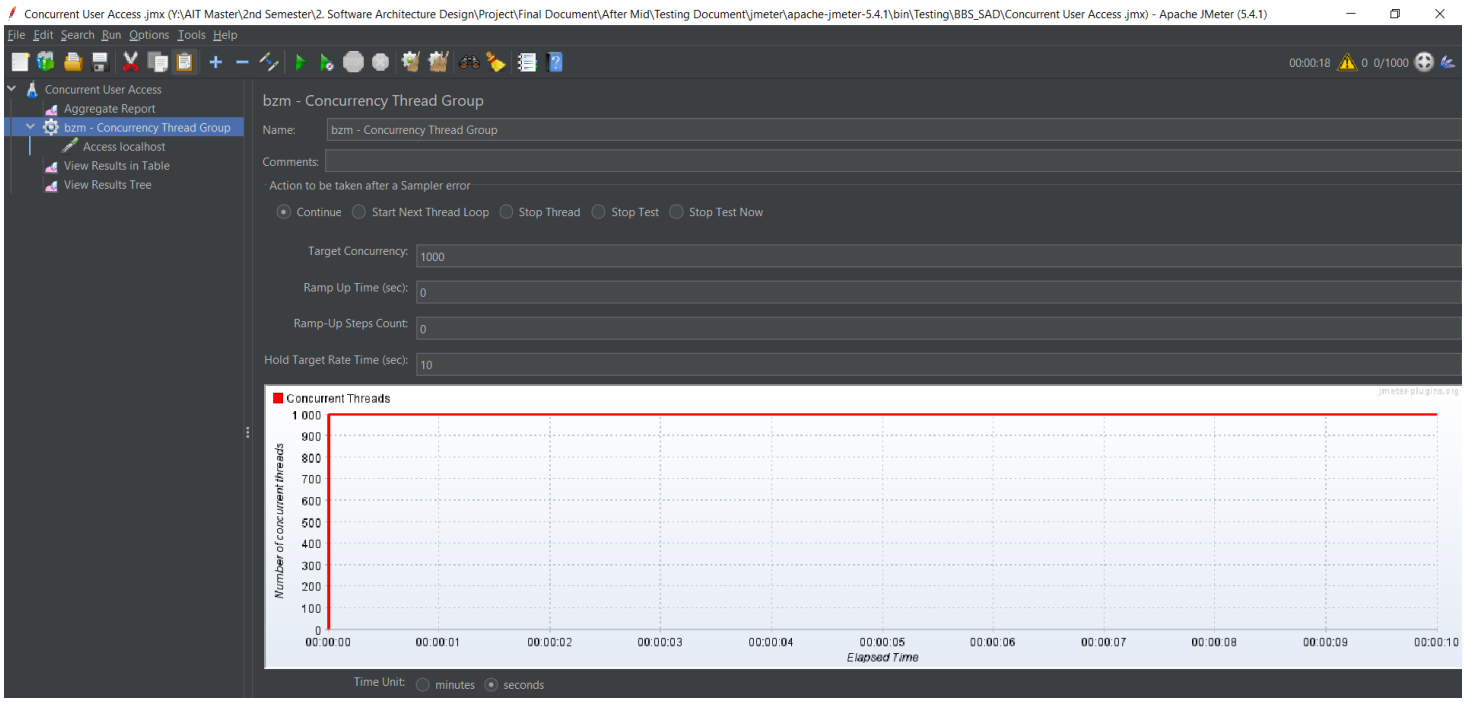
Our application was able to handle the multiple access over 100000.



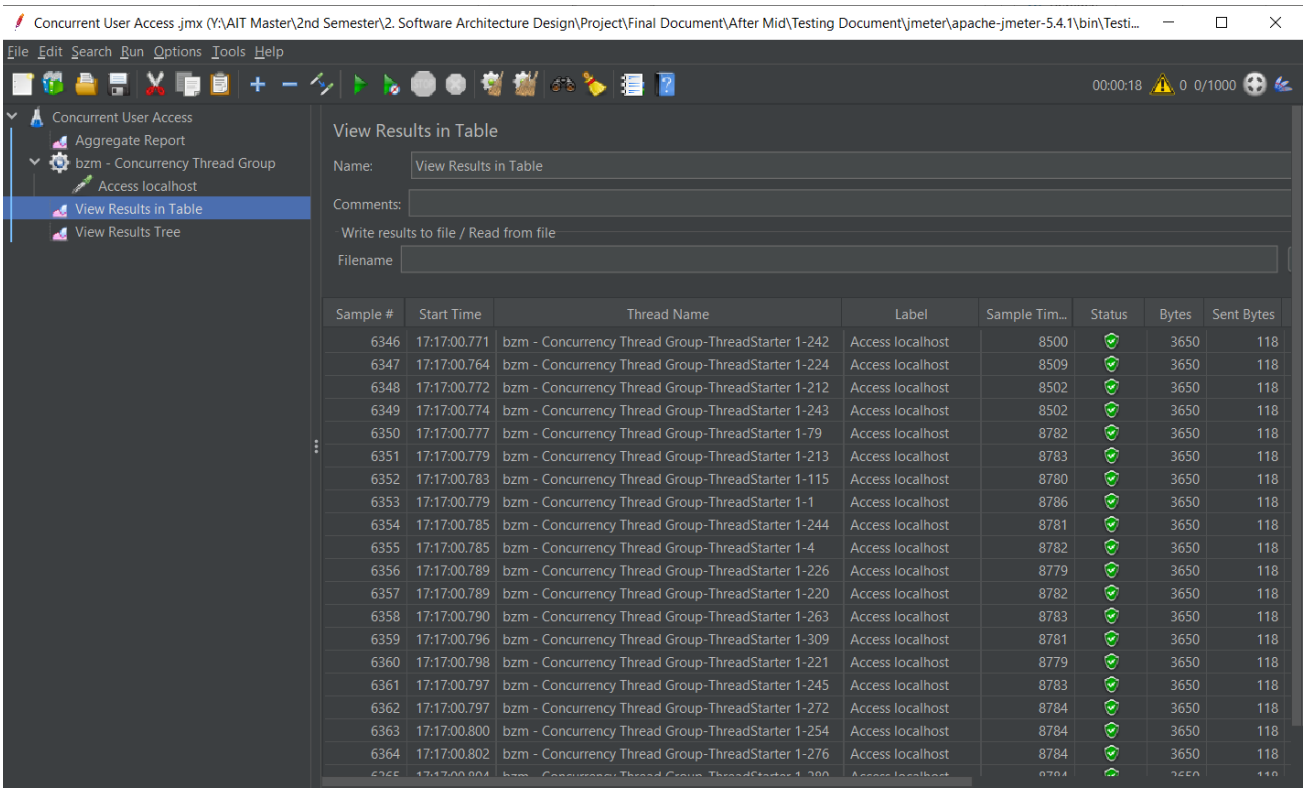
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status
399981	16:45:08.681	100000 Users 1-98691	HTTP Request	3856	
399982	16:45:09.051	100000 Users 1-98498	HTTP Request	3488	
399983	16:45:09.647	100000 Users 1-99251	HTTP Request	2893	
399984	16:45:10.633	100000 Users 1-99274	HTTP Request	1909	
399985	16:45:08.649	100000 Users 1-98501	HTTP Request	3897	
399986	16:45:10.643	100000 Users 1-99252	HTTP Request	1904	
399987	16:45:11.098	100000 Users 1-99255	HTTP Request	1453	
399988	16:45:11.041	100000 Users 1-99272	HTTP Request	1511	
399989	16:45:11.028	100000 Users 1-99673	HTTP Request	1526	
399990	16:45:11.111	100000 Users 1-99254	HTTP Request	1444	
399991	16:45:11.044	100000 Users 1-98892	HTTP Request	1512	
399992	16:45:11.121	100000 Users 1-99243	HTTP Request	1440	
399993	16:45:11.128	100000 Users 1-99276	HTTP Request	1435	
399994	16:45:11.491	100000 Users 1-99242	HTTP Request	1077	
399995	16:45:11.502	100000 Users 1-99253	HTTP Request	1067	
399996	16:45:11.517	100000 Users 1-99256	HTTP Request	1052	
399997	16:45:11.525	100000 Users 1-99300	HTTP Request	1047	
399998	16:45:11.552	100000 Users 1-99660	HTTP Request	1023	
399999	16:45:11.573	100000 Users 1-99235	HTTP Request	1003	
400000	16:45:11.583	100000 Users 1-99275	HTTP Request	994	

5.1.2 Test Concurrent access

We configure 1000 group threads in 10 second, making it concurrent threads or users access the localhost.



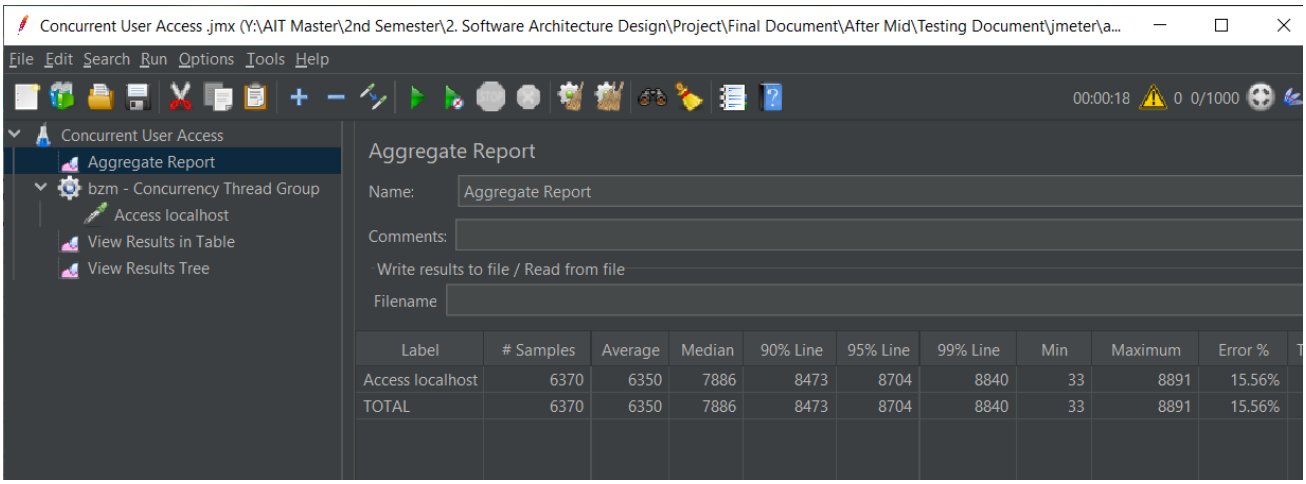
Our application was able to handle the more than 1000 concurrent access.



Sample #	Start Time	Thread Name	Label	Sample Tim...	Status	Bytes	Sent Bytes
6346	17:17:00.771	bzm - Concurrency Thread Group-ThreadStarter 1-242	Access localhost	8500		3650	118
6347	17:17:00.764	bzm - Concurrency Thread Group-ThreadStarter 1-224	Access localhost	8509		3650	118
6348	17:17:00.772	bzm - Concurrency Thread Group-ThreadStarter 1-212	Access localhost	8502		3650	118
6349	17:17:00.774	bzm - Concurrency Thread Group-ThreadStarter 1-243	Access localhost	8502		3650	118
6350	17:17:00.777	bzm - Concurrency Thread Group-ThreadStarter 1-79	Access localhost	8782		3650	118
6351	17:17:00.779	bzm - Concurrency Thread Group-ThreadStarter 1-213	Access localhost	8783		3650	118
6352	17:17:00.783	bzm - Concurrency Thread Group-ThreadStarter 1-115	Access localhost	8780		3650	118
6353	17:17:00.779	bzm - Concurrency Thread Group-ThreadStarter 1-1	Access localhost	8786		3650	118
6354	17:17:00.785	bzm - Concurrency Thread Group-ThreadStarter 1-244	Access localhost	8781		3650	118
6355	17:17:00.785	bzm - Concurrency Thread Group-ThreadStarter 1-4	Access localhost	8782		3650	118
6356	17:17:00.789	bzm - Concurrency Thread Group-ThreadStarter 1-226	Access localhost	8779		3650	118
6357	17:17:00.789	bzm - Concurrency Thread Group-ThreadStarter 1-220	Access localhost	8782		3650	118
6358	17:17:00.790	bzm - Concurrency Thread Group-ThreadStarter 1-263	Access localhost	8783		3650	118
6359	17:17:00.796	bzm - Concurrency Thread Group-ThreadStarter 1-309	Access localhost	8781		3650	118
6360	17:17:00.798	bzm - Concurrency Thread Group-ThreadStarter 1-221	Access localhost	8779		3650	118
6361	17:17:00.797	bzm - Concurrency Thread Group-ThreadStarter 1-245	Access localhost	8783		3650	118
6362	17:17:00.797	bzm - Concurrency Thread Group-ThreadStarter 1-272	Access localhost	8784		3650	118
6363	17:17:00.800	bzm - Concurrency Thread Group-ThreadStarter 1-254	Access localhost	8784		3650	118
6364	17:17:00.802	bzm - Concurrency Thread Group-ThreadStarter 1-276	Access localhost	8784		3650	118

From aggregate report we can find that there is 15.56% error while the concurrent access with 1000 threads in 10 seconds (hold target rate) but when we have same 1000 threads in 10 minutes, it showed 0% of error.

The hold target rate — the duration which we want to run our tests for (how much time you want to keep adding new users each minute).



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %
Access localhost	6370	6350	7886	8473	8704	8840	33	8891	15.56%
TOTAL	6370	6350	7886	8473	8704	8840	33	8891	15.56%

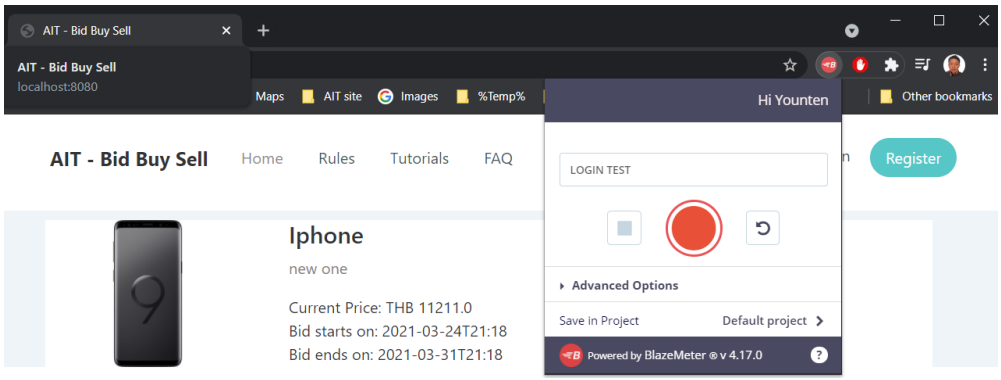
5.1.3 Testing register and login

Record login test in JMeter by following way:

Step 1: add blaze meter plugin to chrome browser.

Step 2: start blaze meter plugin and login to blazemeter

Step 3: Record our scenario – Stop Recording – Export .jmx

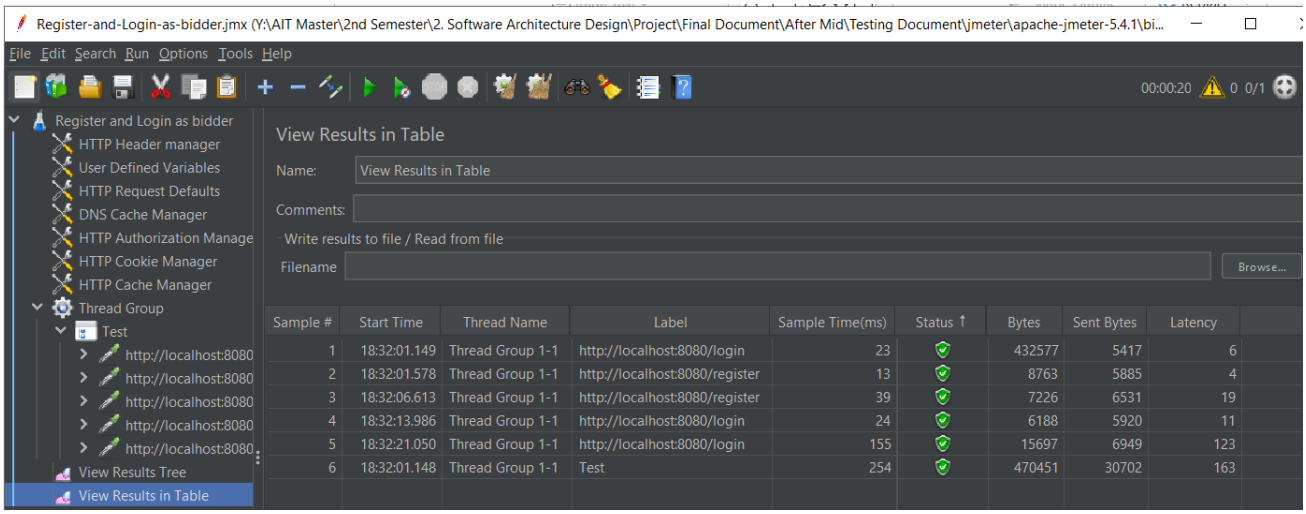


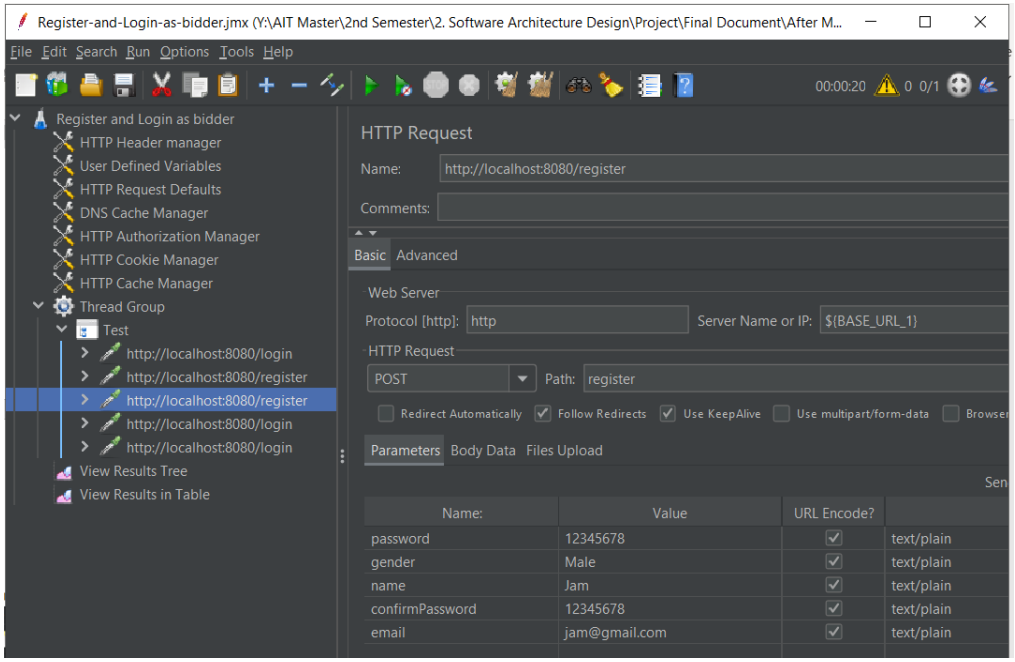
Step 4: Import jmx file in Jmeter

Step 5 : Add Listeners

Step 6 : Run and Validate

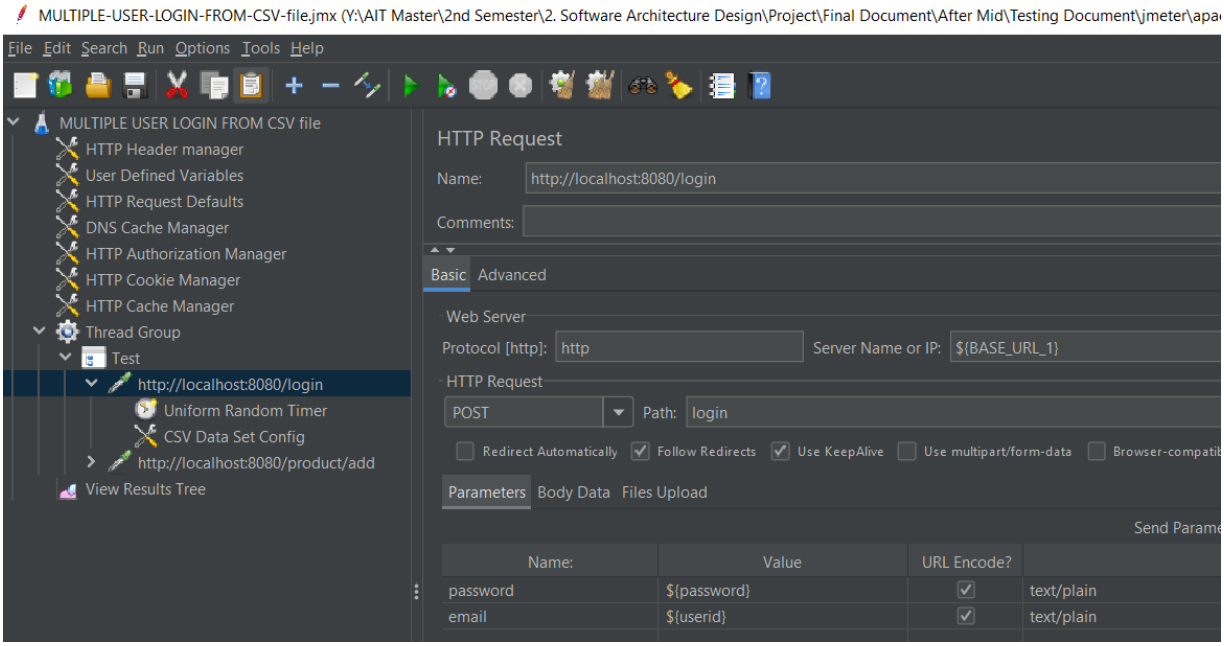
Result of running the record and displayed in table and tree format.



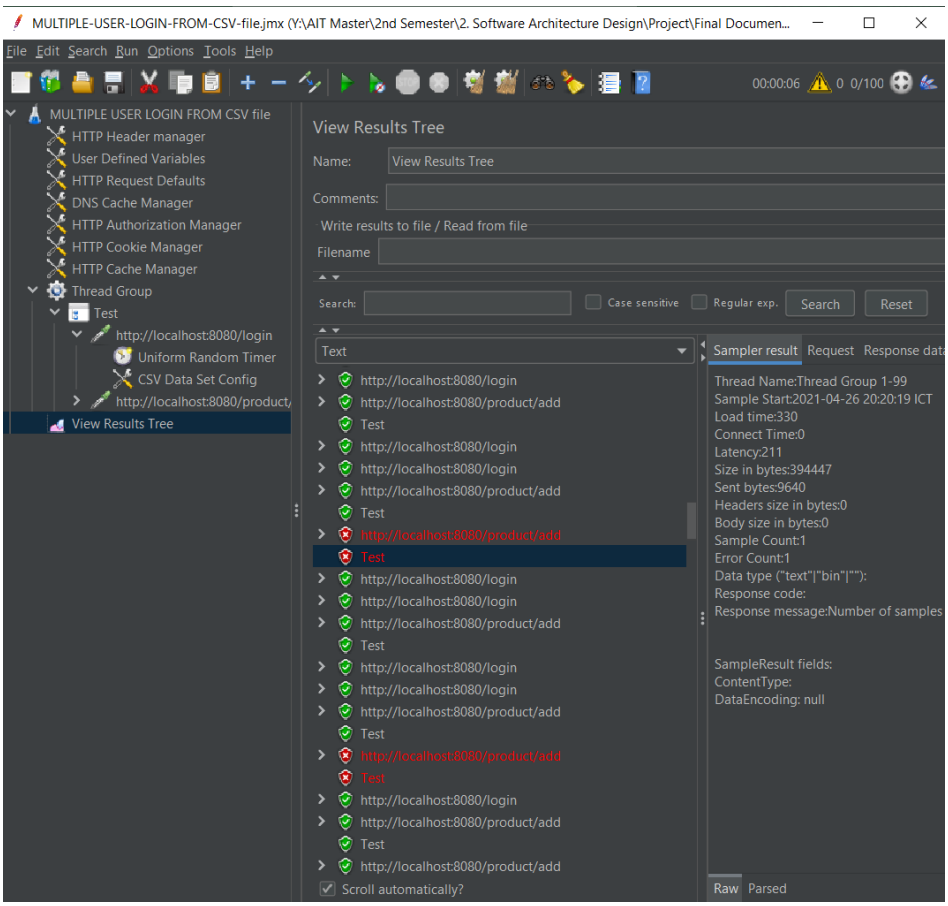


5.1.4 Testing login and other features

We can parametrize our form so that it takes multiple values. For example, in our case, we login multiple users with a userid and password. This feature is particularly useful when our web service has many users. We used csv file to store the userid and password and configures the sampler.



The result was as expected, as we had input some user information wrong and according to it the output is shown.



5.2 User Access Testing and automation framework

Robot Framework is a test automation framework that is Python-based. Test Robot with simple webapp access by writing the script in VS Code with robot file extension and running will show our test case pass or fail. Final output of automation testing is perfect if everything works fine, and we can see the test case being passed or failed.

Test case for checking automation is user will open website, then login into system as buyer or seller, then click add new product and enter details, then add and logout from the system and at the end close the browser after finishing the task.

Following figure shows the test case implemented in robot framework where the test data is in simple and easy-to-edit format. When Robot Framework is started, it processes the test data, executes test cases and generates logs and reports.

```
BBS.robot M ×
Main_Project > BBS.robot > 6. close Browser
You, seconds ago | 1 author (You)
*** Settings ***
Library  Selenium2Library

*** Variables ***
${expect}  LocationMind
${url}  http://localhost:8080/login
${Browser}  chrome
${delay}  1

*** Test Cases ***
1. Open Website
    Open Browser  ${url}  ${Browser}  options=add_experimental_option("excludeSwitches", ["enable-logging"])
    Maximize Browser Window
    Set Selenium Speed   0.3
2. Input username and password
    Input Text  name=email  st121775@ait.asia
    Input Text  name=password  12345678

3. Login
    Click Button  name=submit

4. Check page info
    Click Link    //a[contains(text(),'Add New Product')]
    Input Text  name=name  Iphone XR
    Input Text  name=price  5000
    Input Text  name=description  Perfect in condition and used for 7 months
    Input Text  name=startDate_  m04-d28-Y2021TH12M08S00a
    Input Text  name=finishDate_  m04-d29-Y2021TH12M08S00p

    Click Button  name=addBtn

5. logout
    Click Link    //a[contains(text(),'Log Out')]

6. close Browser      You, seconds ago • Uncommitted changes
    Close Browser
```

Output of the test case after running ***robot BBS.robot:***

```
PROBLEMS 17   OUTPUT   DEBUG CONSOLE   TERMINAL
------------------------------------------------------------------------------
PS C:\Users\Younten Tshering\Desktop\Bid-Buy-Sell-Project\Main_Project> robot .\BBS.robot
==============================================================================
BBS
==============================================================================
1. Open Website                                                       | PASS |
------------------------------------------------------------------------------
2. Input username and password                                        | PASS |
------------------------------------------------------------------------------
3. Login                                                              | PASS |
------------------------------------------------------------------------------
4. Check page info                                                    | PASS |
------------------------------------------------------------------------------
5. logout                                                             | PASS |
------------------------------------------------------------------------------
6. close Browser                                                      | PASS |
------------------------------------------------------------------------------
BBS                                                                   | PASS |
6 tests, 6 passed, 0 failed
==============================================================================
```

All the test cases did pass and shows that our application's adding product work properly or integration of interface is complete.

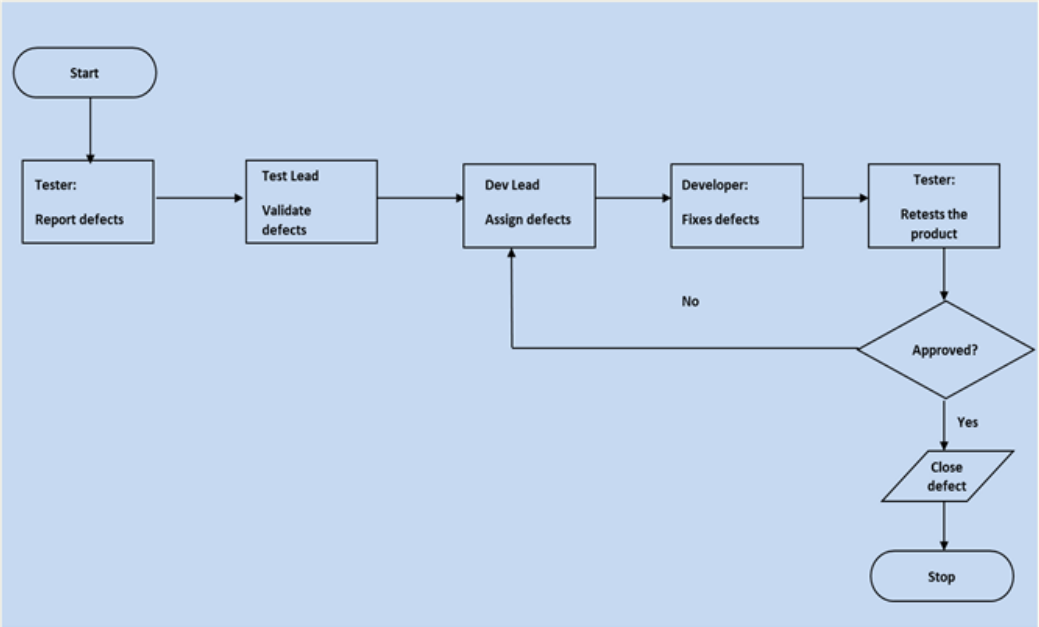
6. Release Control

6.1 Defect tracking & Reporting

A software defect is an error in coding which causes incorrect or unexpected results from a software program which does not meet actual requirements. Testers might come across such defects while executing the test cases.

Therefore, for our project we have assign one member as tester and other as developer. After the defects are identified, the defects are explained to developers so that it is fixed and tested again.

Following flowchart depicts Defect Tracking Process:



7. Risk Analysis

List all risks that we envision and plan to mitigate these risks and a contingency plan in case if we see these risks in reality.

Test Risks and Mitigation Factors

Risk	Prob.	Impact	Mitigation Plan
SCHEDULE Testing schedule is tight. If the start of the testing is delayed due to design tasks, the test cannot be extended beyond the UAT scheduled start date.	High	High	The testing team can control the preparation tasks (in advance) and the early communication with teammate.
DEFECTS Defects are found at a late stage of the cycle or at a late cycle; defects discovered late are most likely be due to unclear specifications and are time consuming to resolve.	Medium	High	Defect management plan was developed which our team made to meet physically since all member were together to ensure prompt communication and fixing of issues.
Delayed Testing Due To New Issues	Medium	High	During testing, there is a good chance that some "new" defects may be identified and may become an issue that will take time to resolve. There are defects that can be raised during testing because of unclear document specification. These defects can yield to an issue that will need time to be resolved. If these issues become showstoppers, it will greatly impact on the overall project schedule. If new defects are discovered, the defect management is placed to immediately provide a resolution.

8. Conclusion

Test Strategy is not a piece of paper. It is the reflection of whole Quality Assurance activities in the software testing life cycle. Referring this kind of document time to time in the test execution process and follow the plan till the software release.

When the project nears the release date it is fairly easy to cut on testing activities by ignoring what we have defined in the test strategy document. But it is advisable to discuss with our team whether or not cutting down on any particular activity will help for release without any potential risk of major issues post-release.

Our team focus is on test execution with proper documentation. But having a basic test strategy plan always helps to clearly plan and mitigate risks involved in the project. Our teams will capture and document all high-level activities to complete test execution on time without any issues.

9. Appendix

9.1 Simple Tips to Write Test Strategy Document

1. Include product background in the test strategy document. In the first paragraph of your test strategy document answer – Why stakeholders want to develop this project? This will help to understand and prioritize things quickly.
2. List all important features you are going to test. If you think some features are not part of this release, then mention those features under "Features not to be tested" label.
3. Write down the test approach for your project. Clearly, mention what types of testing you are going to conduct?
   i.e., Functional testing, UI testing, Integration testing, Load/Stress testing, etc.
4. Answer questions like how you are going to perform functional testing? Manual or automation testing? Are you going to execute all test cases from your test management tool?
5. Which bug tracking tool you are going to use? What will be the process when you will find a new bug?
6. What is your test entry and exit criteria?
7. How will you track your testing progress? What metrics are you going to use for tracking test completion?
8. What documents will you produce during and after the testing phase?
9. What risks do you see in Test completion?

9.2 Setup Robot Framework

9.2.1 JDK 1.8-1

The Java Development Kit (JDK) allows you to code and run Java programs.

Following are the steps to be followed for downloading and installing jdk in our system:

1. Goto https://www.oracle.com/in/java/technologies/javase-downloads.html click on *Download* JDK for latest version.
2. Next, Accept the license Agreement.
3. Click on download latest Java JDK depending on version of our PC.
4. Once Download is completed, then run the .exe for installation of JDK. Then click Next.
5. Select Proper PATH for Java installation and click next.
6. Once installation is complete then click close.

9.2.2 Python

Before downloading python, see what version is to be installed, the follow the steps:

1. Open web browser: https://www.python.org/downloads/release/python-2717/. Search for desired version and select a link download either Windows x86-64 executable installer or Windows x86 executable installer.
2. Run the executable Installer. Make sure you select the Install launcher for all users and Add Python to PATH checkbox. Then select Install Now.
3. Verify Python was installed or not by typing python -V in command Prompt.

Before downloading, make sure your system has PIP installed.

You can follow this link https://phoenixnap.com/kb/install-pip-windows for pip.

```
Administrator: Command Prompt
C:\WINDOWS\system32>python.exe -m pip install --upgrade pip
Collecting pip
  Downloading pip-21.0.1-py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 1.3 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
    Uninstalling pip-20.2.3:
      Successfully uninstalled pip-20.2.3
Successfully installed pip-21.0.1

C:\WINDOWS\system32>robot --version
Robot Framework 4.0.1 (Python 3.9.4 on win32)

C:\WINDOWS\system32>Type: pip
The filename, directory name, or volume label syntax is incorrect.

C:\WINDOWS\system32>pip install robotframework-selenium2library
Collecting robotframework-selenium2library
  Downloading robotframework_selenium2library-3.0.0-py2.py3-none-any.whl (6.2 kB)
Collecting robotframework-seleniumlibrary>=3.0.0
  Downloading robotframework_seleniumlibrary-5.1.3-py2.py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 401 kB/s
Requirement already satisfied: robotframework>=3.1.2 in c:\users\younten tshering\appdata\local\programs\python\python39
\lib\site-packages (from robotframework-seleniumlibrary>=3.0.0->robotframework-selenium2library) (4.0.1)
Collecting selenium>=3.141.0
  Downloading selenium-3.141.0-py2.py3-none-any.whl (904 kB)
     |████████████████████████████████| 904 kB 1.7 MB/s
Collecting robotframework-pythonlibcore>=2.1.0
  Downloading robotframework_pythonlibcore-2.2.1-py2.py3-none-any.whl (10 kB)
Collecting urllib3
  Downloading urllib3-1.26.4-py2.py3-none-any.whl (153 kB)
     |████████████████████████████████| 153 kB 2.2 MB/s
Installing collected packages: urllib3, selenium, robotframework-pythonlibcore, robotframework-seleniumlibrary, robotfra
mework-selenium2library
Successfully installed robotframework-pythonlibcore-2.2.1 robotframework-selenium2library-3.0.0 robotframework-seleniuml
ibrary-5.1.3 selenium-3.141.0 urllib3-1.26.4

C:\WINDOWS\system32>pip list
Package                       Version
----------------------------- -------
pip                           21.0.1
robotframework                4.0.1
robotframework-pythonlibcore  2.2.1
robotframework-selenium2library 3.0.0
robotframework-seleniumlibrary  5.1.3
selenium                      3.141.0
setuptools                    49.2.1
urllib3                       1.26.4
```

9.2.3 Robot Framework Library

- Once python is installed, you have to access to the pip installer.
- Open command prompt and type

pip install robotframework

9.2.4 Robot Framework Selenium Library

Use the following command in command prompt to install robot framework selenium library.

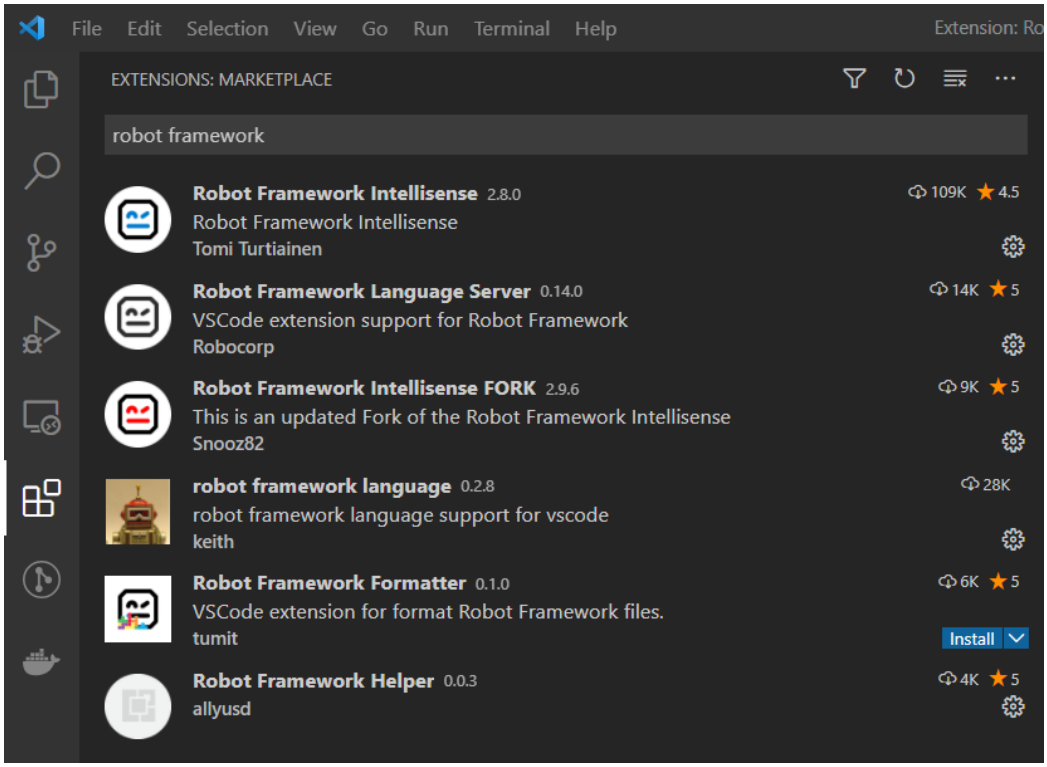
pip install robotframework-seleniumlibrary

9.2.5 Develop Test Script

Before having the script, we need to install VS Code for Develop Test Script.

Steps to install VS Code and configure:

1. Download Visual Studio Code
2. Run the execution file and make it run.
3. We must add some few extension as shown in below diagram:



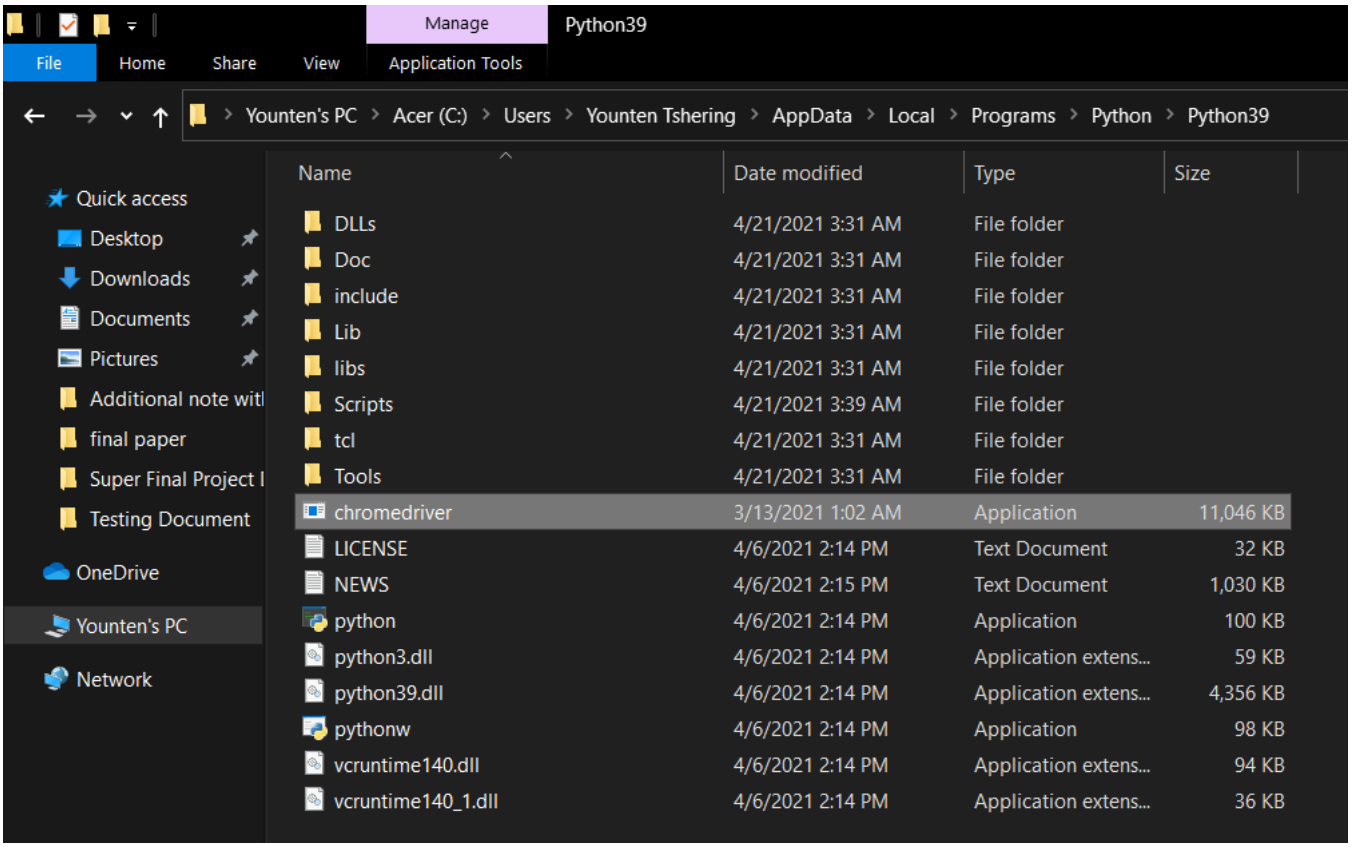
9.2.6 Chrome Driver

Steps for chrome driver installation:

1. Check Version of Google Chrome
2. Download Chrome Driver using this link:

   https://sites.google.com/a/chromium.org/chromedriver/

3. Click Latest release, then click on the driver zip file as per the version of your system to download chromedriver eg: chromedriver_win32.zip
4. Unzip/ extract zip file in python folder as shown below:



5. Run executable exe file.
6. Make sure that system environments variables are properly configured.

   For more details or complete guide, follow the link:

   https://www.youtube.com/watch?v=gleaQkECggo.

9.2.7 Try to write test script and run test script.

Before running the script, we need to complete the install of chrome driver and then:

1. Create sample Robot Script for testing
2. Create variables
3. Create functions
4. Testing Step: Test cases

For more information follow the link: https://medium.com/edureka/robot-framework-tutorial-f8a75ab23cfd.

Test Robot with simple website access:

Write the script in VS Code with robot file extension and run. The example shown is the file uploaded by professor in basecamp, but project testing is included in the project documentation.

```
sdqi.robot ×
C: > Users > Younten Tshering > Downloads > sdqi.robot > ...
*** Settings ***
Library  Selenium2Library

*** Variables ***
${expect}  LocationMind
${url}  http://lmwebmap.gisserv.com
${Browser}  chrome
${delay}  1

*** Test Cases ***
1. Open Website
    Open Browser  ${url}  ${Browser}  options=add_experimental_option("excludeSwitches", ["enable-logging"])
    Maximize Browser Window
    Set Selenium Speed   0.3
2. Input username and password
   Input Text  name=usernameTextBox  tester
   Input Text  name=passwordTextBox  tester
3. Login
   Click Button  name=submitButton
4. Check page info
   Click Link    xpath=(//a[@href="#"])[2]
   ${result}  Get Text  xpath=(//div)[8]
   Log To Console  ${result}
   Should Contain  ${result}  ${expect}

5. close Browser
   Close Browser
```

Output if the chrome driver is not installed:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                   1: powershell

PS C:\Users\Younten Tshering\Downloads> robot .\sdqi.robot
==============================================================================
Sdqi
==============================================================================
1. Open Website                                                       | FAIL |
WebDriverException: Message: 'chromedriver' executable needs to be in PATH. Please see https://sites.google.com/a/chromium.org/chromedriver/home
------------------------------------------------------------------------------
2. Input username and password                                        | FAIL |
No browser is open.
------------------------------------------------------------------------------
3. Login                                                              | FAIL |
No browser is open.
------------------------------------------------------------------------------
4. Check page info                                                    | FAIL |
No browser is open.
------------------------------------------------------------------------------
5. close Browser                                                      | PASS |
------------------------------------------------------------------------------
Sdqi                                                                  | FAIL |
5 tests, 1 passed, 4 failed
==============================================================================
Output:  C:\Users\Younten Tshering\Downloads\output.xml
Log:     C:\Users\Younten Tshering\Downloads\log.html
Report:  C:\Users\Younten Tshering\Downloads\report.html
PS C:\Users\Younten Tshering\Downloads>
```

Final output of automation testing if everything works fine and we can see the test case being passed or failed.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                                                    1: powersh

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Younten Tshering> cd .\Downloads\
PS C:\Users\Younten Tshering\Downloads> robot .\sdqi.robot
==============================================================================
Sdqi
==============================================================================
1. Open Website                                                       | PASS |
------------------------------------------------------------------------------
2. Input username and password   Follow link (ctrl + click)           | PASS |
------------------------------------------------------------------------------
3. Login                                                              | PASS |
------------------------------------------------------------------------------
4. Check page info                                                    ..Copyright © 2020 LocationMind Inc. All Rights Reserved.
4. Check page info                                                    | PASS |
------------------------------------------------------------------------------
5. close Browser                                                      | PASS |
------------------------------------------------------------------------------
Sdqi                                                                  | PASS |
5 tests, 5 passed, 0 failed
==============================================================================
Output:  C:\Users\Younten Tshering\Downloads\output.xml
Log:     C:\Users\Younten Tshering\Downloads\log.html
```

9.3 Setup JMeter

JMeter for the purpose of load testing is considered as the one of the best tools. Apache JMeter is a Java open-source GUI-based software that is used as a performance testing tool for analyzing and measuring the performance of our application.

JMeter is extensible, meaning that we can write custom script, using the pre-built interface of JMeter, making it even more powerful. Of course, it already has many powerful features such as

a) creating reusable test plans (in XML format) so we can reuse these similar test plans across all our products,

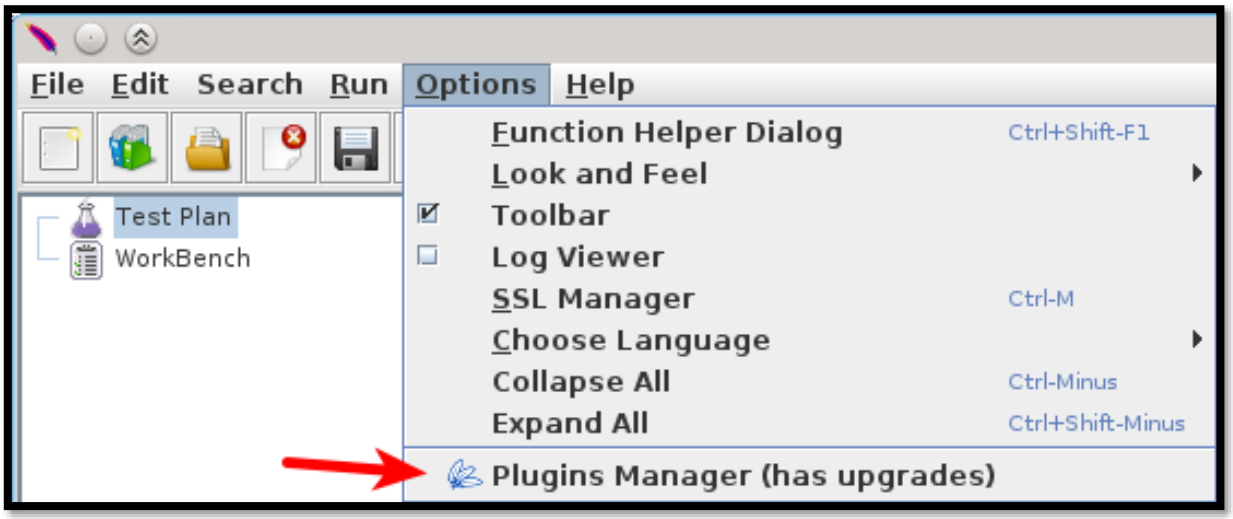
b) supporting various protocols such as HTTP, JDBC, SOAP, JMS, and FTP,

c) supporting various testing such as stress testing, distributed testing, web service testing, allowing user to record HTTP and HTTPS to create test plan,

d) having loads of plugins which we can installed via Option > Plugin Manager and support dashboard report generation.

9.3.1 Requirements

- Make sure you got java installed (Requires Java 8+)
- Install JMeter via http://jmeter.apache.org
- Download the Plugins Manager JAR file and put it into JMeter's lib/ext directory. Then start JMeter and go to "Options" menu to access the Plugins Manager.



9.3.2 JMeter - overview of how to use.

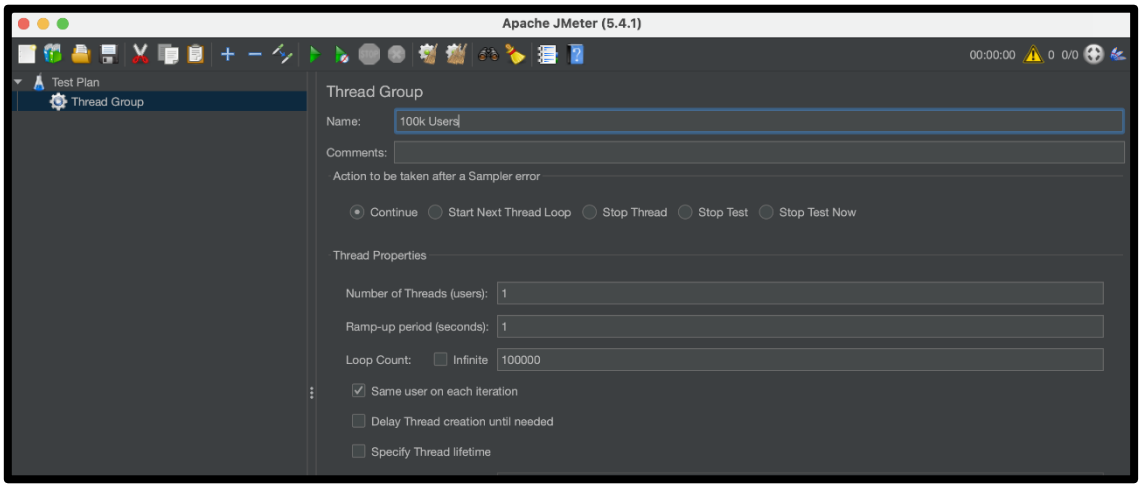
JMeter is conceptualized around a concept called "Test Plan". To create a test plan which is the central execution unit of a test, we first need to understand the four important elements of a test plan:

1. Thread Group
2. Samplers
3. Listeners
4. Configuration

Thread group

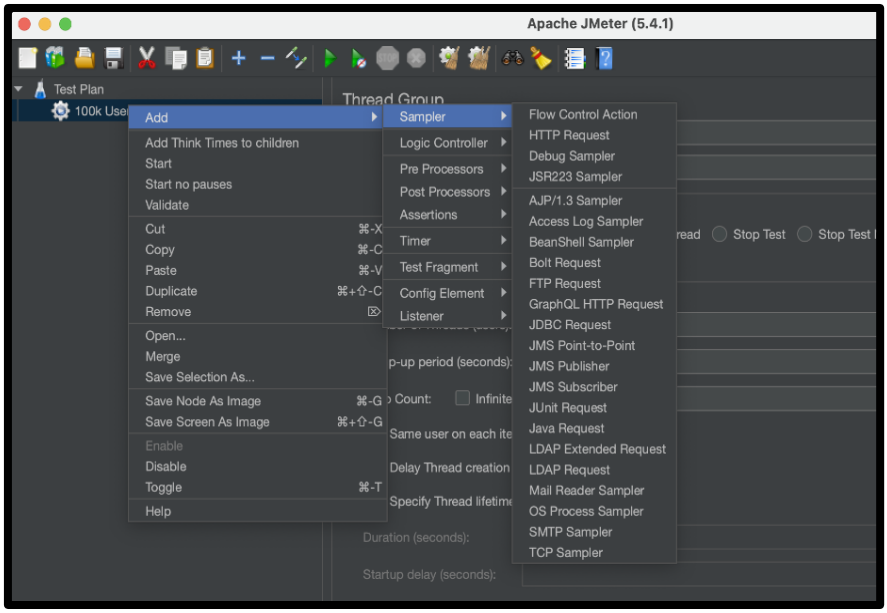
- Represents a collection of threads. Each thread represents one user. Basically, in a test plan, we must specify how many users you want. If you set 100 threads, it simply means you have 100 users.
- Note that by default, these threads are not concurrent. If you would like to use concurrent threads, go to Option > Plugin Managers, and install Custom Thread Group which allows for testing concurrent threads.
- In the JMeter interface, under Test Plan, the first thing you want to do is create a thread group by right clicking like the following:

- Under the threads group, we can specify 100k users in the Name and Loop Count (100000) fields like this:



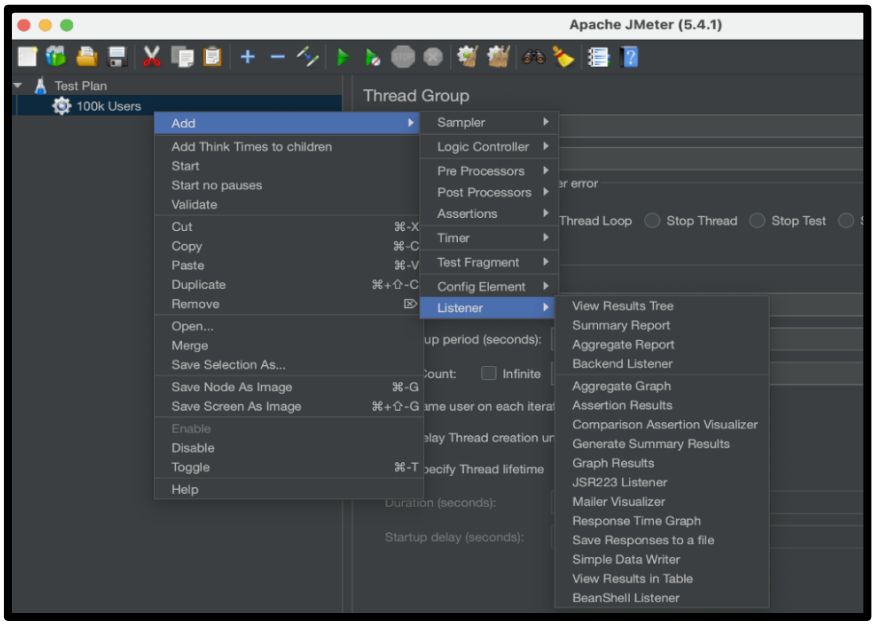
Samplers

- As we have mentioned already, JMeter supports testing several protocols such as HTTP, FTP, JDBC, and many more. For example, if we want to send a download request to a server, then we can use FTP requests for our thread groups; if we want to check how a certain query will run in our database, use JDBC request for our thread groups (before we can use, we need to set the connection configuration - see Configuration); if we want to check how our web service works, use HTTP request. If we want to check how our email will work, use SMTP Sampler.
- Under the Thread Group we have just created, we can right click, select Sampler, and we will see many possible protocols we can test.
- We shall look at specifically HTTP requests and JDBC requests later in our short demo section.



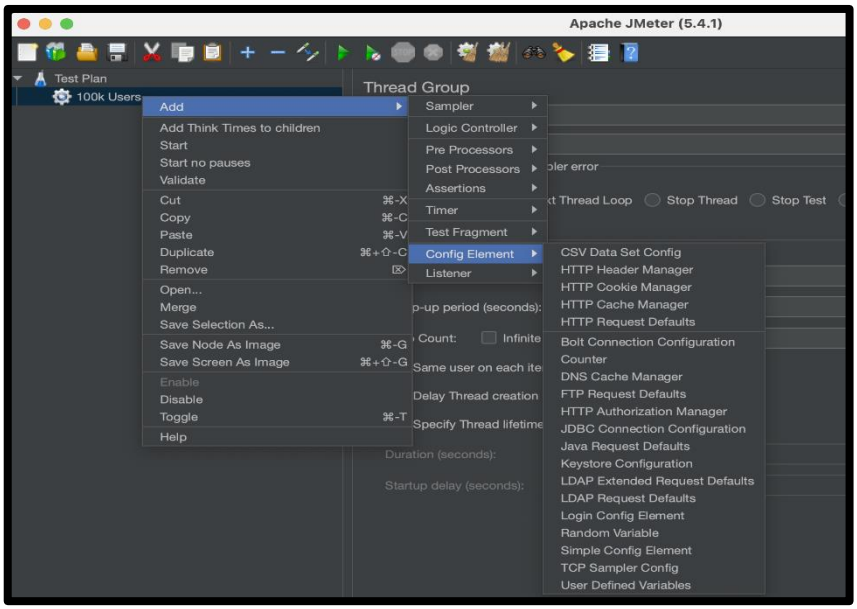
Listeners

- Once we set the Thread Group and the associated Samplers, we want to see the results and that's the Listeners.
- JMeter supports many nice visualizations in (1) graph, (2) table, (3) tree, and (4) text format.



Configuration

The last element to set for Thread Group is Configuration, which is about setting up some variables such as database connection. For example, for CSV Data set Config, we can put in any variable we want in a CSV format which will be automatically parsed and can be used when we script. HTTP Cookie will store all cookies of a particular website for future requests. HTTP Request Defaults set some default variables such as domain; so instead of typing http://google.com as our target 100 times, we can just config one time here. Login Config Element allows us to set up the username and password in a user request.



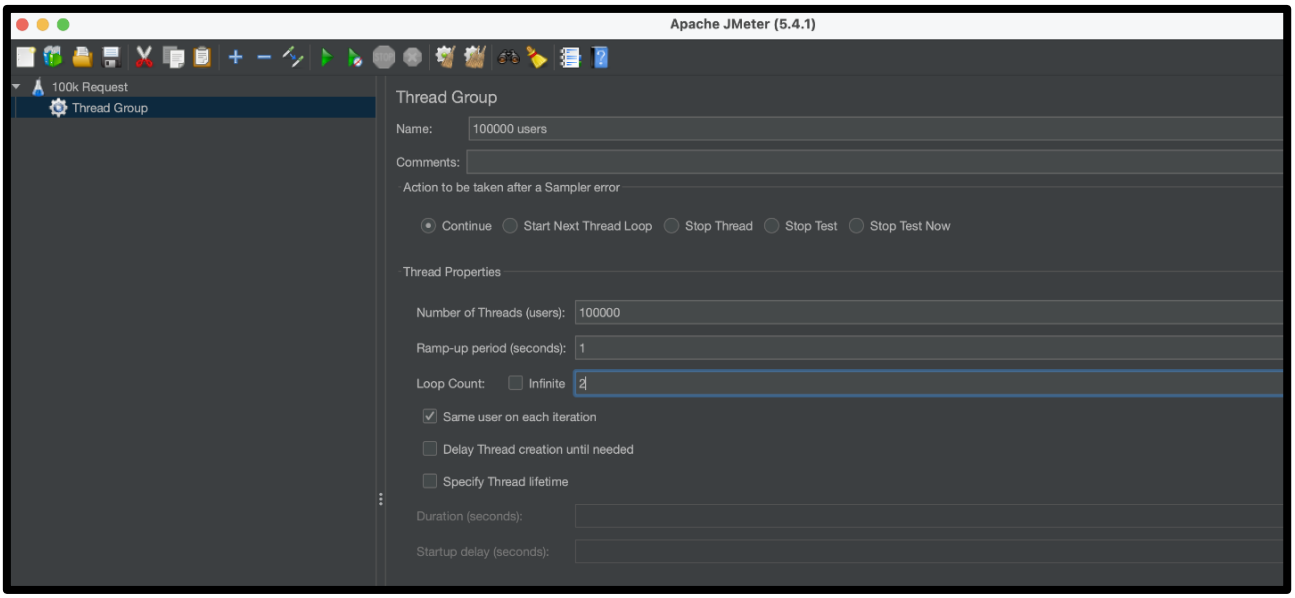
9.3.3 Testing Demo

Demo 1: Simple HTTP Request

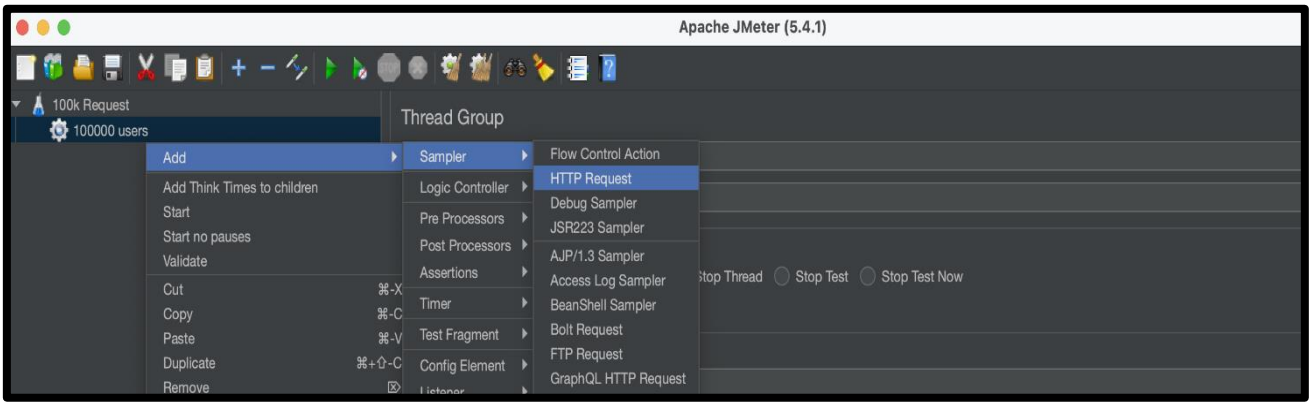
Very basic HTTP Request load testing

1. First let's name the Test Plan as 100k Request

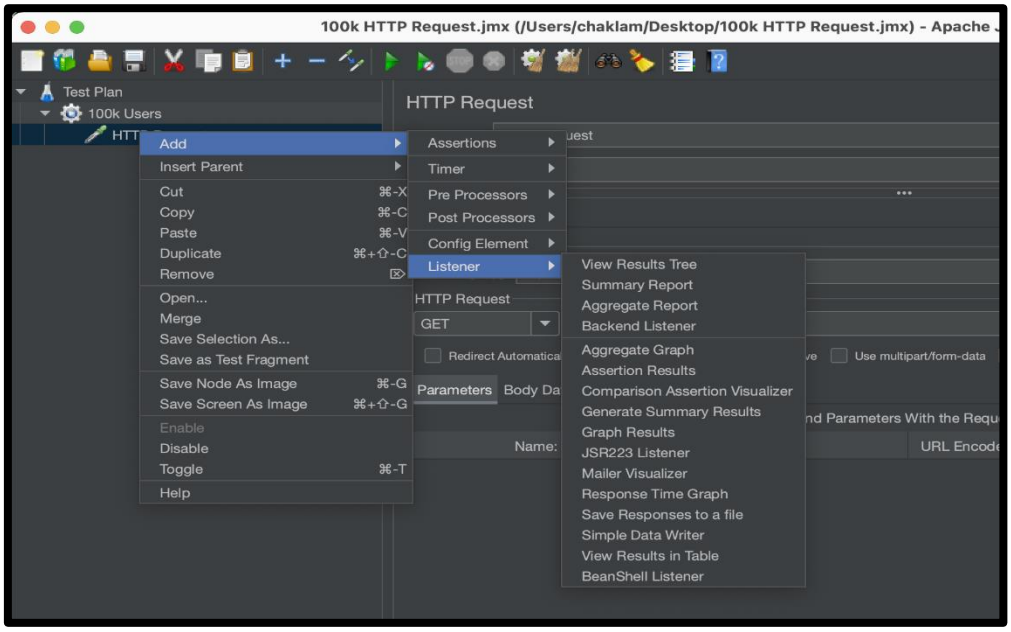
2. Right click the Test Plan and add Thread Group. In the Thread Group, put any Name, and specify the Number of threads and Loops.



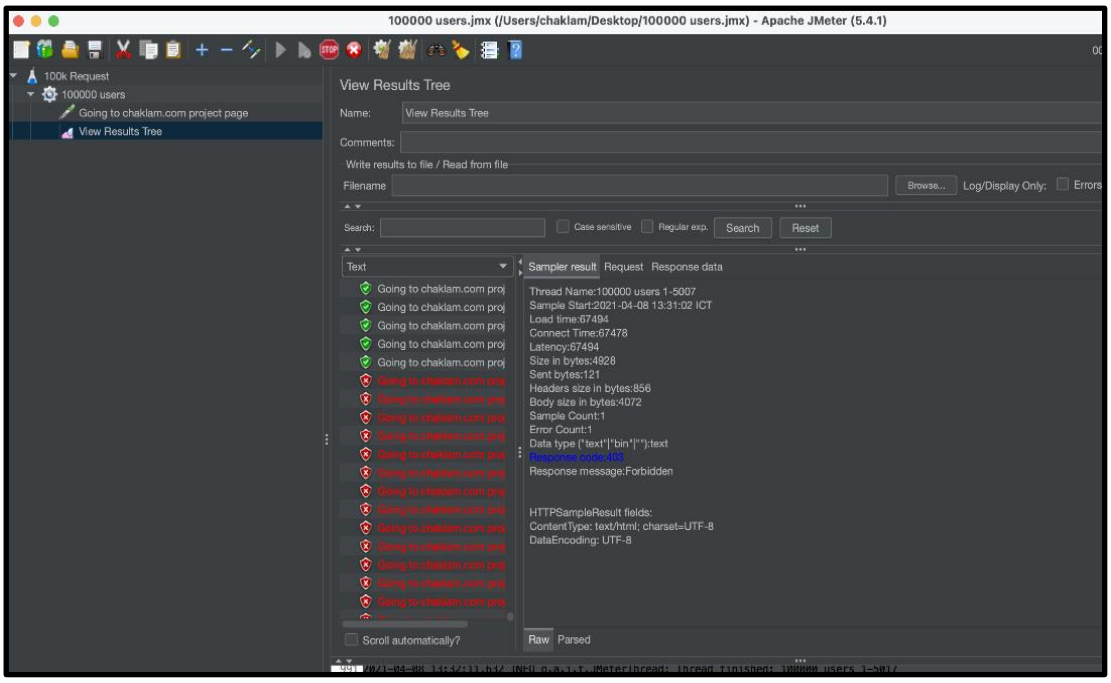
3. Once we have the Thread Group, we specify what are the testing protocols we will be using. Right click on the Thread Group and add Sampler > HTTP Request.



4. Specify the HTTP Request parameters as
    a. Protocol: HTTP
    b. Server name: localhost
    c. Port: 8080
    d. Path: /projects
5. Next, before we actually run the test, let's add some visualization so we can better understand our result. Right click on the Thread Group and add Listener > View Result Tree



6. Before you click the Play (Green Triangle) button, the system will ask us to save. Save it so that we can reuse this whole Test Plan for our other project. Here is the result of my test result. As we can see, website can hardly handle 100k users as there are way too many reds.



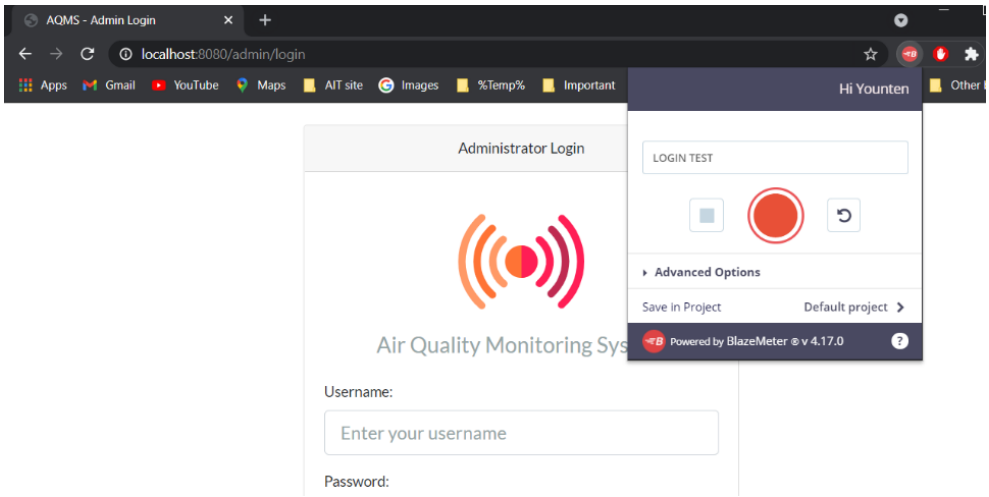
Demo 2: Recording

We can use the Record feature to record all our actions so we can replay all our action as a test procedure. ***Record test in JMeter by following way:***

Step 1: add blaze meter plugin to chrome browser.

Step 2: start blaze meter plugin and login to blazemeter

Step 3: Record our scenario – Stop Recording – Export .jmx



Step 4: Import jmx file in Jmeter

Step 5 : Add Listeners

Step 6 : Run and Validate