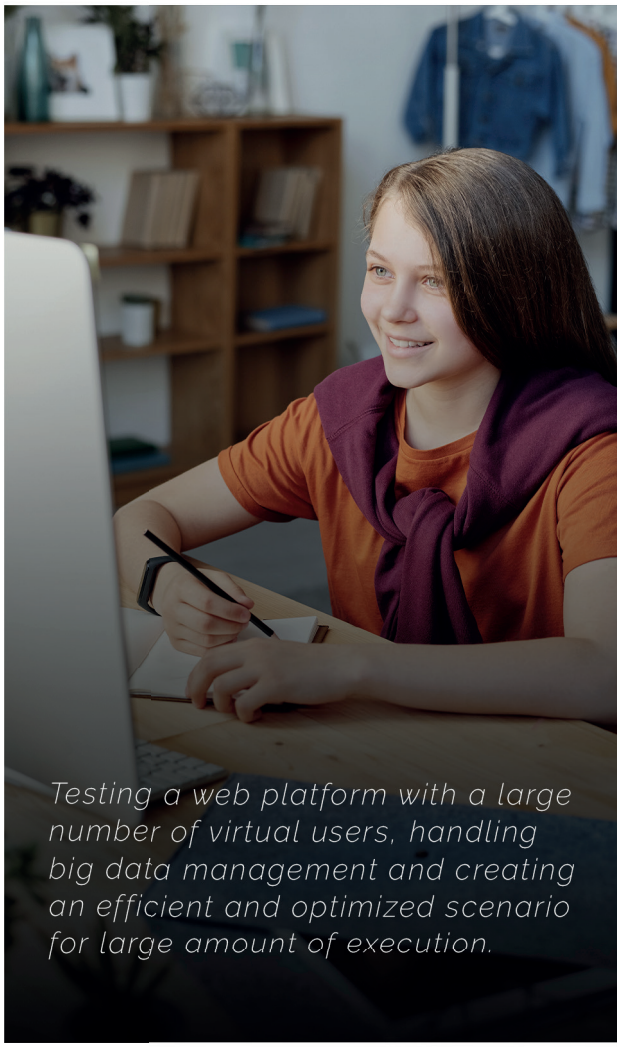
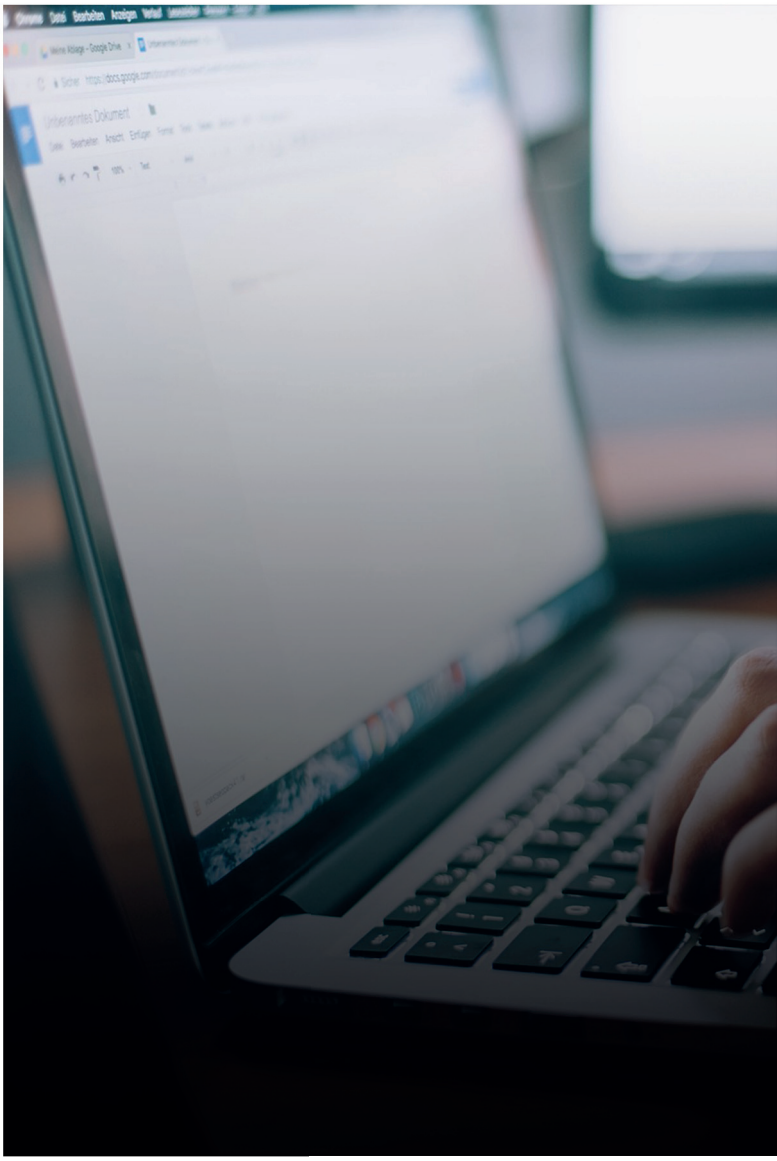

# INTRODUCTION SUMMARY:

EBA is an online education platform that allows students to take distance education during the pandemic. It is government funded and operates as a K-12 learning management system. However, due to the unexpected epidemic, the development of EBA was urgently decided and a large number of users had to be served in a short time. For this reason, we tested the stability and quality of the improvements by performing load tests.

We can clearly say that the load tests that EBA wants us to do are not the same size as the work size we usually do. Because EBA is a government-funded K-12 learning management system, it has a much higher user density than a standard website. The biggest challenge for us was to process and report the big data from Amazon's servers and create a manageable and optimized scenario for large amount of execution. In working to overcome these challenges, guiding EBA to pinpoint its problems was the ultimate goal throughout the process.



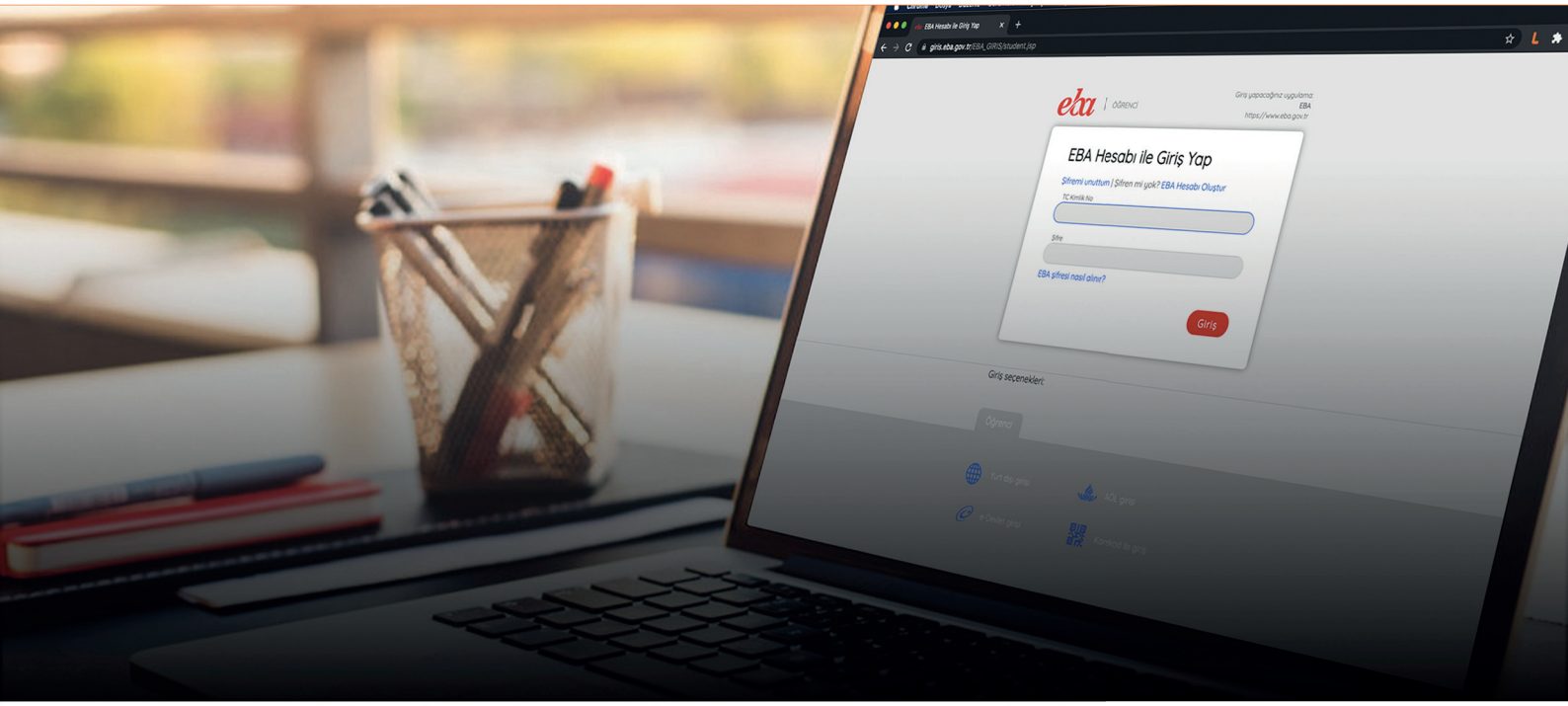
# INTRODUCTION TEXT:



The social isolation we experience due to the epidemic in the world has caused many difficulties in the education sector as well. As a result of this situation, the Turkish government has decided to provide Education Informatics Network (EBA) , an online education platform that enables remote classes and exams by teachers and students across the country. EBA is a platform that existed before the pandemic. It used to be an online library platform to help teachers and students access information easily. Available on the web and on both mobile platforms; Android and iOS. with social distancing becoming an important norm, with the government and the Ministry of Health's decision not to open schools, the need for an urgent distance education platform arose. Thus, the process of transforming EBA into an online education platform has begun. The platform is planned to serve as the K-12 learning management system, which makes up the bulk of the population expected to have over 1 million concurrent users. Thus, at a certain point in the development process, the importance of load testing. the platform became clear.

As we reached an agreement after our negotiations with EBA, the project processes started. A team of 4 people consisting of 2 backend developers and 2 test engineers was formed.

In addition, it was decided to use Loadium as the load testing tool, all of which were developed internally in our product to perform performance / load tests on software systems while presenting the results with a real-time report screen. Moreover, while the test engineers were responsible for the development of the user scenario using the open source Apache Jmeter technology, it was of great importance that they could imitate the real user behavior as much as possible, and that the report obtained after the large amount of data flow was clear and understandable. Also, the backend developers were responsible for the integration of Loadium with Amazon servers, which means optimizing server connections and ensuring the stability of the data flow from Amazon to Loadium's report screen.



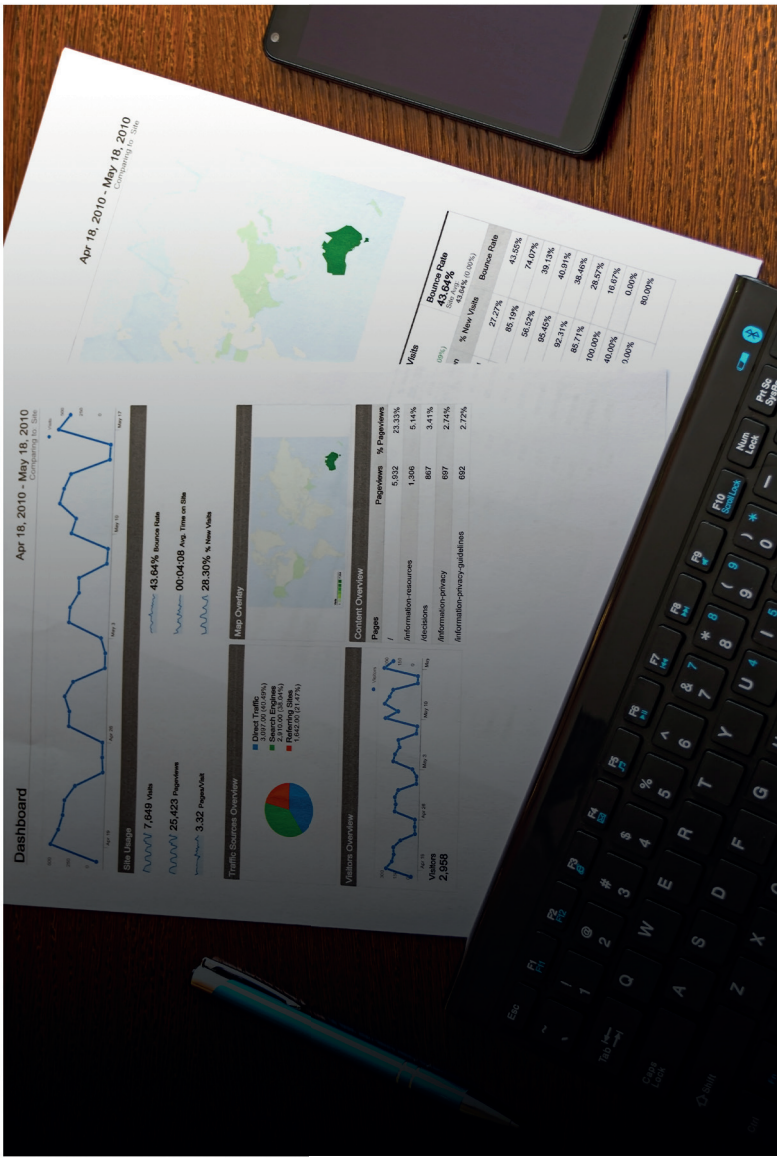

Initially, a document for the test case was obtained from EBA and a date was set for the load testing to be carried out. The scenario was designed with an end-to-end principle; Users log in, watch lectures, take exams, review exam results and sit idle in the system. However, the linearity and length of the scenario negatively affected the efficiency of the test. In such scenario flows, if a bottleneck occurs at one point on the servers, this bottleneck causes all virtual users to be adversely affected, thus reducing the efficiency of load testing. However, there were also concerns about data transfer, so it was decided to keep the script simple for the first run only. Since this is a government funded education website, it was expected to have a much larger number of users compared to our regular customers, so testing had to be done accordingly. More users means more threads will be required to emulate the behavior, resulting in large amounts of data being processed. with this in mind, we have configured our systems by increasing the number of servers responsible for data collection to have more data transmission capacity than usual.

Managing Amazon servers was another challenge; this meant simultaneously checking hundreds of Amazon servers to perform the same action. Testing starts only after all servers are ready. Considering that 600-1000 servers would be required to run the test, this was another concern for us.

We started testing the system using the first script with 250,000 users and increased the amount of userswith each successful test. For the third run of the first scenario, we used 625 servers and it took longer than usual to start. Usually tests with 100 to 400 servers start in about 4 to 7 minutes, but this test took about 12 minutes to start and this spike started to negatively affect the tests. In addition, a bottleneck was detected in the report screen, it was determined that the report data was reflected on the screen with a large delay

After these feedbacks and observations, backend optimization studies started, aiming to be completed before the second load test. The first and an important improvement was for the real-time report display to provide a smooth data flow for the data of the 625 servers. We had 2 servers responsible for transferring data from Amazon to our report screen,and each of these servers was doing what we call &"indexing" and sending queries at the same time, which was not ideal. What we did was split each of these operations into a specific server, so that one server was doing the indexing and the other was doing the querying. In addition, another improvement is that we aimed to reduce the time spent for both initialization and termination by developing a structure that will dynamically increase the number of parts in the Elasticsearch indexes by calculating the amount of load that will come to the index.

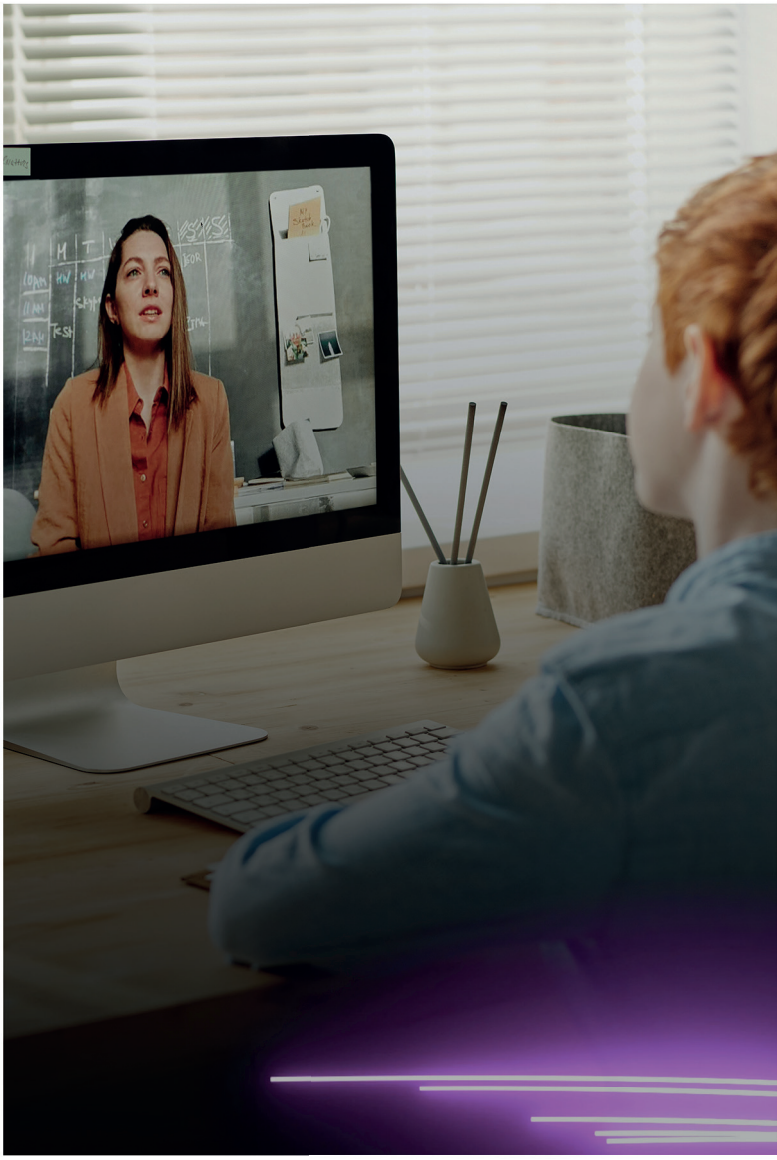
We also suggested a different approach to the scenario, dividing users into two categories, as previously pointed out by the issues with the first scenario; some of them would fulfill the demands related to the exam, the other part would fulfill the demands related to the course. We've also suggested changing the end of the script for a dynamic behavior instead of letting users sit idle. Another takeaway from the first implementation was that EBA's website noticed the unnecessary load placed on the login servers.





The login screen is a screen that users encounter only once, but other screens are visited many times by users. So, if a user has reached the end of the scenario; for example, if the user finishes all the exams (or lessons) included in the scenario, that user will solve all the exams again without logging in again. This design allowed us to greatly increase the efficiency of testing by applying dynamic load to the website, while avoiding unnecessary load on the login screen. Also, in the first scenario there was no waiting time between requests (think time available in real user behavior), so it was specifically added for the time a user took a quiz or attended a lecture.

After completing all these improvements, we restarted the second test with 625 servers and the efficiency gap was clear for both the backend and the scenario. The test started almost 20% faster than the first test. While the transferred data could still be considered too large, the latency of the real-time report display was reduced and more accurately reflected the actual state of the test. Also, with each test iteration and feedback for the backend, development continued to grow, achieving 60% faster test launch and we were able to present a lag-free report screen for 625 servers.



As a result of reducing the backend problems by solving them and preparing a more efficient scenario, a more efficient working process was continued to analyze the problems of EBA systems. For example, even though we reduced the load on the login screen in the scenario, EBA servers were still having login issues, so this result helped them identify and fix the problem. Likewise, problems with other departments such as exam and lecture screens were also discovered, so these results led them to work more efficiently in certain areas of their systems, thus increasing the quality of EBA education service offered to many students and teachers.

## ABOUT LOADIUM

Loadium started its journey in 2009 as an IT company, later specialized in software test automation solutions for mobile, web and desktop applications to ensure great digital experiences with the highest quality. Testinium is our flagship software test automation product offered on cloud and on-premises deployment. Loadium is our load testing solution which is based on Apache's Jmeter and SBox is our image transmission automation solution which helps automating tests for set-top boxes including any image media player.

Learn more at: **www.loadium.com**