Performance/Load Testing

Introduction

In this assignment, you will learn what is performance testing, how to use a performance test tool, how to design a performance test, and execute performance tests.
By completing this assignment, you will know how to:

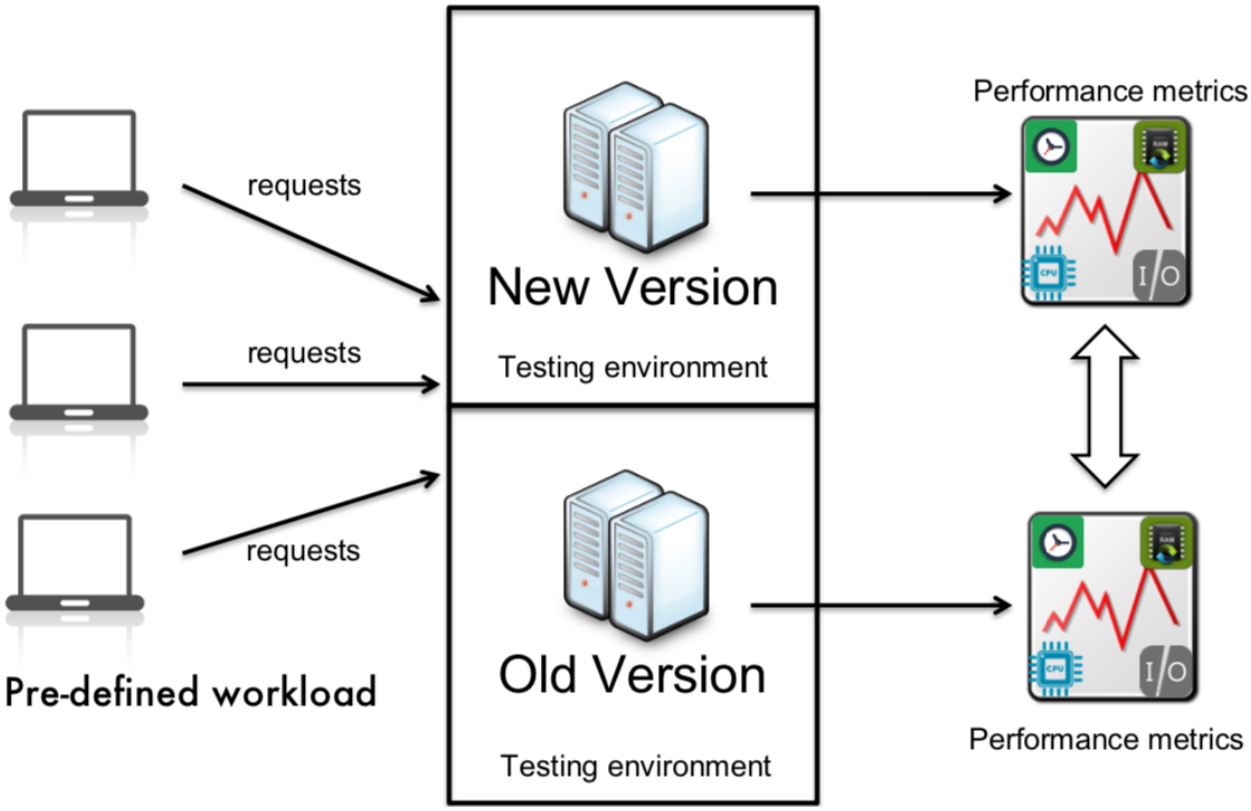
(1) Configure the load driver
(2) Execute the load test
(3) Analyze the results of this load test

Definition

- Performance testing. It is a type of testing for determining the speed of a computer, network, or device. It checks the performance of the components of a system by passing different parameters in different load scenarios.
- Load testing. It is the process that simulates actual user load on any application or website. It checks how the application behaves during normal and high loads. This type of testing is applied when a development project nears its completion.

Mechanism

Performance/Load testing consists of three processes: 1) designing a proper load, 2) executing a load test, and 3) analyzing the results of a load test.
In practice, testers set up the testing environment and define the workload. Then the same requests are sent to exercise the system and the performance metrics are collected. Finally, testers compare the two performance metrics and identify the performance regression. Below is a figure showing the mechanism of performance/load testing:



Tools

There exist various tools that conduct a performance/load test:

- **JMeter**: JMeter is an open-source tool that can be used for performance and load testing for analyzing and measuring the performance of a variety of services.
- **HP LoadRunner**: An enterprise performance testing version of Loadrunner and a platform enabled both global standardization.
- **ReadLine13**: is a load testing platform that brings the low-cost power of the cloud to JMeter and other open-source load testing tools.

In this assignment, we will use **JMeter** for performance/load testing, and **Blazemeter** (google chrome extension), and through the following sections, you will gain more hands-on knowledge and experience about performance/load testing.

JMeter - Features

JMeter is a free open-source load driver from the Apache foundation. JMeter is a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. It can be used to simulate a heavy load on a server, group of servers, network, or object to test its strength or to analyze overall performance under different load types.
https://jmeter.apache.org/usermanual

Apache JMeter features include load and performance testing many different applications/server/protocol types:

- Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
- SOAP / REST Web Services
- FTP
- Database via JDBC
- LDAP
- Message-oriented middleware (MOM) via JMS
- Mail - SMTP(S), POP3(S) and IMAP(S)
- Native commands or shell scripts
- TCP
- Java Objects

Requirements

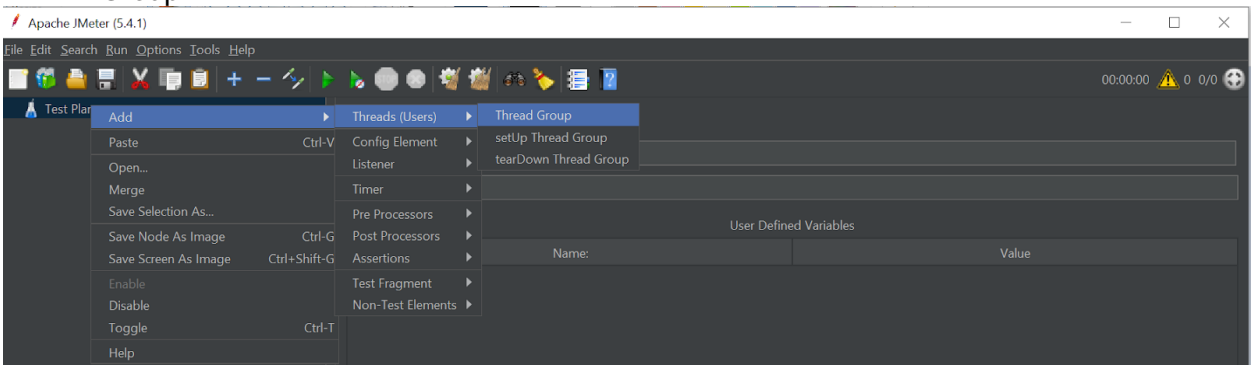
Please make sure that you have JDK, Apache JMeter, and Chrome installed on your computer. Disclaimer: The environment below was used to prepare this assignment. We cannot guarantee that different versions than the ones stated below will work.

- Windows 10 x64
- Java SDK 15
- Apache JMeter 5.4.1
- Google Chrome 99.x
- BlazeMeter Chrome extension 5.10

Experiments

Part A

1. Download and extract JMeter.
2. Open JMeter by double clicking on ApacheJMeter.jar (jmeter.sh in *NIX systems).
3. Add a new thread group by right-clicking on Test Plan->Add-> Thread (users)->Thread Group.



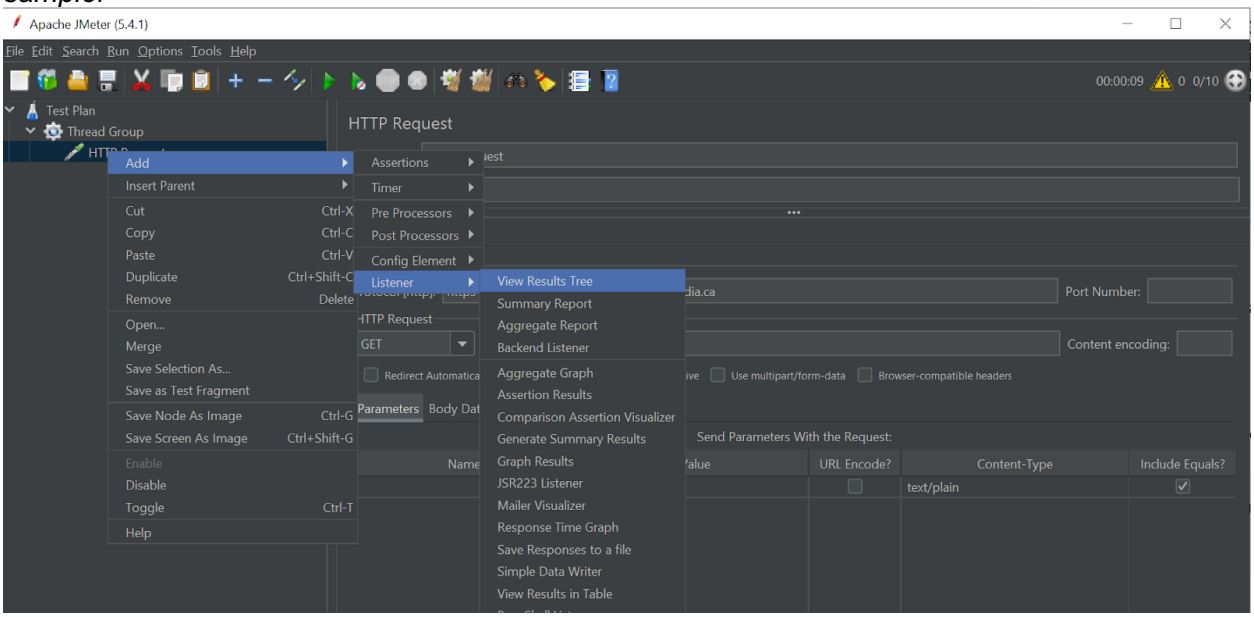
4. Name the Thread group as *Users*

5. In *Thread properties* set the number of threads to *10.*

6. Now we are going to create the sampler to indicate which website to test. Right click on the thread group *Users*->Add->Sampler->HTTP Request

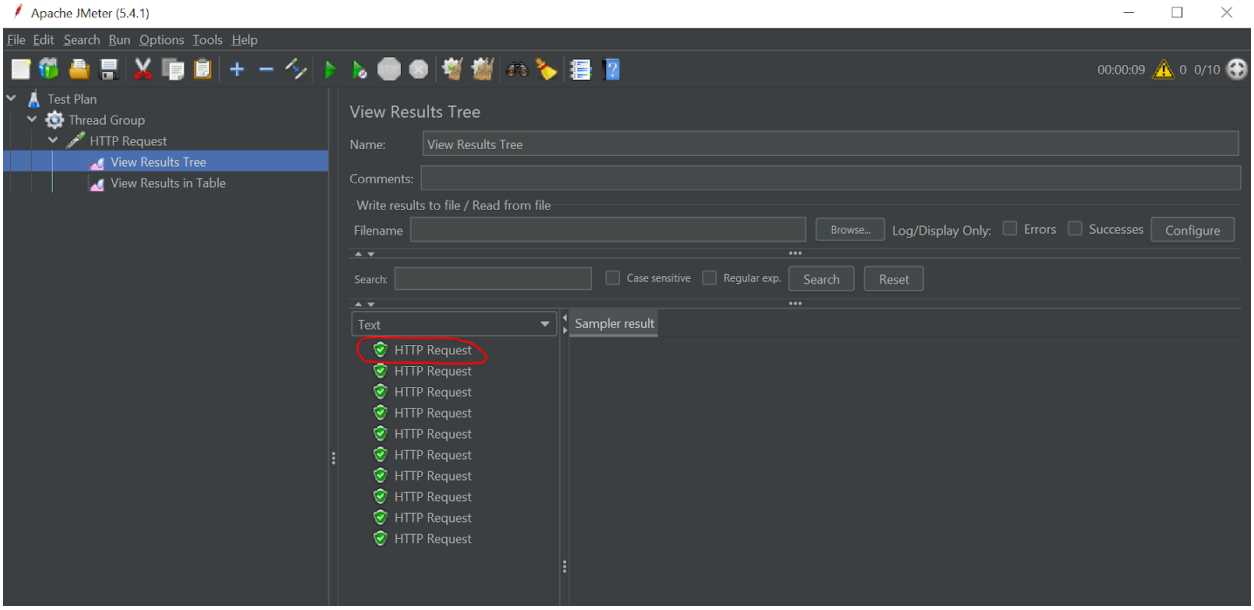
7. In the web server textbox, select **https** as *protocol* and *Server Name:* **www.concordia.ca**

8. Add a ***View Results Tree*** listener by right-clicking on the HTTP Request *sampler*

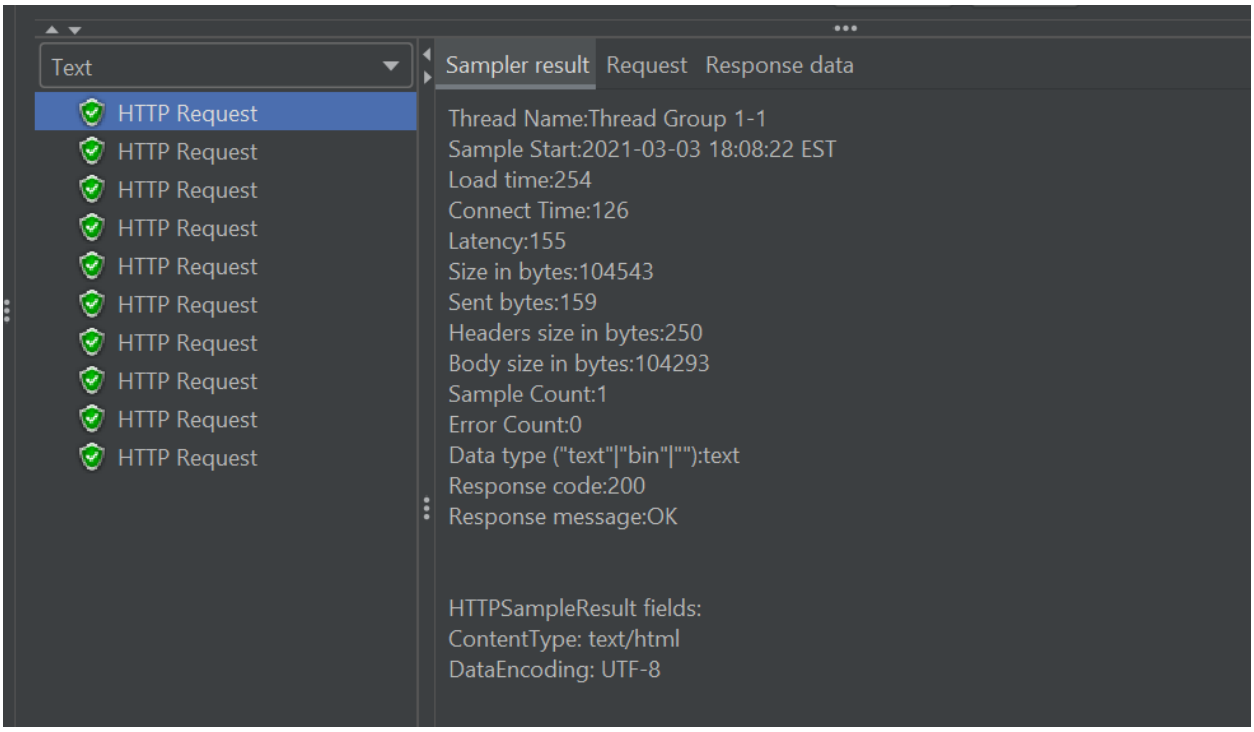


9. Add a ***View Results in Table*** *listener* by right-clicking on the HTTP Request *sampler.* JMeter will render the results in two different views by adding the listeners.

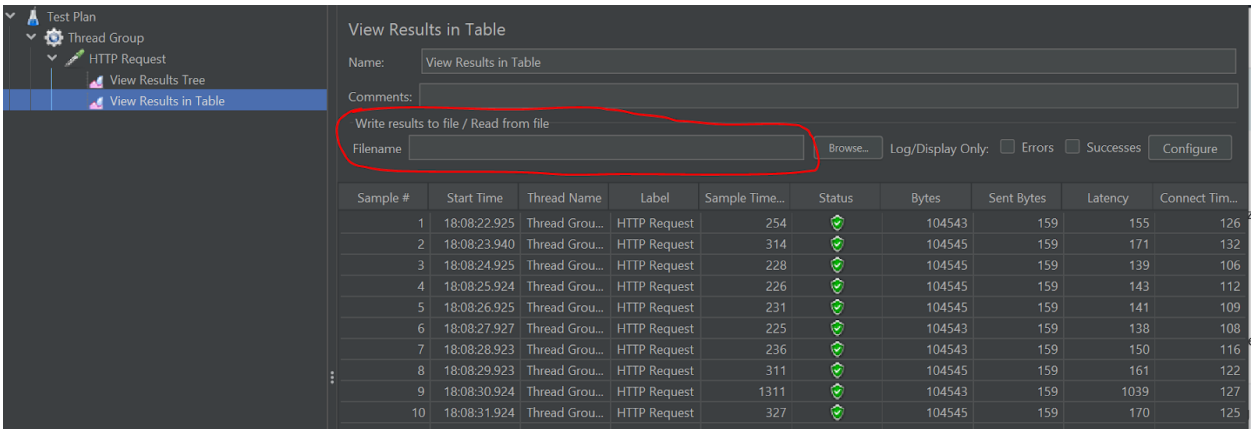
10. Execute the test by left-clicking on the *play* button (green triangle). The green shield indicates the successful execution of the *HTTP Request. You might be prompted to* Save your test plan. Save it as **your_name_Part_A.jmx**



11. Left-click on any of the HTTP requests to see the details



12. Now click on *View results in Table*. You will see a table with the 10 samples. Note that you can export the results to a file by providing a filename in the corresponding box for further analysis



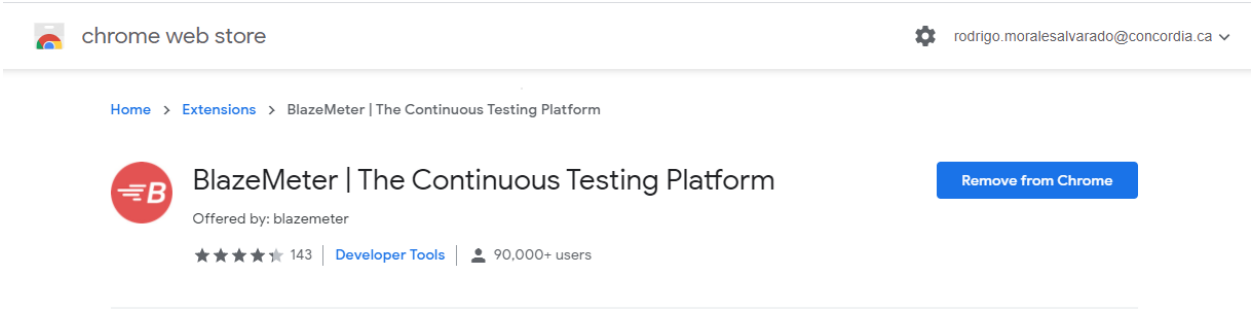
See the following link in SoF if you have questions with respect to the thread group.

Part B

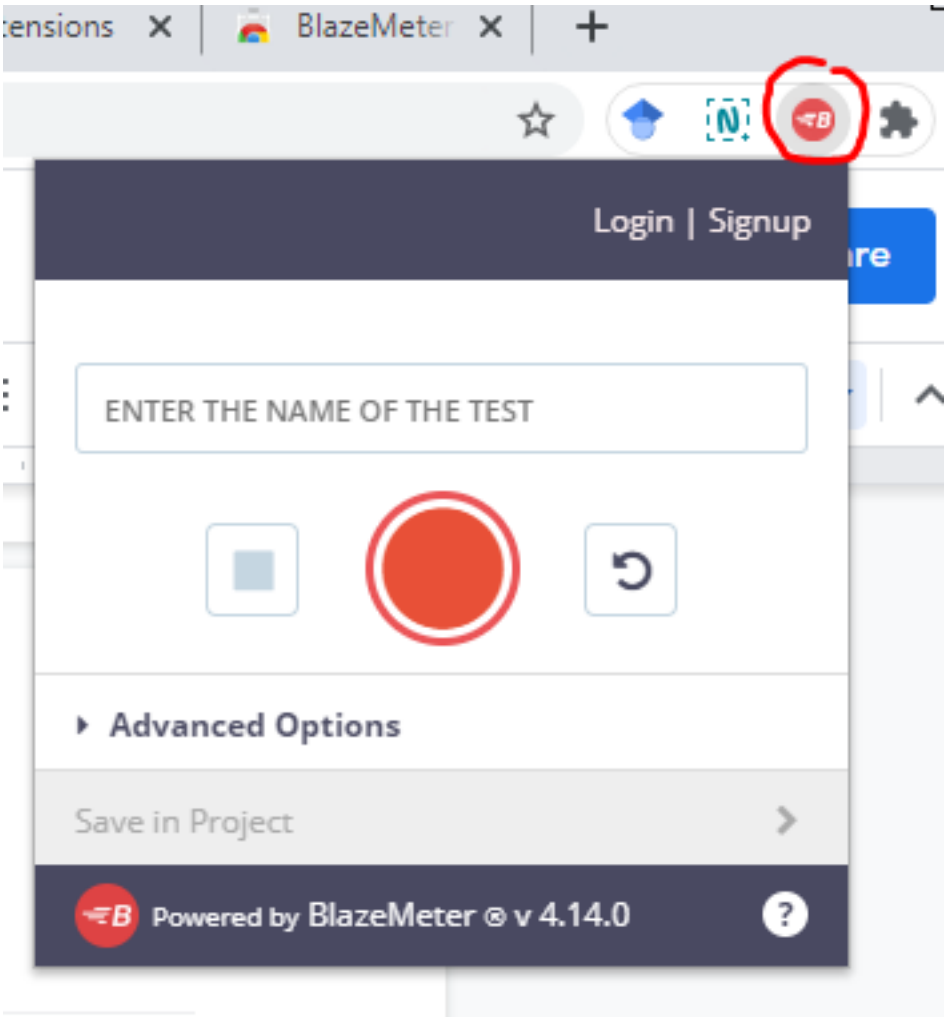
Using *BlazeMeter* (Google Chrome extension) to perform load tests.
Like Selenium, BlazeMeter is a Record/Playback testing tool.

Note that to export the scenario as *JMeter script*, **it is necessary to log in using a Google account, or create a new user.**

1. Add the BlazeMeter add-on from chrome store. Once you added you should see a confirmation page like below:



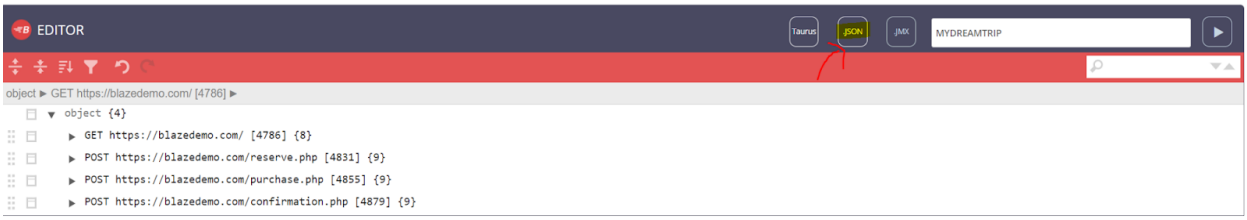
2. Click on the **B** icon (you need to pin it first by clicking on the extensions button) to record a new test case in the extensions



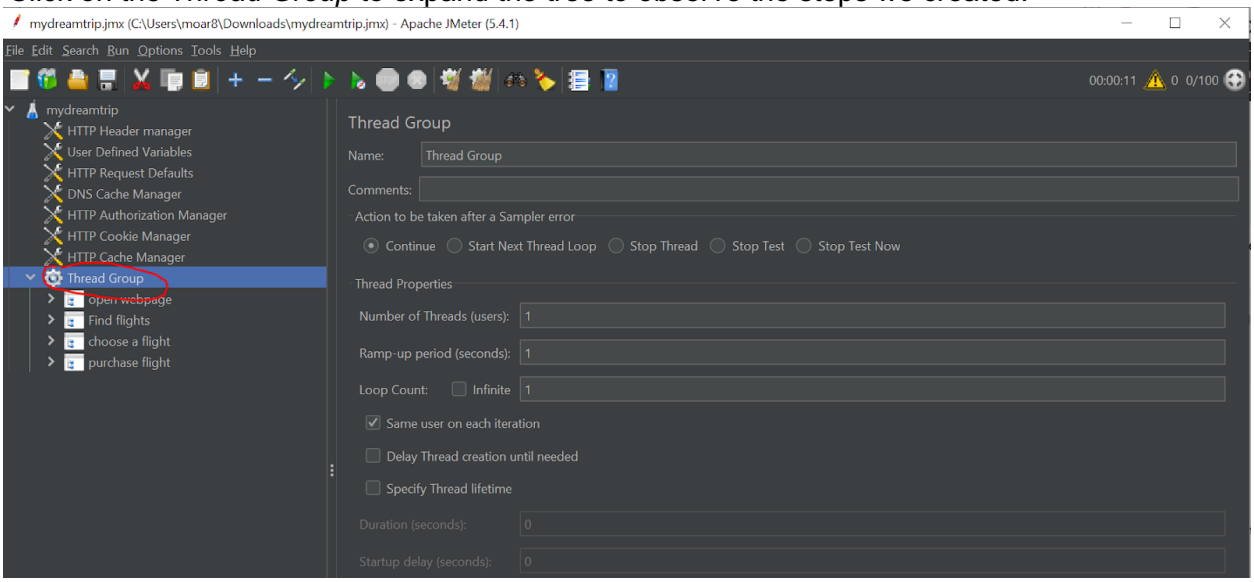
3. Name your test as **mydreamtrip**. Click on start recording.
4. Open the blaze demo website at https://blazedemo.com/ . We will use this dummy website for recording our test case.
5. Add a step **Find flights** using the label option inside Blazemeter. Next, choose a departure and destination city and click on the *Find Flights button* on the dummy website.
6. Add a new step called "**select a flight"**. Then, click on the desired flight.
7. Add a new step called "**pay the flight"**. Then fill in the required information (only your name must be real).
8. You will be redirected to a confirmation page. Stop recording.

9. Save the scenario. To save the scenario, log in to BlazeMeter. click on save and select Select jMeter (JMX). If you want to review the actions recorded, click on edit. Note that typically when you browse, you might be redirected to other websites to display advertisements, or collect information by calling third-party APIs (e.g., Google). These domains must be excluded from your test cases as a good practice.

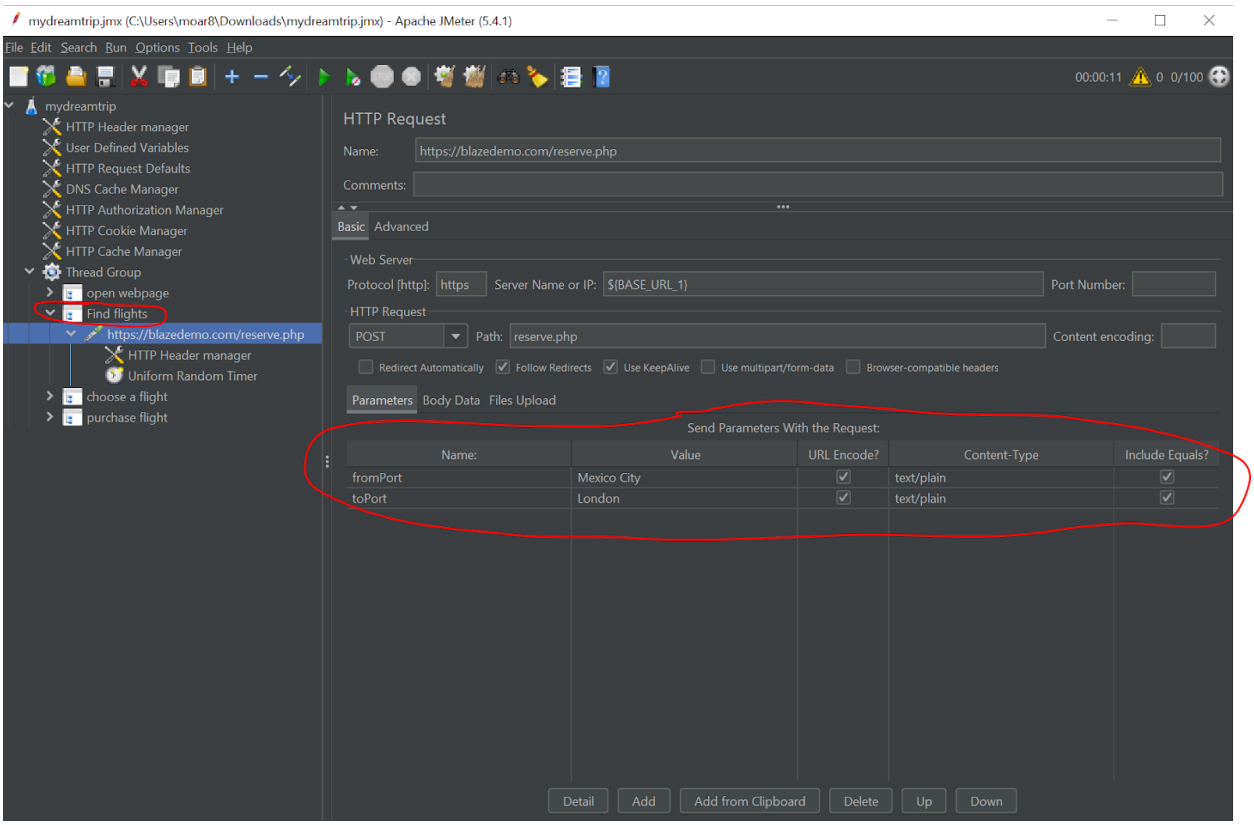
Once you are logged in, click on download your scripts. Select JMeter (MX), and click on **Save**.



10. Open JMeter, and load **mydreamtrip** test plan. You will find a screen similar to the one below. Click on the *Thread Group* to expand the tree to observe the steps we created.

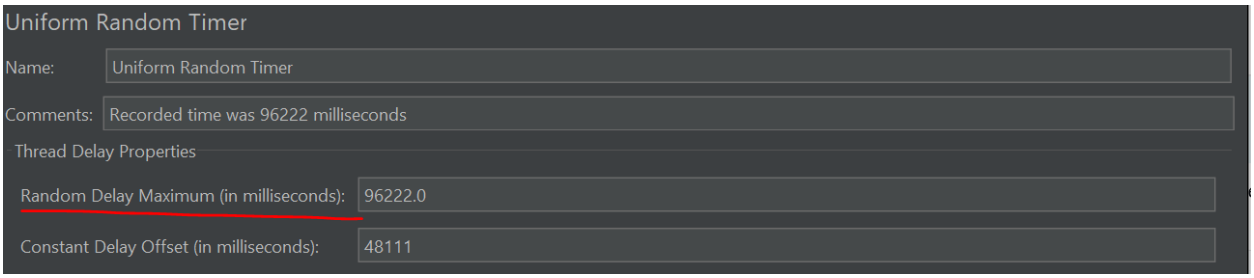


11. Open the “Find flights” thread group. Observe how the information that we filled is sent as parameters in the http post request.



Notice that you can modify the data, for example, to fill in the information of a different user.

12. On the Find flights step, left-click to expand, and then click on the *Uniform Random Timer*. This is known as the think time, or the time taken by a real user to fill the form. "**Random Delay Maximum"** is the maximum time to fill the form estimated by BlazeMeter to fill the form, and it serves as an upper bound each time you execute the test.



13. Select the *Thread Group*, and add *aggregate report* and *summary report* listeners (similar to what you did for **Part A**, steps **8-9).** The *aggregate report* displays both the median and average, while *summary report* shows the standard deviation, among other differences.

14. Change the number *of threads* to 100, and *ramp up* to 10 seconds in the **Thread Group** properties.

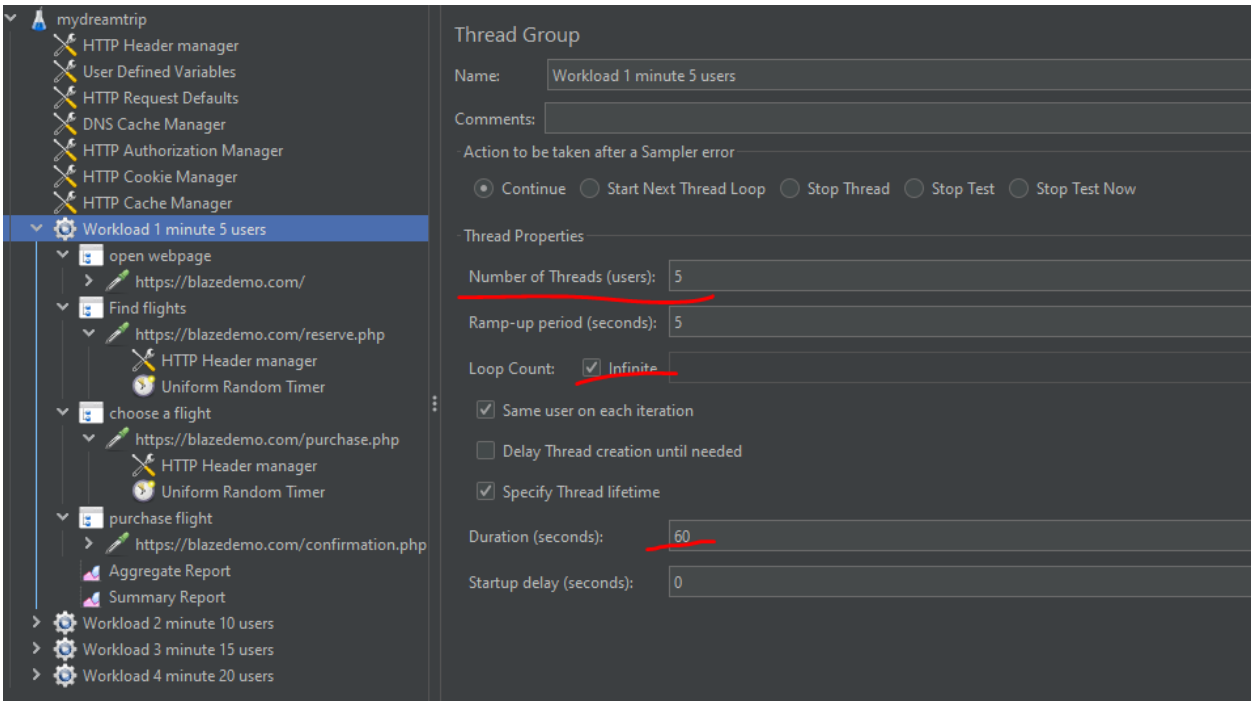
15. Run the test plan and save the Aggregate results as **partB_aggregate.csv,** by clicking on "Save Table Data" at the bottom of the Aggregate report form. Save the test plan as **your_name_part_B.jmx**

Part C

(Use JMeter to load-test blazedemo with **varied** workloads using **CLI** mode

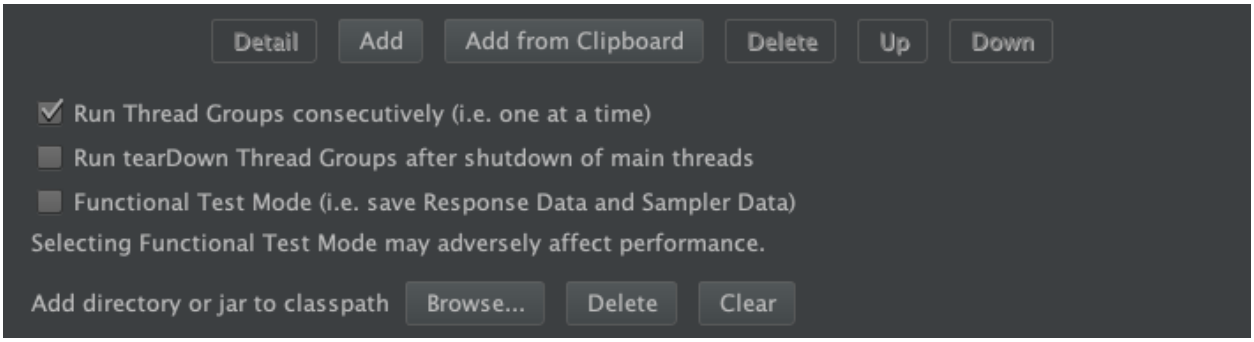
Varied workloads: The first minute, 5 users access blazedemo . The second minute 10 users access blazedemo . The third minute 15 users access blazedemo . The fourth minute 20 users access blazedemo. Save a new copy of **your_name_part_B.jmx** as **your_name_part_C.jmx.**

1. This part can be done by creating four thread groups (one thread group represents one type of workload) under **Test Plan**.



For each workload, specify the number of users.

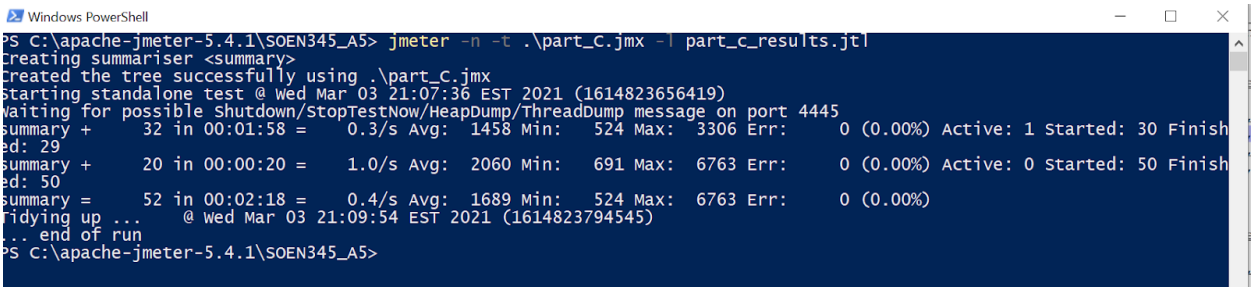
2. Config the order of execution of such four workloads:
Click Test Plan
and Check “Run Thread Groups consecutively”



3. Save the changes to the test plan. Remove all the listeners, and add the following Listeners: *View Results in Table, and Aggregate* report at the level of the test plan.

4. Conduct varied workloads using JMeter CLI mode:

Open a terminal where your .jmx file is stored. Remember to add **apache-jmeter-5.4.1\bin** to your path before running the following command:
jmeter -n -t .\your_name_part_C.jmx -l your_name_part_c_results.jtl
Where
-n: It specifies JMeter is to run in console mode
-t: Name of JMX file that contains the Test Plan
-l: Name of JTL(JMeter text logs) file to log results



Assignment #4: report

Student number:
Student name:

We want to measure the performance of 3 search engines.

- gibiru.com
- duckduckgo.com
- mojeek.com

We measure performance based on response time, and percent of requests with errors.

We use the following scenario. As we did before, use BlazeMeter to generate the JMeter script.

1. Open the search engine
2. Search for *cute animal videos*
3. Search for *funny videos*

Create three different load profiles **for each** of the searches and name them accordingly:

Profile	Number of threads	Ramp-up period
1	25	25
2	50	50
3	75	75

To aggregate results add an **aggregate report** to the thread group.
Remember to tick the check "Run thread groups consecutively" in the plan. For each thread group, set the action "stop thread" after a sampler error.

Answer the following questions:

a. List the average response time from each website in ascending order

b. For each website, how many samples were generated?

c. What is the percentage of requests with errors?

d. Plot a chart with the average response time of each website using different profiles and discuss the results.

e. Based on the results, which search engine would you recommend and why?

f. In 30 to 50 words, discuss the advantages and disadvantages of using BlazeMeter in conjunction with JMeter for performance testing.

Grading:
60% load profiles (20% each)
20% Chart and results discussion (d)
10% Arguments provided in (f)
5% Arguments provided in (e)
5% Written report presentation

To receive points for this assignment, you must provide the following files:

- Provide your tests plans (generated from BlazeMeter). The test plans must include the three load profiles, with their corresponding aggregate reports.
- Export the results generated as CSV, and attach the file for each search engine.
- Attach the screenshot of the execution of JMeter in CLI mode.
- Attach the log files for each load test execution (.jtl).

All files must include your name and student id as a prefix. Otherwise, we will deduct 5% of the total weight of this assignment.

Total 100 points.

Graduate attribute	Description	Score out 100%	Comments
[INV-1] Background and hypothesis formulation	Students identify and describe the reasons for the investigation.		
[INV-2] Designing experiments	Students design experiments to assess the performance of a website with different workloads		

[INV-3] Conducting experiments and collection of data	Students conduct experiments to collect data regarding the performance of a website with different workloads		
[INV-4] Analysis and interpretation of data	Students analyze the data collected and derive conclusions		