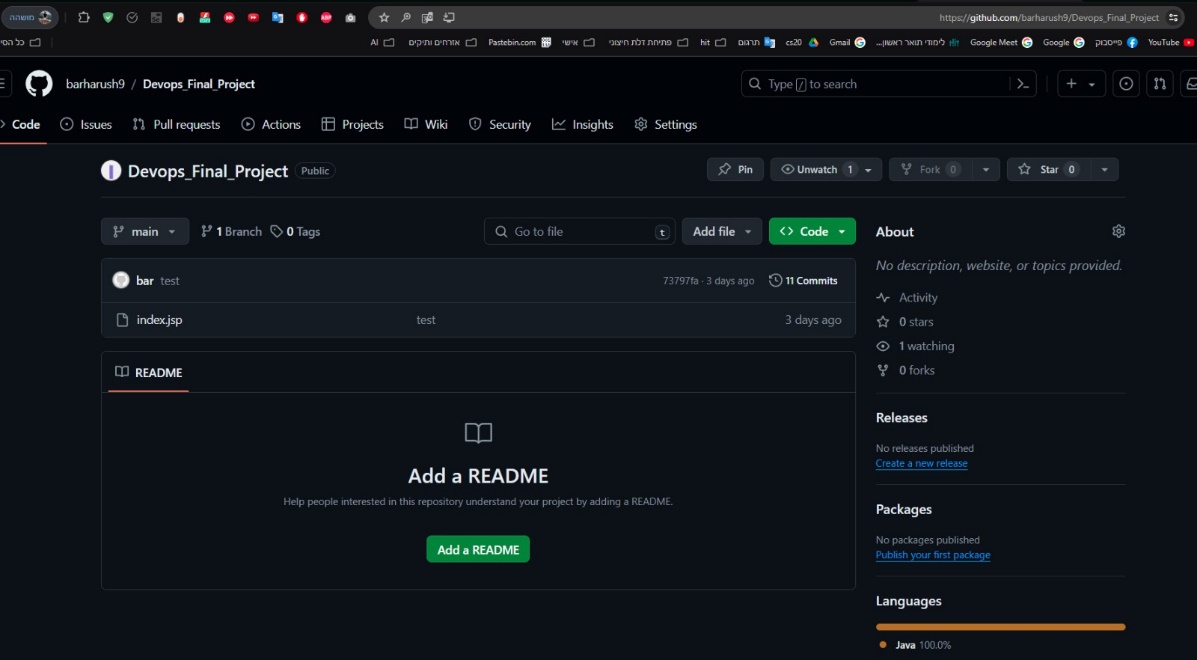
**Devops Final Project**

**Names:**

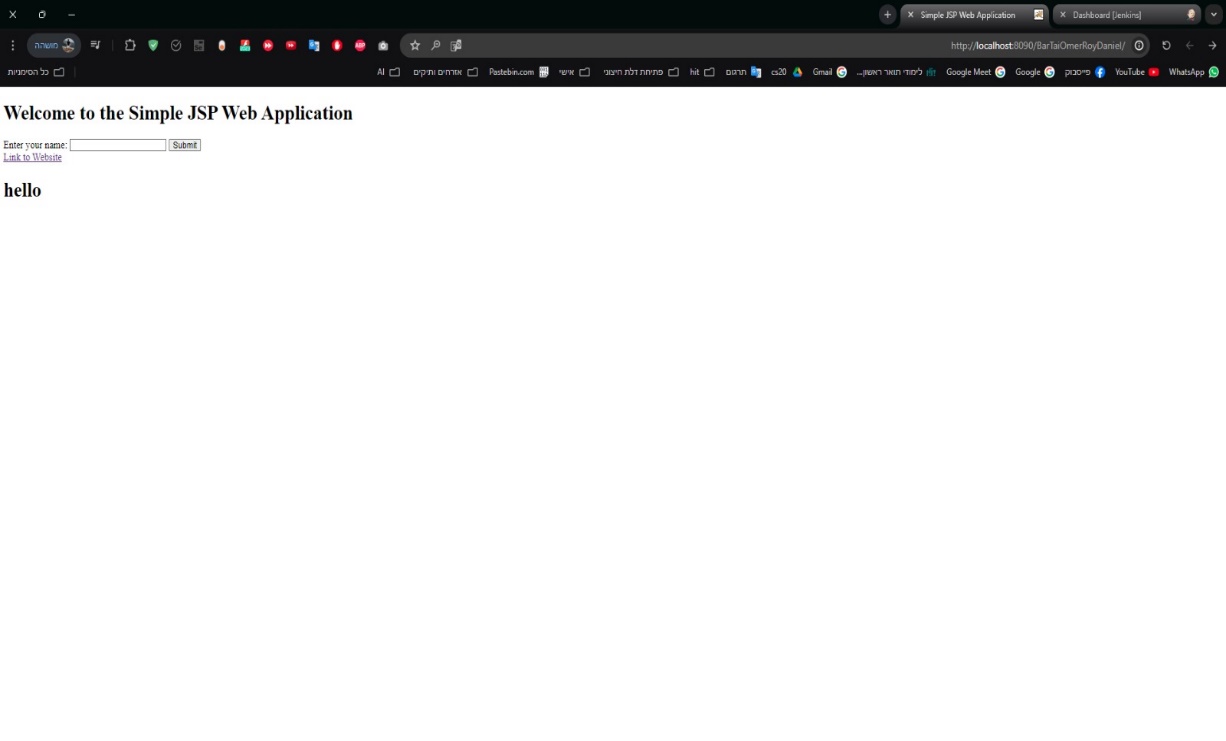
* Bar Harush 319023461
* Tair Schapira 210015848
* Roy Shlomo Chen 315998971
* Omer Sruia 318911161
* Daniel Zagoury 209380021

**Introduction:** We created a simple JSP based web application with the required features. By integrating CI we streamlined the integration of code changes into a shared repository. Each change triggers an automated build and testing sequence using Jenkins. The CD implementation entailed creating a Jenkins job that automatically deployed the web application to a Tomcat server. In addition, we exposed the application to the internet using a public IP address. We then conducted a 4-minute load test using gatling to simulate up to 90 concurrent users, ensuring the application could handle the expected user load without performance degradation. In addition, we performed a 4-minute stress test with gatling to try and push the application beyond its normal operating limits.

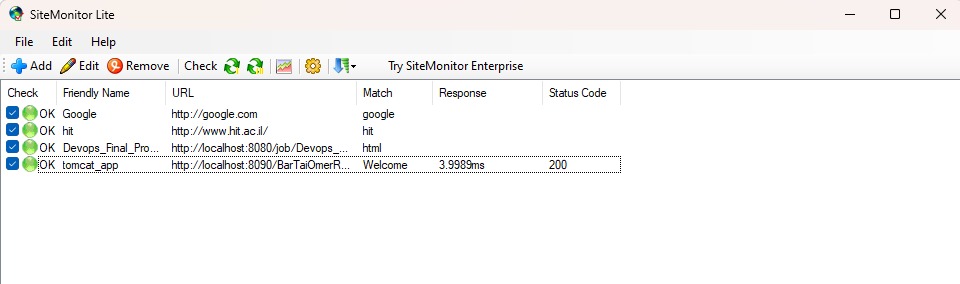
1. The JSP file (index.jsp) is attached to the email.
2. Screenshot of your GitHub:



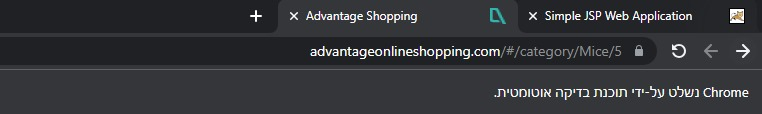
1. Screenshot of your app in Tomcat:



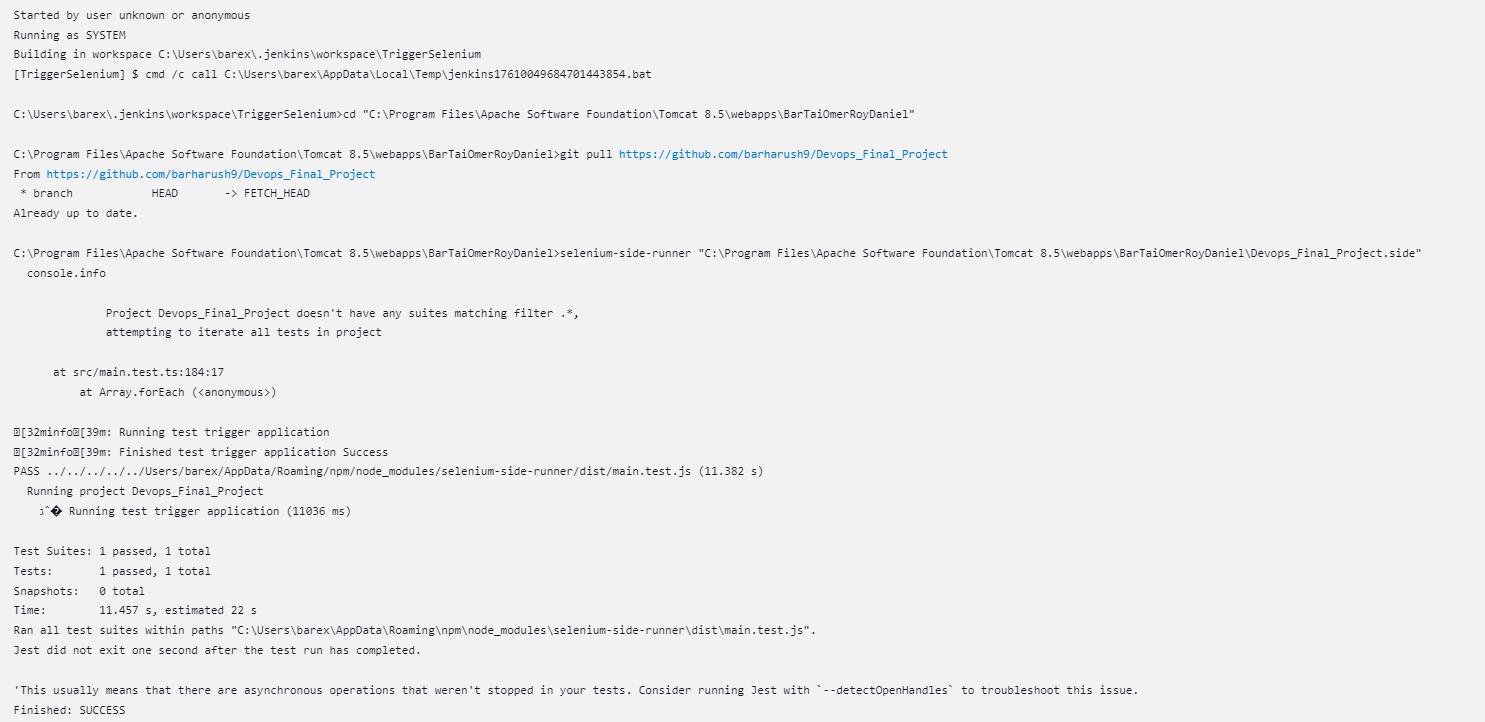
1. We set up availability monitoring for our DevOps project using "SiteMonitor Lite" to ensure continuous uptime and performance. The primary focus was on the URL http://localhost:8090/BarTaiOmerRoyDaniel hosted on our Tomcat server. This URL is critical for our internal application, and monitoring it helps us detect any downtime or performance issues promptly. To automate this process, we run SiteMonitor Lite through a Jenkins job, which triggers the monitoring tool. By tracking response times and status codes, we gain insights into the application's performance, helping us identify and address potential bottlenecks quickly.



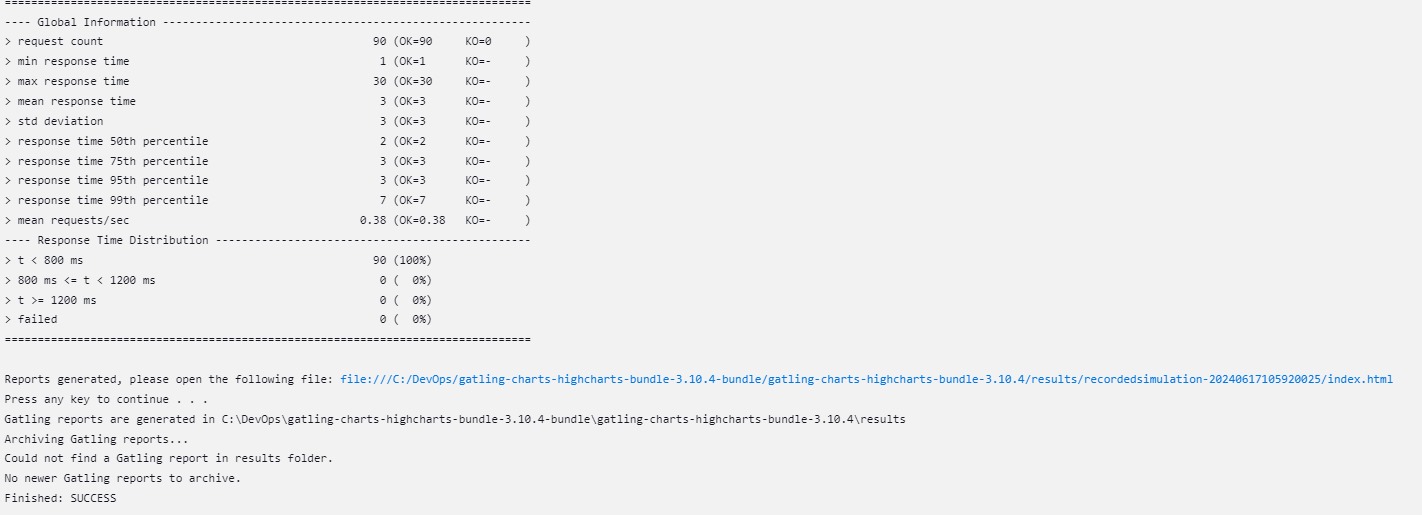
1. Selenium IDE file is attached to the email.
2. Selenium IDE screenshot of passed run:



The following is a screenshot of the console output of the Jenkins jobs that triggered the Selenium IDE:



1. Description of HAR scenario in words: We performed actions on our website as a user. We entered a name and clicked the submit button. This action displayed a greeting message with our name. By recording these actions, we documented the interactions possible on our site and saved the file. This way, we can later test the system with a large number of users to see if it can handle the load and identify any points of failure.
2. The HAR file is attached to the email.
3. **Load Test Screenshot**: The load test processed 90 requests with a mean response time of 3 ms and no recorded failures. Overall, the application's performance under load was efficient and stable, with consistently low response times.



v

**Stress Test Screenshot**: The stress test processed a total of 5208 requests with an impressively low mean response time of 1 ms and no recorded failures. Overall, the application's performance under stress was highly efficient and stable.

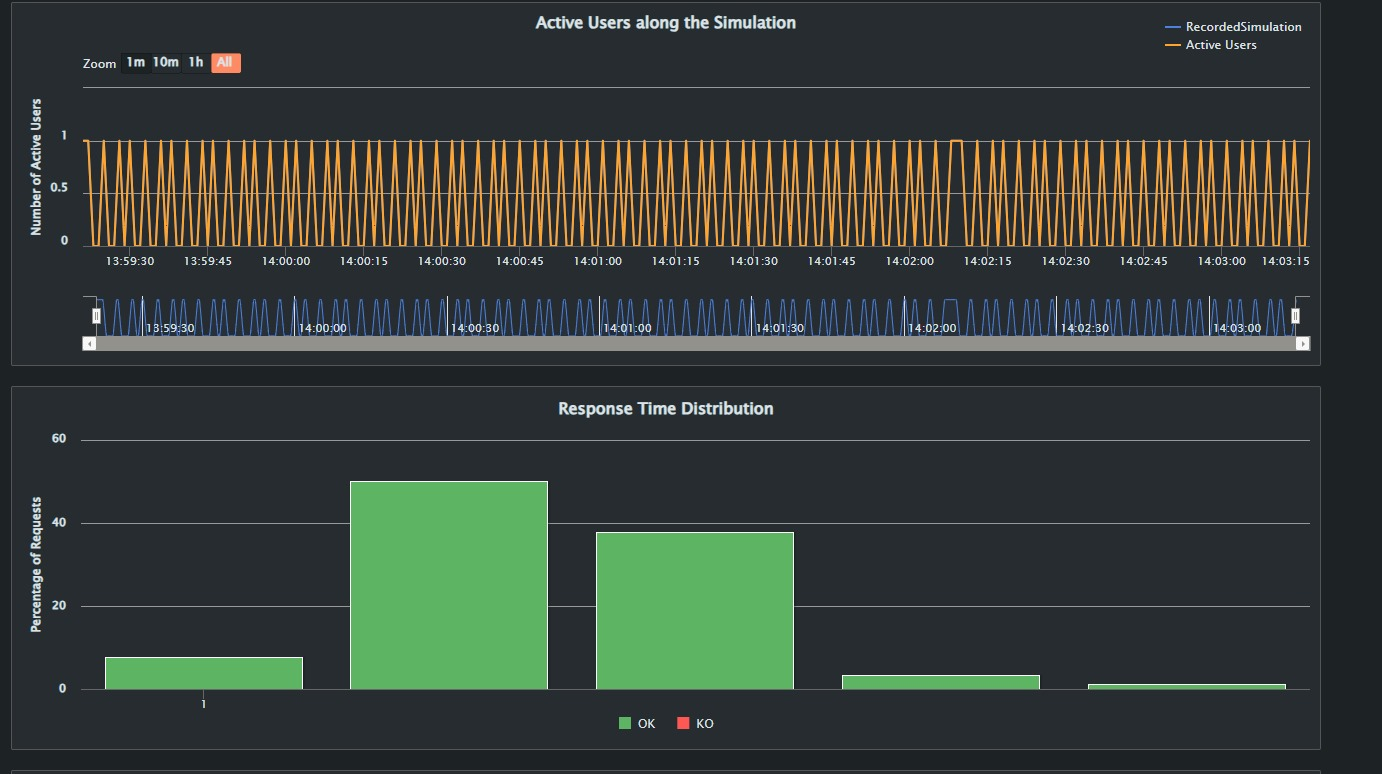
A close-up of a white sheet

Description automatically generated

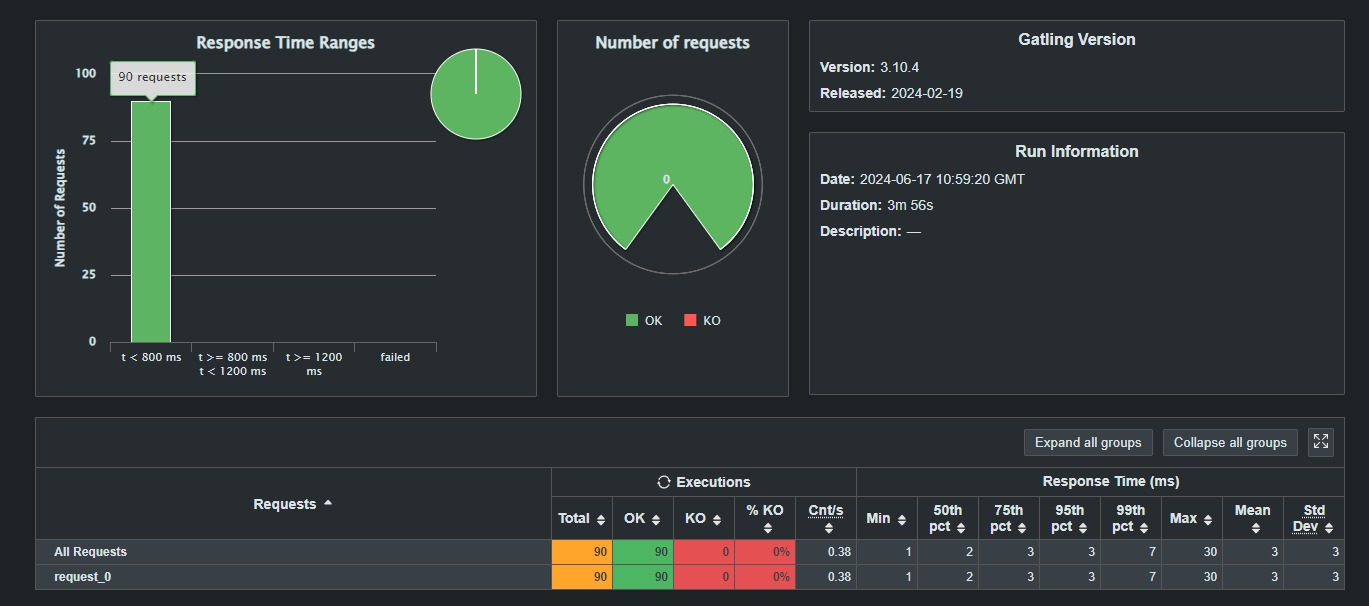
v

1. **Load Test Graphs:**

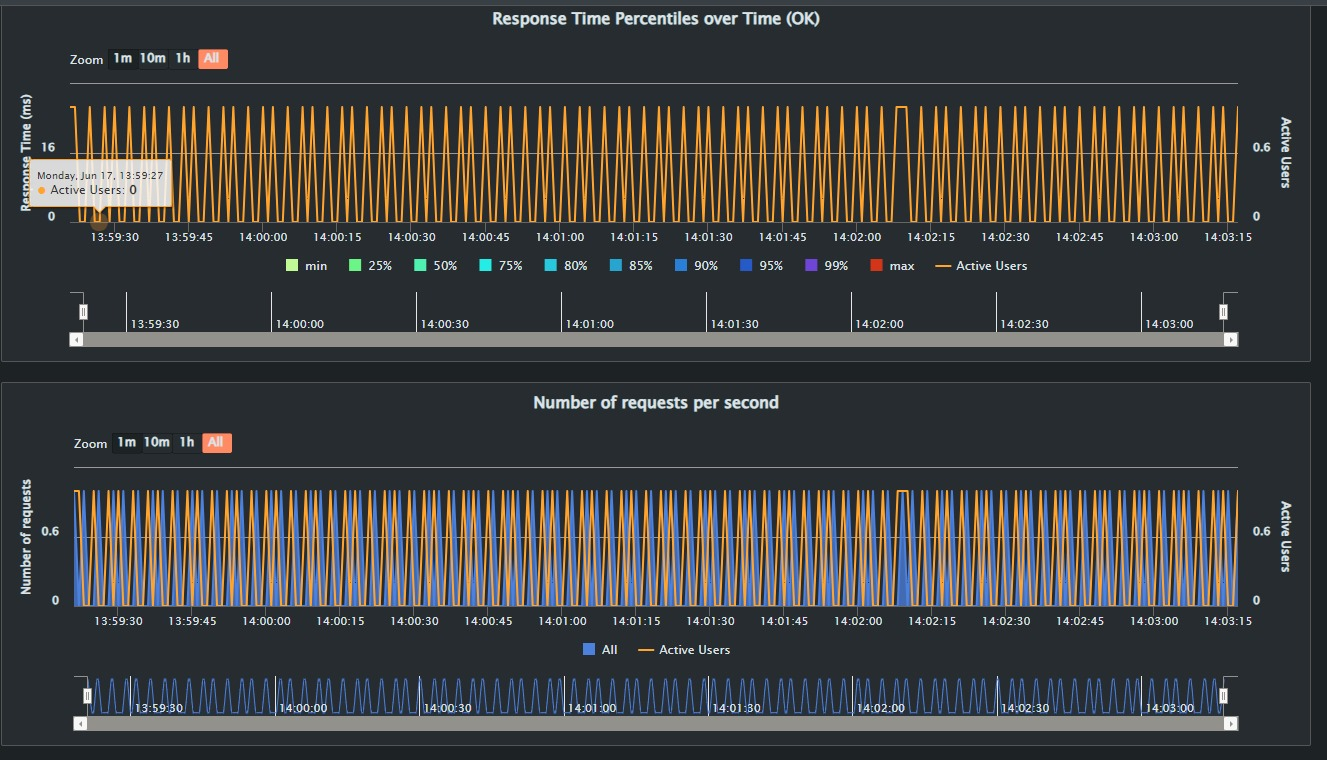
## The "Active Users along the Simulation" graph provides a visual representation of the number of users interacting with the system over time. The consistent load of around 0.6 users indicates that the system was under a steady stream of user activity, which is essential for accurately assessing the system's performance under load.



The “Response Time Ranges” shows that all 90 requests had a response time of less than 800 milliseconds. The “Number of Requests Pie chart shows that all 90 requests were successful. The “Execution table” provides detailed statistics for the request response times. A large portion of requests (around 50%) had a response time of 2 milliseconds and a large number also at 3 milliseconds. The distribution shows that the system is capable of handling requests very quickly, with all response times falling well within the acceptable range (below 30 milliseconds).



The “Response Time Percentiles over Time” graph shows the response time percentiles for successful requests over the duration of the test. Each time gap (the x-axis) shows the amount of active users that fall withing that time range. The orange line shows the active users and the amount per second of active users remains consistent. The “Number of Requests per Second” shows the rate at which requests were made per second over the duration of the test. This again shows that there were about 0.6 users per second (90 users in total over a duration of 4 minutes).



Finally, the load test results demonstrate that the system performed exceptionally well under the simulated load of 90 users. The system consistently handled requests with very low response times and no failures, maintaining stability and efficiency throughout the test duration. The steady request and response rates, combined with the low and consistent response times, indicate that the system is capable of handling similar loads in real-world scenarios.

**Stress Test Graphs:**

The **“Response Time Percentiles over Time”** graph displays response time percentiles over the duration of the test. The response times are consistently low, with the 99th percentile and maximum response time remaining below 35 ms.There is minimal variation in response times across different percentiles, indicating stable performance. The graph shows that the system can handle a consistent load without significant delays or performance degradation.

תמונה שמכילה טקסט, תכונות מולטימדיה, תוכנה, צילום מסך

התיאור נוצר באופן אוטומטי

The **“Number of Requests per Second”** graph shows the number of requests made per second over the duration of the test. The number of requests remains stable with a consistent rate of 22 requests per second. The system handled a constant load of 22 active users. There are no significant spikes or drops in the request rate, suggesting efficient processing of requests. The graph clearly demonstrates the system’s ability to maintain a steady request rate under stress.

תמונה שמכילה טקסט, תוכנה, תכונות מולטימדיה, צילום מסך

התיאור נוצר באופן אוטומטי

The **“Active Users along the Simulation”** graph shows the number of active users over time during the simulation. This is another representation of the stress test indicating the systems stability. The number of active users remains consistent at around 22 throughout the test.

תמונה שמכילה צילום מסך, טקסט, תכונות מולטימדיה, תוכנה

התיאור נוצר באופן אוטומטי

This table summarizes the execution and response times for the stress test. A total of 5280 requests were made, all of which were successful, resulting in a 0% failure rate. The system maintained a steady request rate of 22 requests per second. The response times were extremely low, with the 50th percentile at 1 ms, the 75th percentile also at 1 ms, and the 99th percentile at 2 ms. The maximum response time recorded was 35 ms, with a mean response time of 1 ms and a standard deviation of 1 ms, indicating consistent and efficient performance throughout the test.

A screenshot of a computer

Description automatically generated

In conclusion, the stress test results demonstrate that the system can handle a significant and consistent load effectively. All graphs indicate stable performance with consistently low response times and no significant spikes or drops in request rates. The number of active users remained steady throughout the test, ensuring that the load was consistent. This stability and efficiency suggest that the system is well-optimized and reliable, capable of maintaining high performance under stress.