

Functional Tests:

We can perform functional tests by checking for the functionalities written in the requirements table. Requirements 1 through 14 are “must have and should have” functional requirements. Requirements 15 through 17 are “could have” requirements.

test_display() function checks if the webpage correctly shows the elements it is supposed to show. **Validates RE-01 and RE-06.**

test_navigation_buttons() function checks if there are previous and next buttons on the website and if they correctly show the next and previous image as they are supposed to. **Validates RE-02 and RE-03.**

test_wrap_around() function checks if wrap arounds work, **Validating RE-04 and RE-05.**

test_comment_submission() function submits comments and checks if they are correctly submitted. Checks the order of the comment, as username first and comment after, and then checks if the text fields are cleared or not. **Validates RE-07, RE-08 and RE-11.**

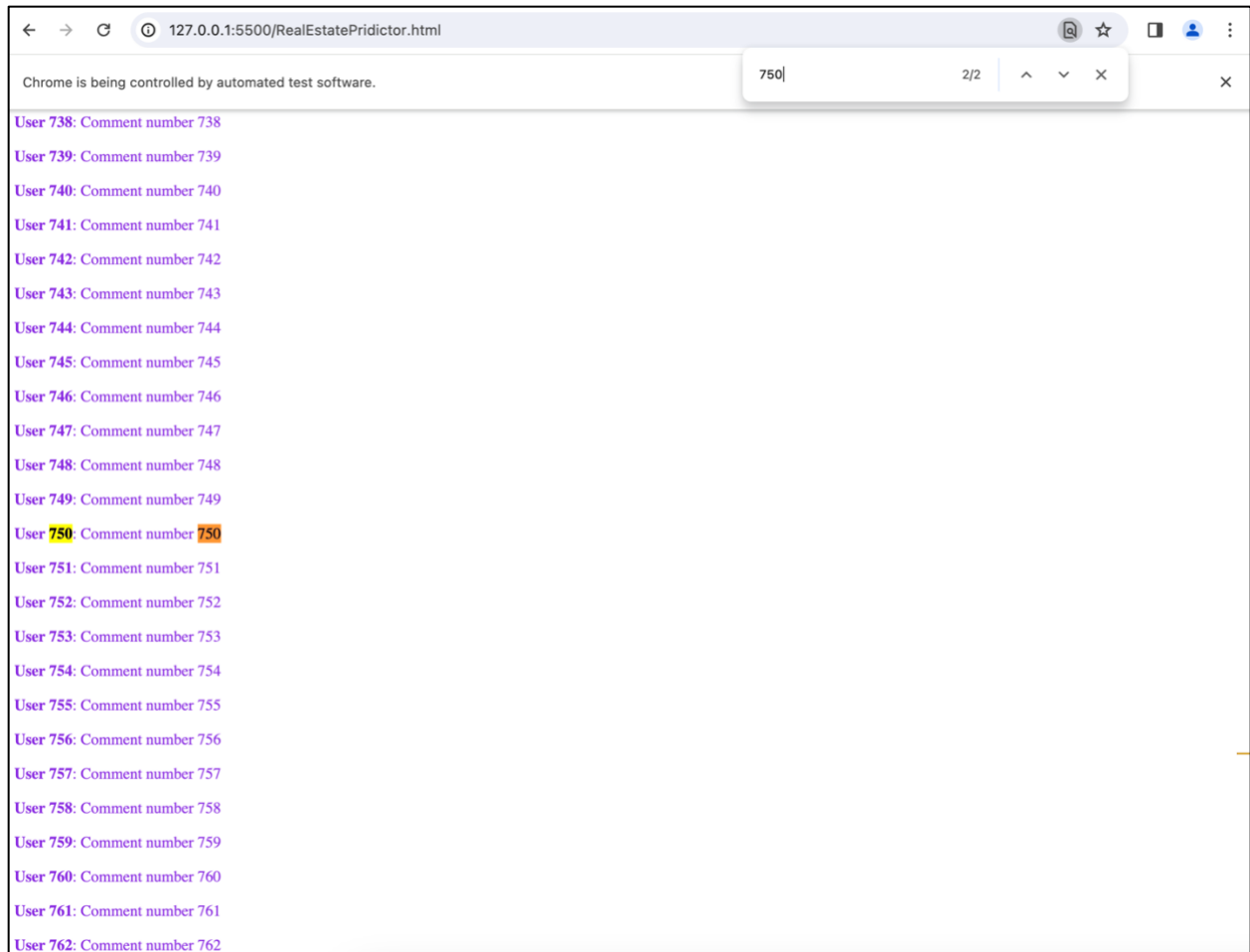
test_comments_persistence() function checks for the persistency of comments. **Validates RE-09.**

test_1000_comments() function checks if each webpage can accommodate at least 1000 comments, and if the comments are scrollable. **Validates RE-10 and RE-12.**

In the same test we added a time delay of 60 seconds and by using CMD-F we searched 750 to check if we can search any comment by finding it on the page. Although it can be searched on the page, there is no in-built functionality in our real estate predictor to search the comment. So, in a way we can say that the comments, if they are searched by a unique key present in them, then that comment can be searched by using CMD-F. However, there is no functionality implemented to search for instance all comments of a particular user. Thus, we can say **that the RE-13 is not implemented.**

These requirements are not implemented!

RE-13	The comments should be searchable.	M
RE-14	In any discussion, the real estate agent should be identifiable as such.	M
RE-15	The comments could have a like button	L
RE-16	The comments could keep track of the number of likes for each comment	L
RE-17	The comments could have spell check.	L



Raw output of functional test:

.....

Ran 6 tests in 274.647s

OK

Stress and Scalability Test Report:

Stress and Scalability test were conducted on a mac 2019 with intel i9, 16 GB ram. For stress test, the webpage was overloaded with 1500 of comments being entered on each of the pages. Time taken to enter each 100 comments were recorded for comparison, and resources usage was tracked. For Scalability Test, in similar way to Stress test, 1500 comments were added iteratively by 1, 5 and 10 threads in parallel and time was recorded completion of each thread.

Stress Test:

Here is the raw output of the stress test:

Submitted 100 comments on page 0, time taken: 17.031492233276367 seconds
Submitted 200 comments on page 0, time taken: 17.964168071746826 seconds
Submitted 300 comments on page 0, time taken: 18.95777988433838 seconds
Submitted 400 comments on page 0, time taken: 22.160690307617188 seconds
Submitted 500 comments on page 0, time taken: 24.021145820617676 seconds
Submitted 600 comments on page 0, time taken: 29.330840826034546 seconds
Submitted 700 comments on page 0, time taken: 35.16910791397095 seconds
Submitted 800 comments on page 0, time taken: 42.491716146469116 seconds
Submitted 900 comments on page 0, time taken: 48.729485750198364 seconds
Submitted 1000 comments on page 0, time taken: 56.29409193992615 seconds
Submitted 1100 comments on page 0, time taken: 66.24230194091797 seconds
Submitted 1200 comments on page 0, time taken: 79.10155487060547 seconds
Submitted 1300 comments on page 0, time taken: 90.14845085144043 seconds
Submitted 1400 comments on page 0, time taken: 102.60793113708496 seconds
Submitted 1500 comments on page 0, time taken: 116.48570203781128 seconds
Submitted 100 comments on page 1, time taken: 13.808525085449219 seconds
Submitted 200 comments on page 1, time taken: 15.44584608078003 seconds
Submitted 300 comments on page 1, time taken: 17.822314977645874 seconds
Submitted 400 comments on page 1, time taken: 21.198618173599243 seconds
Submitted 500 comments on page 1, time taken: 26.248559951782227 seconds
Submitted 600 comments on page 1, time taken: 31.009228944778442 seconds
Submitted 700 comments on page 1, time taken: 36.576435804367065 seconds
Submitted 800 comments on page 1, time taken: 43.62559103965759 seconds
Submitted 900 comments on page 1, time taken: 52.291404008865356 seconds
Submitted 1000 comments on page 1, time taken: 60.331857204437256 seconds
Submitted 1100 comments on page 1, time taken: 70.84150075912476 seconds
Submitted 1200 comments on page 1, time taken: 83.18100500106812 seconds
Submitted 1300 comments on page 1, time taken: 96.58001208305359 seconds
Submitted 1400 comments on page 1, time taken: 110.30696320533752 seconds
Submitted 1500 comments on page 1, time taken: 124.4517879486084 seconds
Submitted 100 comments on page 2, time taken: 14.21062707901001 seconds
Submitted 200 comments on page 2, time taken: 15.576992988586426 seconds

Submitted 300 comments on page 2, time taken: 18.097819089889526 seconds
Submitted 400 comments on page 2, time taken: 21.80224108695984 seconds
Submitted 500 comments on page 2, time taken: 28.638401746749878 seconds
Submitted 600 comments on page 2, time taken: 34.645166873931885 seconds
Submitted 700 comments on page 2, time taken: 40.04827404022217 seconds
Submitted 800 comments on page 2, time taken: 46.77599787712097 seconds
Submitted 900 comments on page 2, time taken: 53.82850885391235 seconds
Submitted 1000 comments on page 2, time taken: 63.56348991394043 seconds
Submitted 1100 comments on page 2, time taken: 73.96190810203552 seconds
Submitted 1200 comments on page 2, time taken: 86.9245400428772 seconds
Submitted 1300 comments on page 2, time taken: 100.55238389968872 seconds
Submitted 1400 comments on page 2, time taken: 115.02224516868591 seconds
Submitted 1500 comments on page 2, time taken: 131.87899327278137 seconds
Submitted 100 comments on page 3, time taken: 14.213139772415161 seconds
Submitted 200 comments on page 3, time taken: 16.50778603553772 seconds
Submitted 300 comments on page 3, time taken: 22.971533060073853 seconds
Submitted 400 comments on page 3, time taken: 24.900105953216553 seconds
Submitted 500 comments on page 3, time taken: 31.173030138015747 seconds
Submitted 600 comments on page 3, time taken: 34.29809617996216 seconds
Submitted 700 comments on page 3, time taken: 40.17784905433655 seconds
Submitted 800 comments on page 3, time taken: 48.63104701042175 seconds
Submitted 900 comments on page 3, time taken: 56.27932691574097 seconds
Submitted 1000 comments on page 3, time taken: 67.29518103599548 seconds
Submitted 1100 comments on page 3, time taken: 77.90213584899902 seconds
Submitted 1200 comments on page 3, time taken: 90.40666317939758 seconds
Submitted 1300 comments on page 3, time taken: 106.00745701789856 seconds
Submitted 1400 comments on page 3, time taken: 121.53048491477966 seconds
Submitted 1500 comments on page 3, time taken: 137.84068298339844 seconds
Submitted 100 comments on page 4, time taken: 14.268728256225586 seconds
Submitted 200 comments on page 4, time taken: 16.133392095565796 seconds
Submitted 300 comments on page 4, time taken: 19.283796787261963 seconds
Submitted 400 comments on page 4, time taken: 23.83200216293335 seconds
Submitted 500 comments on page 4, time taken: 30.70002508163452 seconds
Submitted 600 comments on page 4, time taken: 34.97180914878845 seconds
Submitted 700 comments on page 4, time taken: 41.85740399360657 seconds
Submitted 800 comments on page 4, time taken: 51.23526668548584 seconds
Submitted 900 comments on page 4, time taken: 59.730507612228394 seconds
Submitted 1000 comments on page 4, time taken: 69.99820709228516 seconds
Submitted 1100 comments on page 4, time taken: 81.72521996498108 seconds
Submitted 1200 comments on page 4, time taken: 94.8781008720398 seconds
Submitted 1300 comments on page 4, time taken: 109.15100526809692 seconds
Submitted 1400 comments on page 4, time taken: 126.6663978099823 seconds
Submitted 1500 comments on page 4, time taken: 143.85952305793762 seconds

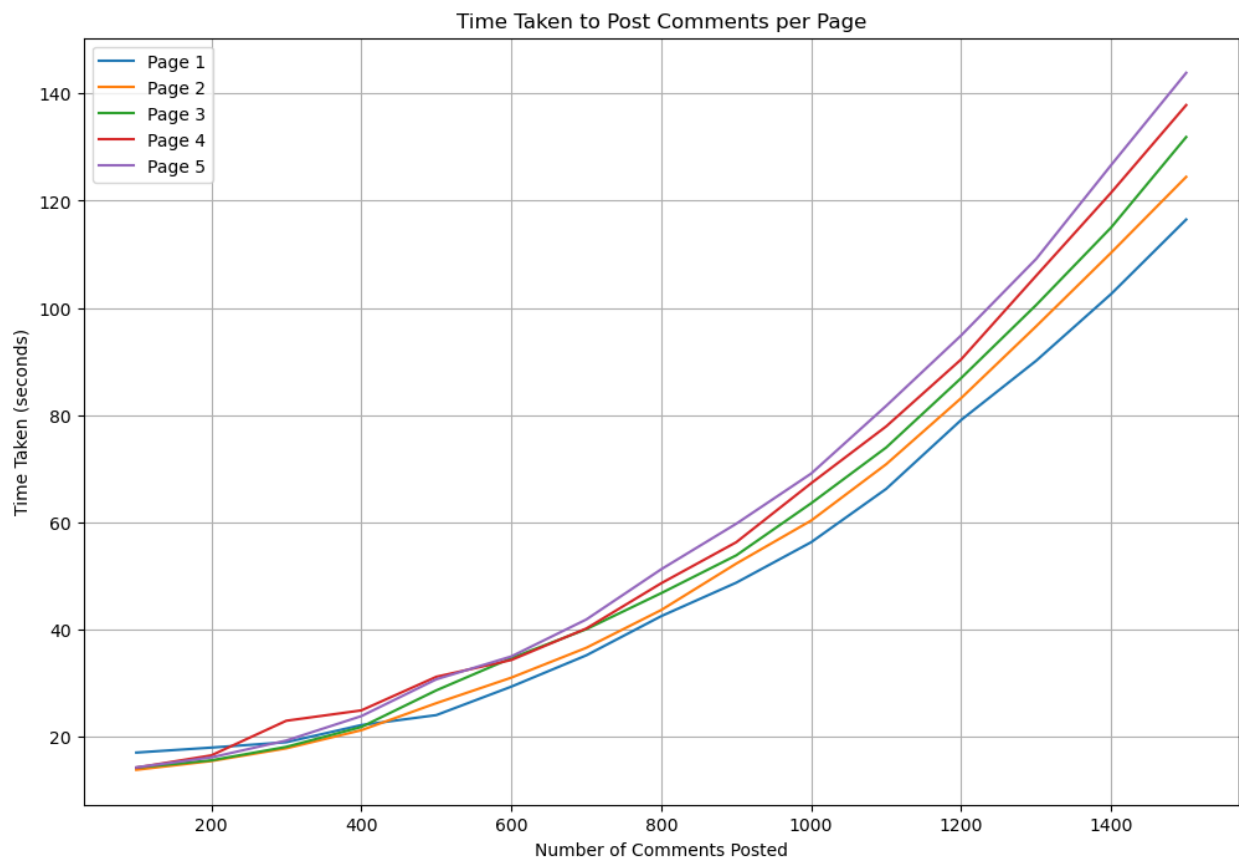
.

Ran 1 test in 4227.673s

OK

Let's analyze the first set of output, which is bolded. The time to enter 100 comments were recorded to see if more time would be required as number of comments will keep increasing. What we can see from the implementation of the webpage is that the comments are all just one string concatenated and bolded to appear like how they do on the web page. So, as the size of the string will keep increasing the time required to add more characters into it would increase. However, we saw no peculiar increment in memory consumption from beginning to end of each batch (1500 comments on one page).

To summarize the data in the output, here is graph:







At first look it would seem Like, as the number of comments is increasing the time required to enter the comment is increase. But that's not the case, the tie which was recorded was just to enter 15 times 100 comments on each page. What we need to notice is that the first 100 to 200 comments took less 20 seconds only on all the webpages. But the last 100 comments on each webpage took more than 115 seconds to be entered. This indicates

that as the number of comments on each house in our application would increase, the time to add newer comments would drastically increase.

Another important thing to note is that there is a gradual increase in time required to enter each 100 comments from the first house to the last house. All the pages took somewhat same time for their first 800 comments, but after that the higher the page the more time was required to enter same number of comments. For instance, when 1100 comments were already posted on the first page, the next 100 comments took around 80 seconds to be posted. In contrast to the last page when 1100 comments were already posted, the next 100 comments took 94 seconds. Why these 94 seconds stand out so much is because the first hundred comments on any page only took about 15 – 20 seconds, but the 11th hundred comments took almost 6 times more time than that. This kept getting more and more worse, with the last 100 comments on the first page being posted in 116 seconds to last 100 comments being posted on the last page in 143 seconds.

Bottom line hear is, successive comments on same page exponentially increases the time required to add newer comments. Higher number of comments on any page would lead to slower processing of newer comment on any page.

Process Name	Mem... <small>▼</small>	Threads	Ports	PID	User
java	2.65 GB	60	228	543	rashidkhan
WindowServer	1.08 GB	16	3,198	185	_windowserver
kernel_task	490.3 MB	275	0	0	root
mysqld	386.7 MB	34	59	66995	rashidkhan
mysqld	355.0 MB	39	64	370	_mysql
Code Helper (Plugin)	324.1 MB	15	71	50253	rashidkhan
 Google Chrome	315.0 MB	45	1,266	650	rashidkhan
Creative Cloud UI Helper (Renderer)	284.0 MB	35	427	829	rashidkhan
 Microsoft Word	266.3 MB	40	934	50434	rashidkhan
Code Helper (Renderer)	250.7 MB	27	260	49072	rashidkhan
Google Chrome Helper (Renderer)	249.8 MB	21	303	51468	rashidkhan
Google Chrome Helper (GPU)	192.3 MB	14	267	761	rashidkhan
Code Helper (Plugin)	175.9 MB	17	77	50167	rashidkhan
Code Helper (GPU)	166.1 MB	10	162	49069	rashidkhan
Google Chrome Helper (Renderer)	131.5 MB	18	155	55181	rashidkhan
Google Chrome Helper (Renderer)	129.5 MB	20	509	49499	rashidkhan
 Creative Cloud Core Service	108.5 MB	34	468	820	rashidkhan
 Code	96.1 MB	36	516	49066	rashidkhan

Below is a screen shot when the last 100 comments out of 1500 comments on the last page were being entered. As we can see their memory consumption by all the instances of Google Chrome and Visual studio code is all less then Microsoft word which was just existing in the background. Apart from the main instance of Google Chrome which is above word, however that instance was not the test instance created by selenium, that window had youtube, lear@seneca, stackoverflow, outlooks and whatsapp web.

Scalability test:

Here is the raw output of a number of runs which was conducted one after another (after 5 10 minutes) in the sae environment to test scalability:

10 users inserting 500 comments in parallel

User 7 completed 500 actions in 435.967866897583 seconds.
User 4 completed 500 actions in 436.85020184516907 seconds.
User 6 completed 500 actions in 436.9159417152405 seconds.
User 8 completed 500 actions in 437.83718609809875 seconds.
User 0 completed 500 actions in 437.9104881286621 seconds.
User 9 completed 500 actions in 438.5692939758301 seconds.
User 1 completed 500 actions in 438.619637966156 seconds.
User 2 completed 500 actions in 438.6181888580322 seconds.
User 3 completed 500 actions in 439.20077681541443 seconds.
User 5 completed 500 actions in 439.2075879573822 seconds.

5 users inserting 500 comments in parallel

User 2 completed 500 actions in 253.79927611351013 seconds.
User 0 completed 500 actions in 254.36794114112854 seconds.
User 3 completed 500 actions in 254.36635518074036 seconds.
User 1 completed 500 actions in 254.67795395851135 seconds.
User 4 completed 500 actions in 254.67131805419922 seconds.

10 users inserting 1000 comments in parallel

User 2 completed 1000 actions in 1090.8130779266357 seconds.
User 3 completed 1000 actions in 1090.7135581970215 seconds.
User 9 completed 1000 actions in 1090.7853329181671 seconds.
User 0 completed 1000 actions in 1092.3844101428986 seconds.
User 7 completed 1000 actions in 1093.4479048252106 seconds.
User 1 completed 1000 actions in 1093.9694321155548 seconds.
User 4 completed 1000 actions in 1094.8543281555176 seconds.
User 5 completed 1000 actions in 1095.7774310112 seconds.
User 8 completed 1000 actions in 1096.0047438144684 seconds.
User 6 completed 1000 actions in 1095.867789030075 seconds.

5 users inserting 1000 comments in parallel

User 3 completed 1000 actions in 714.4226140975952 seconds.
User 1 completed 1000 actions in 715.6942782402039 seconds.
User 4 completed 1000 actions in 715.6531059741974 seconds.
User 0 completed 1000 actions in 717.490602016449 seconds.
User 2 completed 1000 actions in 717.4894409179688 seconds.

10 users inserting 1500 comments in parallel

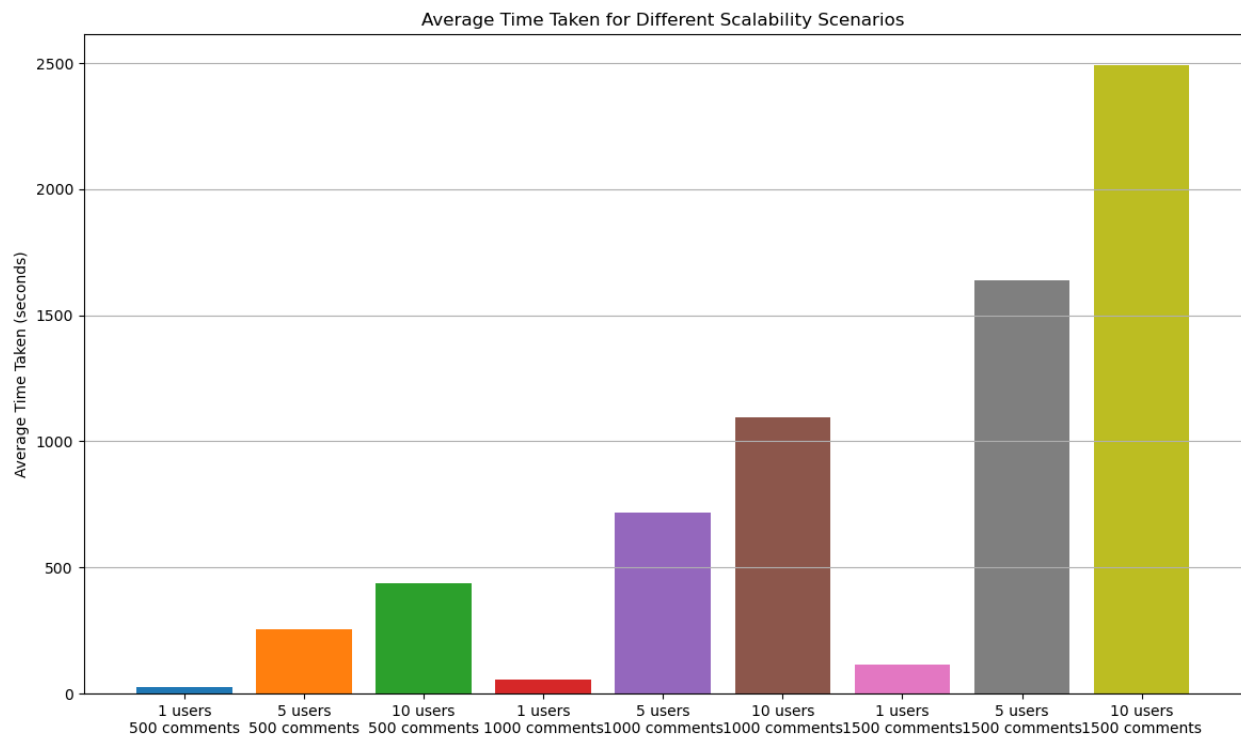
User 1 completed 1500 actions in 2478.954972267151 seconds.

User 8 completed 1500 actions in 2485.22540807724 seconds.
User 5 completed 1500 actions in 2488.208236694336 seconds.
User 7 completed 1500 actions in 2488.1926708221436 seconds.
User 9 completed 1500 actions in 2488.1977791786194 seconds.
User 6 completed 1500 actions in 2488.2094299793243 seconds.
User 3 completed 1500 actions in 2490.4012799263 seconds.
User 2 completed 1500 actions in 2493.9364380836487 seconds.
User 4 completed 1500 actions in 2493.8909888267517 seconds.
User 0 completed 1500 actions in 2494.0865919589996 seconds.

5 users inserting 1500 comments in parallel

User 3 completed 1500 actions in 1634.2463738918304 seconds.
User 0 completed 1500 actions in 1634.1093497276306 seconds.
User 2 completed 1500 actions in 1635.9946057796478 seconds.
User 4 completed 1500 actions in 1638.3100340366364 seconds.
User 1 completed 1500 actions in 1639.6163139343262 seconds.

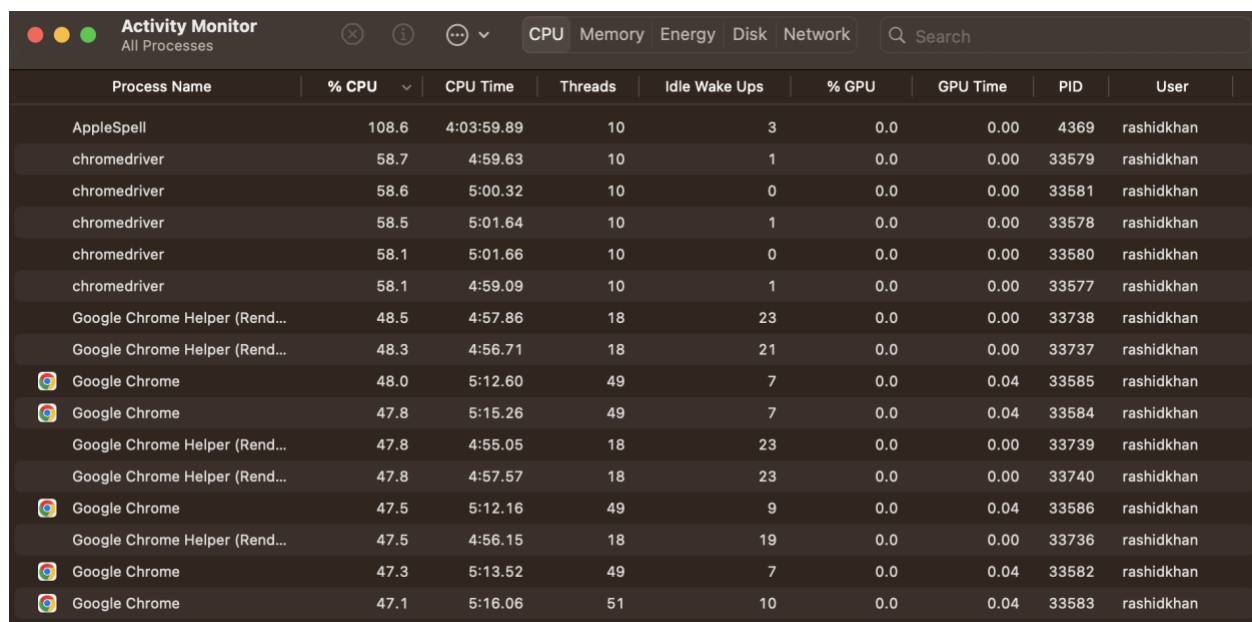
The output shows the time it required by 5 and 10 threads each to enter 500, 1000 and 1500 comments each in parallel. We already have the output of entering 500, 1000 and 1500 comments when entered by a single user in our stress test. To summarize the analysis here is a graph which shows the average time required for different scalability scenarios as mentioned:



As we can see from the graph, if there is only one user interacting with our website, the performance is very good. The time required to enter 500, 1000 and even 1500 comments by one user only at a time is comparatively very less than the time required when 5 users were entering only 500 comments. As users get even more higher to 10, the time required to enter 500 comments went even higher to almost 500 seconds. In the second scenario when 1, 5 and 10 users were entering 1000 comments, it took only about 50 seconds when only 1 user was entering 1000 comments, but when number of users grew to 5, with each entering 1000 comments the time required. Increased exponentially all the way from just about 50 to almost 700. When got even worse when 10 users were entering 1000 comments as it took about 400 more seconds compared to the time 5 users took to enter same number of comments.

The bottom line here is, as number of users increases the amount of time to post comments gradually increases.

During the process, Apple Spell (keyboard process) was severely lagging, I couldn't type anything on my laptop. As we can see here, Apple Spell was consuming the most CPU usage, but of course it would as virtually 5 users were using it at that point. It only got worse when I run the test for 10 users after that.

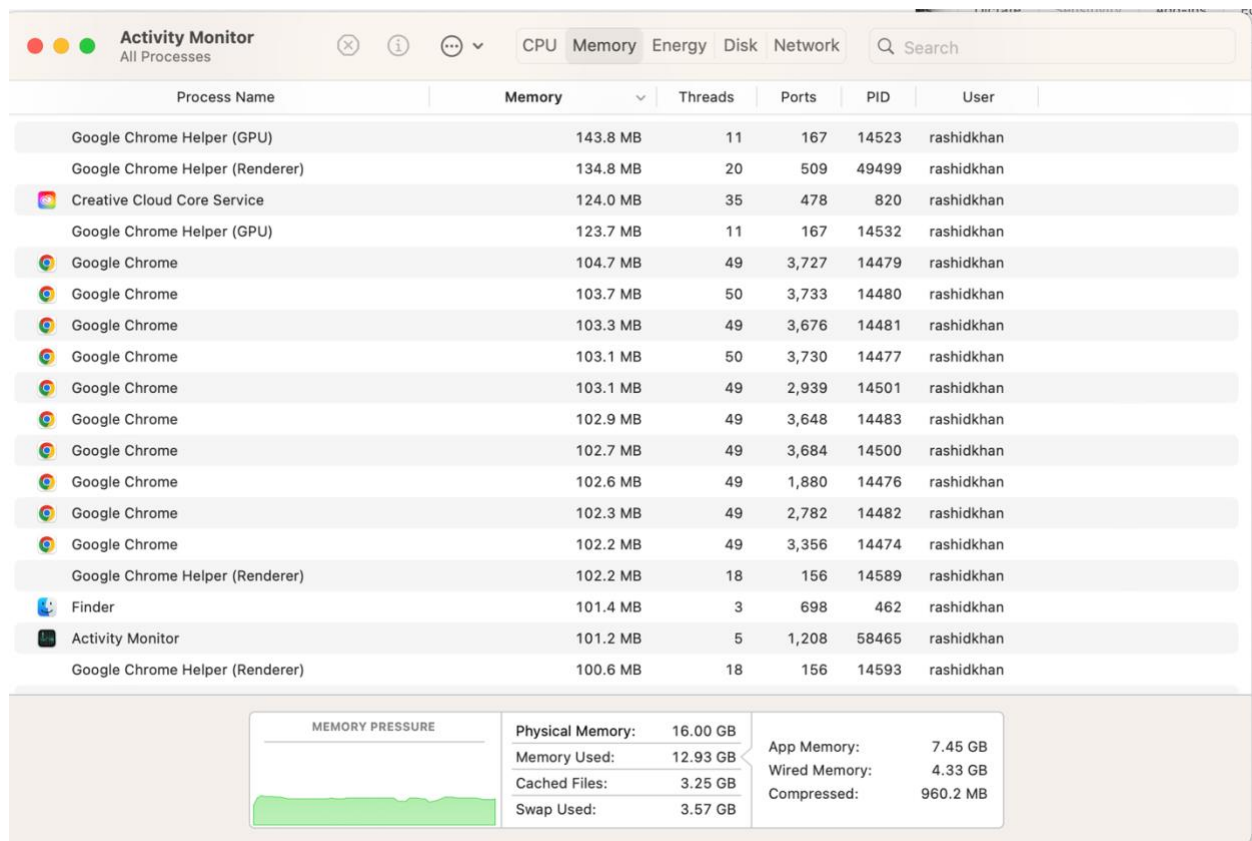


Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	% GPU	GPU Time	PID	User
AppleSpell	108.6	4:03:59.89	10	3	0.0	0.00	4369	rashidkhan
chromedriver	58.7	4:59.63	10	1	0.0	0.00	33579	rashidkhan
chromedriver	58.6	5:00.32	10	0	0.0	0.00	33581	rashidkhan
chromedriver	58.5	5:01.64	10	1	0.0	0.00	33578	rashidkhan
chromedriver	58.1	5:01.66	10	0	0.0	0.00	33580	rashidkhan
chromedriver	58.1	4:59.09	10	1	0.0	0.00	33577	rashidkhan
Google Chrome Helper (Rend...	48.5	4:57.86	18	23	0.0	0.00	33738	rashidkhan
Google Chrome Helper (Rend...	48.3	4:56.71	18	21	0.0	0.00	33737	rashidkhan
Google Chrome	48.0	5:12.60	49	7	0.0	0.04	33585	rashidkhan
Google Chrome	47.8	5:15.26	49	7	0.0	0.04	33584	rashidkhan
Google Chrome Helper (Rend...	47.8	4:55.05	18	23	0.0	0.00	33739	rashidkhan
Google Chrome Helper (Rend...	47.8	4:57.57	18	23	0.0	0.00	33740	rashidkhan
Google Chrome	47.5	5:12.16	49	9	0.0	0.04	33586	rashidkhan
Google Chrome Helper (Rend...	47.5	4:56.15	18	19	0.0	0.00	33736	rashidkhan
Google Chrome	47.3	5:13.52	49	7	0.0	0.04	33582	rashidkhan
Google Chrome	47.1	5:16.06	51	10	0.0	0.04	33583	rashidkhan

Also interestingly to be noted, that the amount of CPU usage was more when only one user was interacting with the website, however when 5 users were interacting, the CPU usage for each was almost 50%, which dropped to about 15% when 10 users were interacting:

Activity Monitor All Processes								
CPU Memory Energy Disk Network								
Search								
Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	% GPU	GPU Time	PID	User
AppleSpell	227.3	2:29:29.26	9	23	0.0	0.00	4369	rashidkhan
WindowServer	47.4	1:00:30.99	16	91	3.0	1:21:48.58	185	_windowserver
mysqld	45.7	3.01	28	136	0.0	0.00	95619	rashidkhan
kernel_task	18.4	57:30.31	275	1092	0.0	0.00	0	root
Google Chrome	17.3	21.20	49	11	0.0	0.00	93877	rashidkhan
Google Chrome	17.2	21.10	50	11	0.0	0.00	93884	rashidkhan
Google Chrome	16.2	20.66	50	10	0.0	0.01	93879	rashidkhan
python3.9	16.1	20.25	11	127	0.0	0.00	93499	rashidkhan
Google Chrome	15.6	20.49	50	10	0.0	0.00	93881	rashidkhan
Google Chrome	15.5	21.05	50	10	0.0	0.00	93880	rashidkhan
Google Chrome	15.3	20.82	49	10	0.0	0.00	93883	rashidkhan
Google Chrome	15.3	21.21	49	10	0.0	0.02	93882	rashidkhan
Google Chrome	14.6	20.92	49	10	0.0	0.01	93876	rashidkhan
Google Chrome	14.4	21.33	49	10	0.0	0.00	93878	rashidkhan
Google Chrome	13.6	20.89	50	9	0.0	0.00	93875	rashidkhan
Google Chrome Helper (Rend...	12.6	13.49	17	19	0.0	0.00	93995	rashidkhan
Google Chrome Helper (Rend...	12.1	13.47	18	24	0.0	0.00	93991	rashidkhan
Google Chrome Helper (Rend...	12.0	13.50	17	24	0.0	0.00	93996	rashidkhan

Regardless, the memory consumption for all of the process was very less. As we saw previously when there was only 1 user the memory consumption was less than Microsoft word. This time, the memory consumption even when 10 users were interacting with the website at the same time was still less than Microsoft word. Infact it was just a little more than finder (file explorer equivalent of mac).



We can definitely forcefully provide the processes more CPU and memory, but as for testing purposes, in the same environment we decided to keep it fair.

Performance test report:

1 users with 100.0 comments on each page:

Total time taken: 97.80700182914734 seconds

Average time taken by each user: 97.80700182914734 seconds

Average navigation time per user: 0.05697564506530762 seconds

Average comment submission time per user: 0.0427470850944519 seconds

5 users with 100.0 comments on each page:

Total time taken: 746.2191870212555 seconds

Average time taken by each user: 149.2438374042511 seconds

Average navigation time per user: 0.0995269416809082 seconds

Average comment submission time per user: 0.058340497493743905 seconds

10 users with 100.0 comments on each page:

Total time taken: 2793.3585352897644 seconds

Average time taken by each user: 279.33585352897643 seconds

Average navigation time per user: 0.18359161009788513 seconds

Average comment submission time per user: 0.1040536286830902 seconds

1 users with 200.0 comments on each page:

Total time taken: 171.61964678764343 seconds

Average time taken by each user: 171.61964678764343 seconds

Average navigation time per user: 0.05731476378440857 seconds

Average comment submission time per user: 0.0369250795841217 seconds

5 users with 200.0 comments on each page:

Total time taken: 1607.0972318649292 seconds

Average time taken by each user: 321.4194463729858 seconds

Average navigation time per user: 0.11077855215072634 seconds

Average comment submission time per user: 0.06921019182205199 seconds

10 users with 200.0 comments on each page:

Total time taken: 6112.1850616931915 seconds

Average time taken by each user: 611.2185061693192 seconds

Average navigation time per user: 0.21046369240283966 seconds

Average comment submission time per user: 0.12476666247844696 seconds

1 users with 300.0 comments on each page:

Total time taken: 282.0951118469238 seconds

Average time taken by each user: 282.0951118469238 seconds

Average navigation time per user: 0.06512971560160319 seconds

Average comment submission time per user: 0.04461323006947835 seconds

5 users with 300.0 comments on each page:

Total time taken: 2640.189147233963 seconds

Average time taken by each user: 528.0378294467926 seconds

Average navigation time per user: 0.12585064061482748 seconds

Average comment submission time per user: 0.08562619466781615 seconds

10 users with 300.0 comments on each page:

Total time taken: 10158.096586942673 seconds

Average time taken by each user: 1015.8096586942672 seconds

Average navigation time per user: 0.2413754865487417 seconds

Average comment submission time per user: 0.160799994913737 seconds

Comparing the performance test report with a baseline test:

1 users with 2.0 comments on each page:

Total time taken: 2.09378981590271 seconds

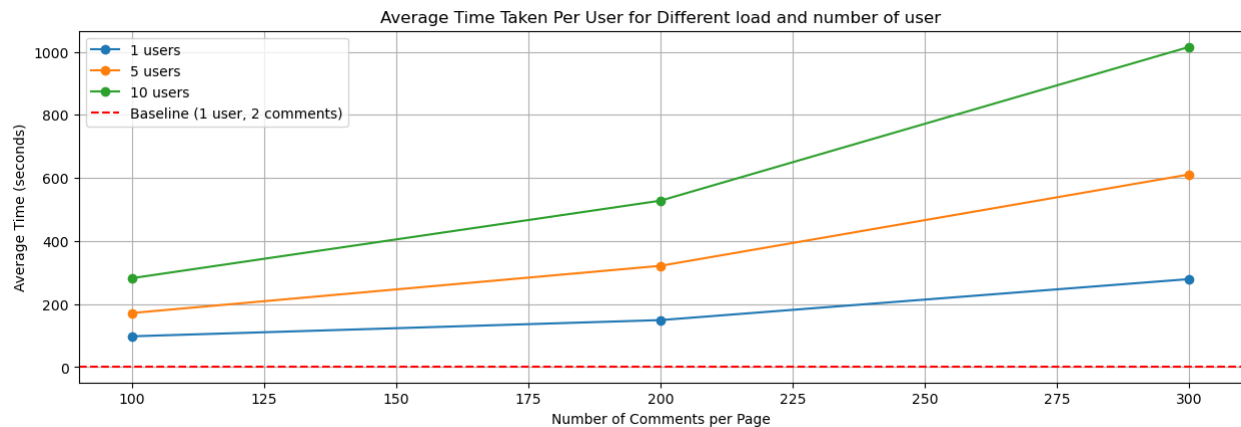
Average time taken by each user: 2.09378981590271 seconds

Average navigation time per user: 0.054038262367248534 seconds

Average comment submission time per user: 0.0420513391494751 seconds

The performance test which we decided to conduct focuses on the workability of the webpage in different conditions. We analyzed the time required to go to the next page, we included an implicit wait until the next page loads completely. We recorded the time it took for clicking the next button and the next page to load successfully. We did this repeatedly in a loop, and then averaged the time taken to go to next page in different conditions such as with low load, medium load and high load per page, and with different number of users. We also did the same for submitting a comment and waiting until the blog reappears on the screen after clicking submit.

Like how we've did previously, let's analyze the output using a graph. With no surprises as of now, we know that the comment submission time increases when either stress or scalability is increased. In the performance test that got validated once more.



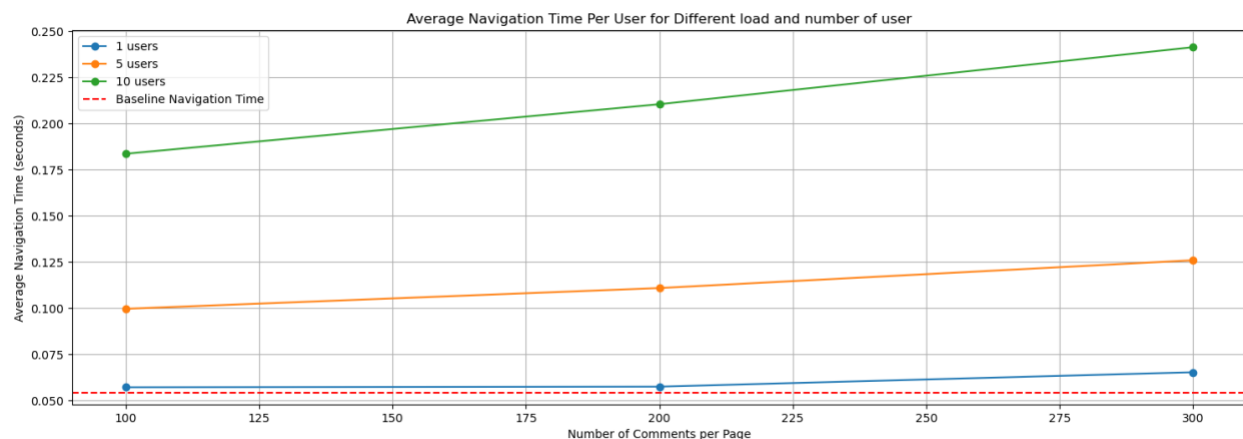
The following graph shows the average time taken per user to submit 100, 200 and 300 comments on each page by different number of users compared to a baseline. The baseline is 1 user submitting only 2 comments on each page. All the values are averages and thus they can be compared.

In the left section of the graph, we see that the Average time required for 1, 5 and 10 users to post 100 comments on each page. So basically, posting 500 comments on the webpage, during this the page was changed at every iteration and there was an explicit wait until the next page would load properly. As we can see for 1 user the time taken was about 100

seconds, comparing this with our stress test report, during stress test we were posting all the comments on the same picture at one time, so there was no next before each comment. Also, we were not waiting for the whole page to load during stress test, we were only waiting for the username and comment section to appear and the submit button to become clickable again. Thus, in our stress test 1 user didn't even take 50 seconds to post 100 comments on one page.

Now comparing with the baseline test, 1 user was submitting 2 comments on each page iteratively and waiting for the page to load completely. Thus, for 1 user with 100 comments on each page the average time will be higher, which we can see from the graph is true, as the number of users increases the time to post comments also increases. The conclusion here is same that as the number of users and comments per page increases, adding newer comments take more and more time. Just the way of performing this test was different. (stress test post "x" comments on one page and then goes to the next page, performance test post "x" comments in total, with "x/5" comment on each page, changing page "x" time.)

Now let's analyze the average time taken to change page as load and scalability increased.

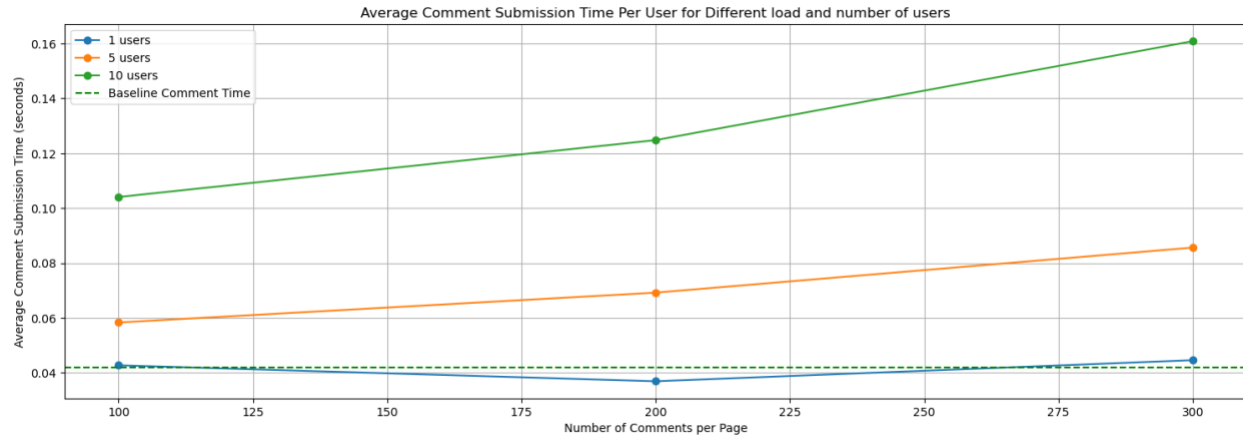


As we can see from the graph, for similar load condition, the average time taken to click next and wait for the page to load took more time as number of users kept increasing. For 1 user when they were soliciting on our website, they experienced top notch responsiveness. However, when 4 more users came on the website, each user started experiencing almost half the responsiveness the first one was experiencing when alone. And when 5 more users came and a total of 10 users were using the website, the responsiveness decreased more than three folds from when there was only 1 user. Interestingly, the responsiveness kept decreasing as the users kept posting more and more comments.

Bottom line here is, the time taken for page to load after when next was clicked, kept increasing as number of users, and number for comments on each page kept increasing. However, The website was very responsive throughout, there were no crashes, and even the

highest average time to load the next page was still only under 0.25 seconds, but overall the performance still degraded.

Now the average single comment submission time.



This graph shows how the average time required to submit a comment in different circumstances of load and scalability. As we can see from the graph, the average time to post a single comment kept increasing gradually as the number of users kept increasing.

Answers to questions:

Q1. What is Robot Framework, and how does Selenium fit inside Robot Framework?

Ans: Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development. It is a keyword-driven testing framework that uses tabular test data syntax. The Robot Framework library, Selenium Library offers a list of keywords for using Selenium WebDriver to communicate with web browsers. It functions as a wrapper of Selenium, facilitating the creation of automated web application testing using the high-level, keyword-driven Robot Framework syntax. Because to this connection, we can use Selenium's power in the Robot Framework's structured environment to automate browser activities as part of their acceptance testing suites.

(<https://www.browserstack.com/guide/robot-framework-and-selenium-tutorial>,
<https://github.com/robotframework/SeleniumLibrary>)

Q2. How could you use Selenium to perform record and playback. That is, how can it record a manual test, and playback (or run again) that manual test?

Ans: For recording and playback capabilities, Selenium offers the Selenium IDE. Using that we can record our interactions with an online application and store them as automated test scripts using this browser extension, which is available for Firefox and Chrome. The application can then be tested by playing back these recorded scripts an unlimited number of times.

1. Install Selenium IDE: Add it as an extension to Firefox or Chrome.
2. Record a Test: Open Selenium IDE, start a new project, and press the record button. Perform your manual test steps on the web application. Selenium IDE will record these actions.
3. Stop and Save: After completing the test steps, stop the recording and save the test script.
4. Playback: Use Selenium IDE to play back the recorded test script to automate the execution of the test steps.

(<https://www.browserstack.com/guide/record-and-playback-in-selenium#:~:text=Click%20on%20%E2%80%9CCreate%20a%20new,and%20create%20new%20test%20cases.&text=To%20create%20a%20new%20test,Playback%20for%20the%20test%20cases.,> <https://www.selenium.dev/selenium-ide/>)

Q3. In this assignment we used Selenium WebDriver. What is Selenium IDE?

Ans: As we talked about this in the previous question, the Selenium IDE is an open source record and playback test automation for the web. We just have to install the extension on our browser. And then we can record a manual test and then the IDE will perform that test for us as many times we want.

Q4. What is Selenium Grid?

Ans: A component of the Selenium Suite called Selenium Grid makes it possible to execute Selenium test scripts concurrently across several operating systems and browsers. It allows for distributed test execution, in which many environments are used to execute the

tests concurrently. This makes it easier to test under multiple circumstances without requiring physical access to the computers or browsers, and it also helps to greatly reduce the amount of time needed to run comprehensive test suites.

The Selenium Grid is made up of a central hub that distributes test orders to several nodes, each of which stands for a distinct operating system, browser, or version. This configuration is a crucial tool for cross-browser and cross-platform testing since it allows testing web applications in several contexts from a single central location.

(If we would know about this earlier we would do scalability test using this)

(<https://www.selenium.dev/documentation/grid/>)