SWEN90007

Software Design & Architecture

Semester 2 - 2019

Submission 4

Testing & Reflection

Wednesday 16:15 PM Workshop

Savan Kanabar

Login: - skanabar@student.unimelb.edu.au

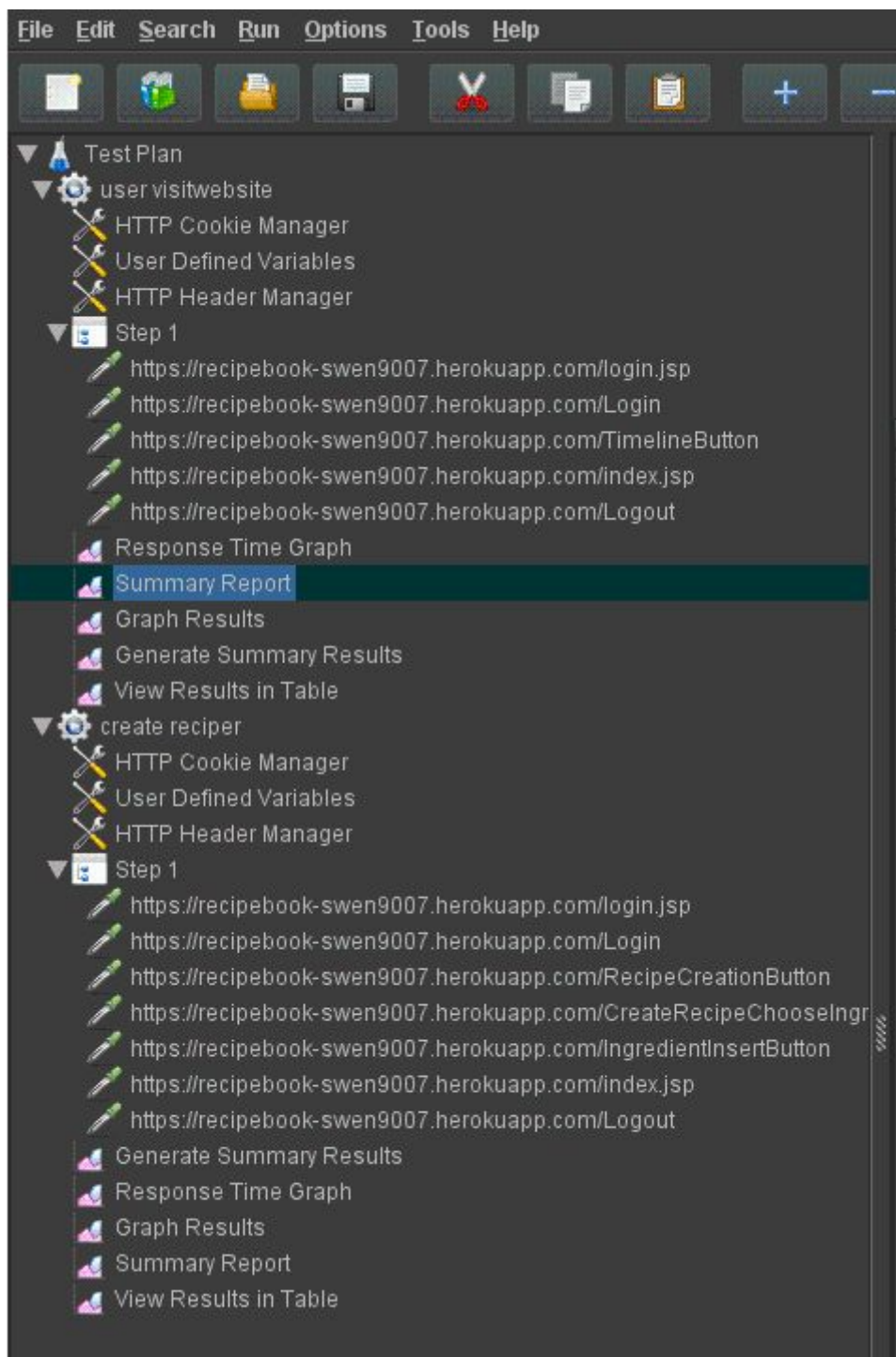ID: - 965371

Umut Cem Soyulmaz

Login: - usoyulmaz@student.unimelb.edu.au

ID: - 989654

Our Test Plan

This image shows our test plan, we have 2 users, one visiting our website to view all the recipes and another user who wants to go to website and post some recipes. We will check for how long our system can handle multiple requests continuously before it breaks, throughput, response time and latency.

Latency

Latency of Timeline

Figure 1Latency of View Timeline

Latency of Create Recipe

Figure 2 Latency of Create Recipe

Throughput

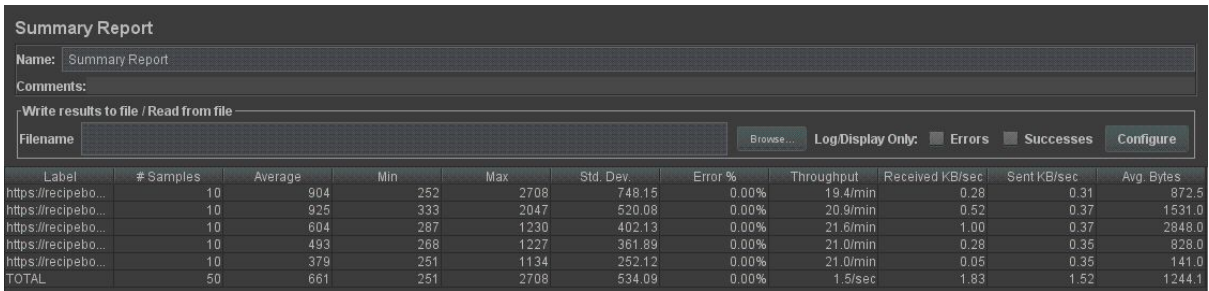## Throughput of Timeline

Figure 3 Throughput of Timeline

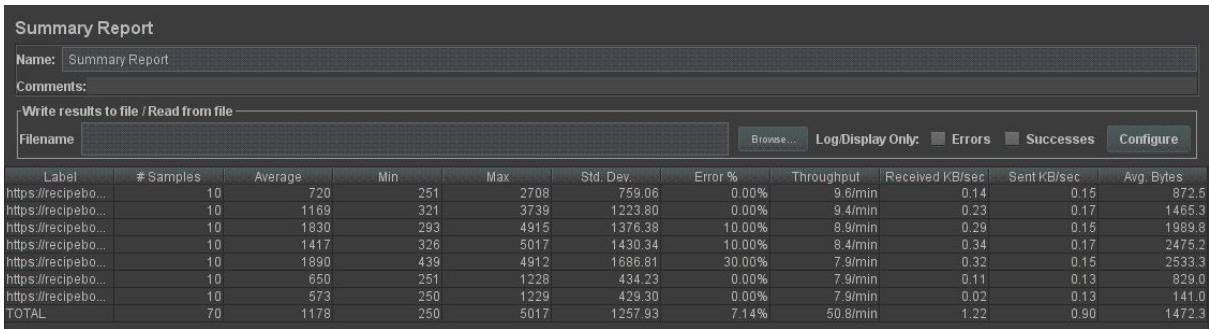## Throughput of Create Recipe

Figure 4 Throughput of Create Recipe

Response Time Graph
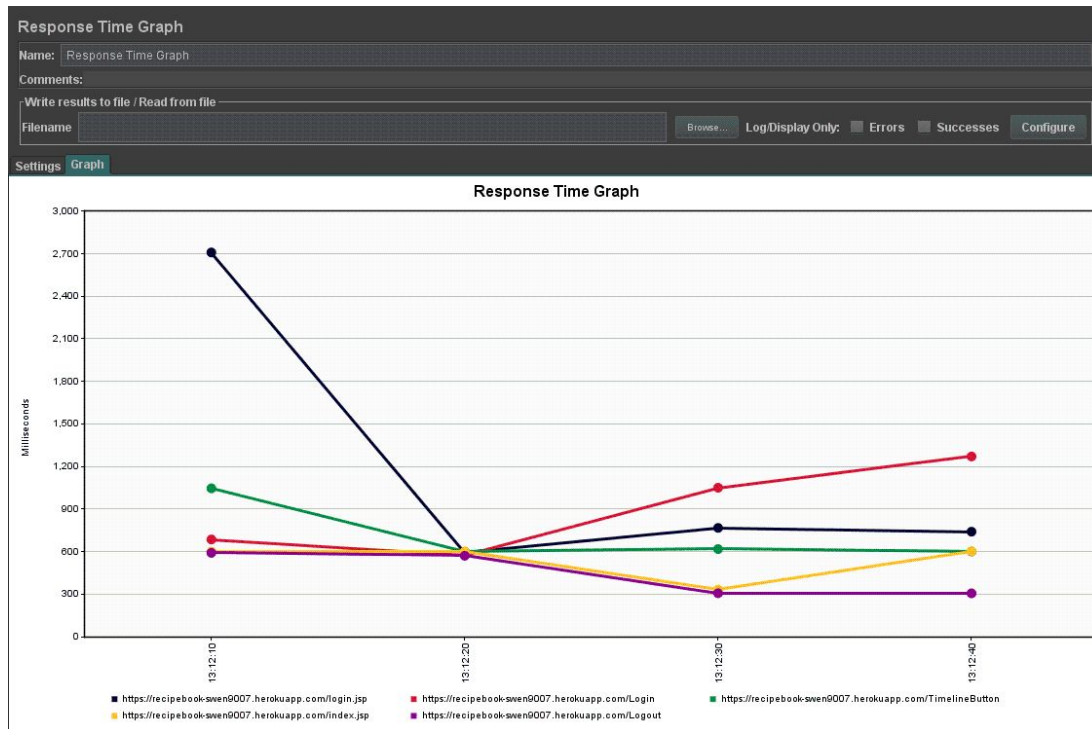
View Timeline

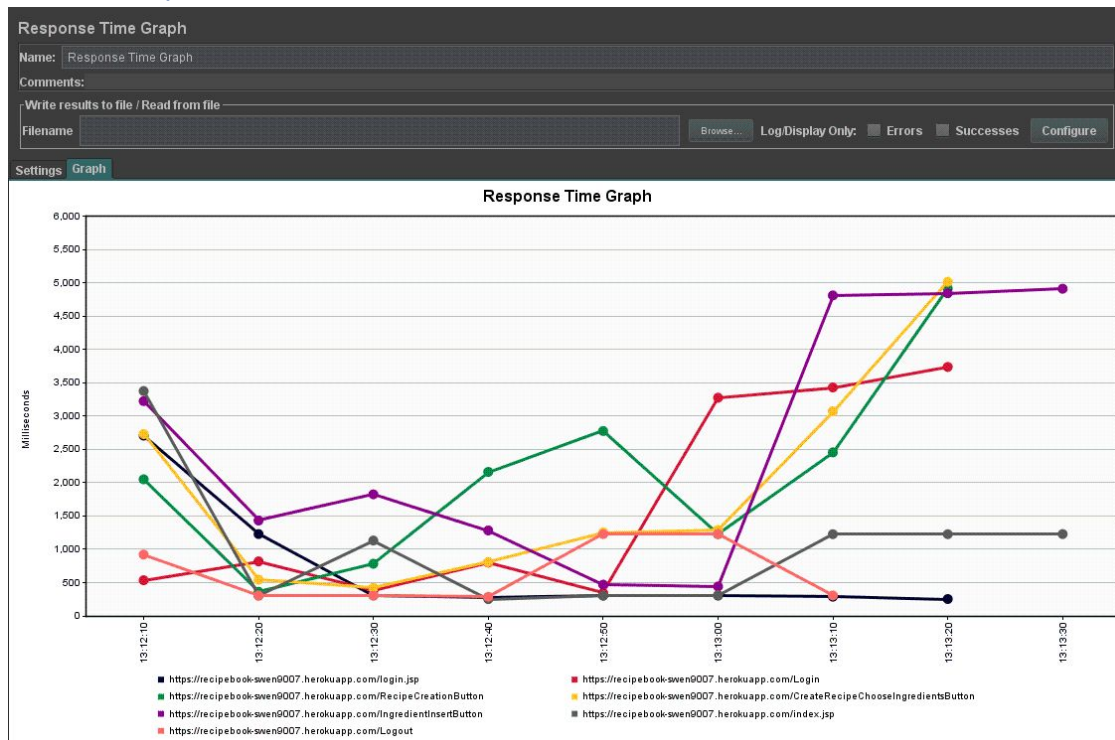*Figure 5 Response time for Timeline*

Create Recipe

Figure 6 Response time for create recipe

Discussion

Figure 1 and Figure 2 shows the latency in our system when a user visits our timeline which contains all the recipes, or they try to create recipes. Here the amount of data sent and received is just few bytes so the performance is acceptable, and latency is low, but if our system grows in terms of users and recipes than our performance will get affected. To mitigate this, we can implement Lazy Load feature in timeline as well, right now it is only associated with comments which are made on recipe and ingredients of the recipe. The patterns which increase the performance are Unit of Work and Identity map as well.

Figure 3 and Figure 4 show the throughput of our timeline page and creating a recipe page, it is almost 1.5 pages per second and almost 51 pages per minute respectively. That is very good throughput and this has been achieved with correct implementation of data mapper pattern and having separate object models for all the components, along with that, Identity Map & Association Table patterns are also responsible in our system for increasing the throughput. For example, we can retrieve multiple ingredients associated with a recipe just by one call to database.

Figure 5 and 6 shows response time graph of our timeline page and create recipe page, the initial response time is higher because Heroku needs time for warm up, otherwise response time for timeline is very good as we can see, Response time for create recipe is a bit higher sometimes because that is the time when the data is sent into backend and processed.

Reflection

One of the main decisions we made in our system was regarding where we should implement our session. We decided to go with Client Side Session because that would give us better performance compared to server side or database side session as we have limited resources on both those sides and now while testing our application, we feel that the decision we took was correct because the numbers of our performance tests are very good.

For future development we would implement Lazy Load for full system rather than just some specific package to increase our performance even more

We would also remove Unit of Work from our system if we continue future development because we can achieve the same objective in easier ways,, for example, we can fetch data in one transaction and store in another and combine this process with optimistic lock or we can also use batch processing which allows us to group similar SQL statements in a batch and process them in one call, this reduces latency overhead.

Limitations

It is not possible for us to stress test the system because we are using Shiro for security and have credentials which limits our actions in jmeter for example, we cannot have 100 users accessing the page simultaneously in this case to gauge the performance and bottleneck in our system. However, we did test what would happen if a user post multiple recipes in a very short time using the loop feature in jmeter and it crashed our deployed app after some time, the number of loops that it ran for was different for all the 3 times we tested and due to this we believe that the issue is free tier of Heroku. The image below shows server crashing after 2 loops.

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file
Filename: [Browse...] Log/Display Only: ☐ Errors ☐ Successes [Configure]

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
67	13:13:28.995	create reciper 2-1	https://recipebook..	5017	⊗	2842	1238	5017	921
66	13:13:24.080	create reciper 2-1	https://recipebook..	4915	⊗	2825	1047	4915	921
68	13:13:34.015	create reciper 2-1	https://recipebook..	4912	⊗	2826	1182	4912	919
61	13:13:13.738	create reciper 2-1	https://recipebook..	4812	⊗	2826	1182	4812	1024
65	13:13:20.340	create reciper 2-1	https://recipebook..	3739	✓	865	1146	3739	0
58	13:13:06.160	create reciper 2-1	https://recipebook..	3276	✓	1532	1146	3276	0
19	13:12:31.652	create reciper 2-1	https://recipebook..	3213	⊗	2912	1182	3213	920
60	13:13:10.665	create reciper 2-1	https://recipebook..	3072	✓	2435	1238	3072	0
52	13:12:59.723	create reciper 2-1	https://recipebook..	2853	✓	1897	1047	2853	0
1	13:12:17.798	create reciper 2-1	https://recipebook..	2708	✓	940	428	2707	2417
45	13:12:53.118	create reciper 2-1	https://recipebook..	2699	✓	1897	1047	2699	0
38	13:12:46.829	create reciper 2-1	https://recipebook..	2332	✓	1897	1047	2332	0
31	13:12:41.231	create reciper 2-1	https://recipebook..	1992	✓	1897	1047	1992	865
5	13:12:22.743	create reciper 2-1	https://recipebook..	1433	✓	1325	1182	1432	920
12	13:12:27.249	create reciper 2-1	https://recipebook..	1432	✓	1613	1182	1432	920
53	13:13:02.579	create reciper 2-1	https://recipebook..	1292	✓	2435	1238	1292	917
39	13:12:49.162	create reciper 2-1	https://recipebook..	1289	✓	2435	1238	1289	921
33	13:12:43.550	create reciper 2-1	https://recipebook..	1282	✓	2477	1182	1282	901
2	13:12:20.516	create reciper 2-1	https://recipebook..	1275	✓	1532	612	1275	895
24	13:12:36.937	create reciper 2-1	https://recipebook..	1264	✓	1897	1047	1264	960
46	13:12:55.818	create reciper 2-1	https://recipebook..	1252	✓	2435	1238	1252	921
37	13:12:45.588	create reciper 2-1	https://recipebook..	1239	✓	1532	1146	1239	910
49	13:12:57.865	create reciper 2-1	https://recipebook..	1229	✓	141	1033	1228	921
8	13:12:24.791	create reciper 2-1	https://recipebook..	1228	✓	865	1036	1228	920
15	13:12:29.296	create reciper 2-1	https://recipebook..	1228	✓	865	1036	1228	904
42	13:12:51.210	create reciper 2-1	https://recipebook..	1228	✓	141	1033	1228	920
56	13:13:04.624	create reciper 2-1	https://recipebook..	1228	✓	141	1033	1228	920
69	13:13:38.928	create reciper 2-1	https://recipebook..	1228	✓	829	1036	1228	921
59	13:13:09.438	create reciper 2-1	https://recipebook..	1227	✓	1897	1047	1227	870
62	13:13:18.551	create reciper 2-1	https://recipebook..	1227	✓	829	1036	1227	920
27	13:12:39.055	create reciper 2-1	https://recipebook..	1179	✓	829	1036	1179	891
20	13:12:34.868	create reciper 2-1	https://recipebook..	1085	✓	829	1036	1085	832
11	13:12:26.677	create reciper 2-1	https://recipebook..	571	✓	2434	1238	570	0

☐ Scroll automatically? ☐ Child samples? **No of Samples** 70 **Latest Sample** 252 **Average** 1178 **Deviation** 1205