UDC 004.415.2
DOI https://doi.org/10.32782/2663-5941/2024.4/22

Oleshchenko L.M.
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Burchak P.V.
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

SOFTWARE SYSTEM ARCHITECTURE DEVELOPMENT FOR INTELLIGENT ANALYSIS OF WEB APPLICATION PERFORMANCE METRICS

Today the number of web applications that process large amounts of data is increasing, which creates new challenges for developers and users. Web applications that handle big data are becoming an essential part of business, government, and everyday life, enhancing efficiency, accuracy, and speed of decision-making. This brings numerous problems, such as ensuring data security and confidentiality, efficient processing and storage of large volumes of information, and the need for continuous monitoring and optimization of web application performance. Resource management and energy efficiency are becoming particularly relevant in the context of energy savings. Solving these problems requires the use of the latest technologies, such as intelligent monitoring and analysis systems, which help ensure the stable and efficient operation of web applications in conditions of constantly increasing data volumes and task complexity. Addressing web application performance issues through intelligent monitoring systems will enable developers to quickly and accurately identify bottlenecks and optimize code, which in turn ensures a better user experience and reduces maintenance costs. Unresolved issues include the complexity of integrating new technologies into existing systems, the need to consider various factors affecting performance, and ensuring a high level of security and data confidentiality during monitoring.

The article analyzes existing software solutions such as Google Lighthouse, Apache JMeter, New Relic, Dynatrace, GTmetrix, Pingdom, AppDynamics, WebPage Test, Sentry, and LoadRunner, their functional capabilities, main advantages, and disadvantages. It examines the possibilities of using machine learning and artificial intelligence technologies in the considered software systems. Based on the analysis, a software system architecture is proposed for analyzing the performance of web applications written in JavaScript, which allows for the collection of numerical data on the factors affecting the performance of the web application, performing regression analysis to determine the assessment of the influence of factors, clustering and classification of the processed data for the correctness of the allocation of recommendations for developers, which must be used in order to improve the performance of the web application and reduce the load on the web server. According to the conducted research, the use of machine learning methods in software systems for web application performance analytics can increase the performance of web applications by an average of 20%.

Key words: *software system architecture, web application performance evaluation, JavaScript, machine learning, AI, regression analysis, clustering, classification, neural networks.*

Introduction. Problem Statement. Monitoring the performance of web applications is critical to ensuring the stable operation of software on the Internet. In modern realities, when saving resources becomes critically important, optimizing the performance of web applications allows to reduce the load on servers and infrastructure, which, in turn, contributes to reducing energy consumption. This is especially relevant for developers who work with the JavaScript language, as web applications in this programming language are extremely popular both in Ukraine and around the world. According to statistics, more than 70% of modern web applications use the JavaScript language for the development of the client part, a significant share of server solutions is also based on Node.js. In Ukraine, this trend is supported by the wide implementation of web technologies in various industries, including the financial sector, education, e-commerce. Using machine learning (ML) techniques to evaluate the performance of web applications allows not only to identify bottlenecks and problems in the execution of the code, but also to predict potential performance problems, which allows to take early measures to eliminate them. The development of such a software system is not only relevant from a technical point of view, but also contributes to the overall efficiency of the use of resources, which is an important contribution to the preservation of the country's energy resources.

The main goal of the article is designing a software system architecture for intelligent analysis of the performance of web applications written in the JavaScript language.

Related research. In the article [1] authors analyze various techniques, including caching, compression, Content Delivery Networks (CDNs), and tools such as WebPageTest and YSlow. The research aims to help Web developers and performance engineers optimize their websites for improved search engine rankings.

The paper [2] presents a research on optimizing trace visualizations for microservices performance analysis. The authors investigate current limitations in trace visualization tools and propose novel techniques for effective performance analysis.

The research paper [3] explored the use of Dynatrace monitoring data to generate performance models for Java EE applications. The research aimed to improve the understanding and optimization of application performance by leveraging monitoring data and developing accurate performance models.

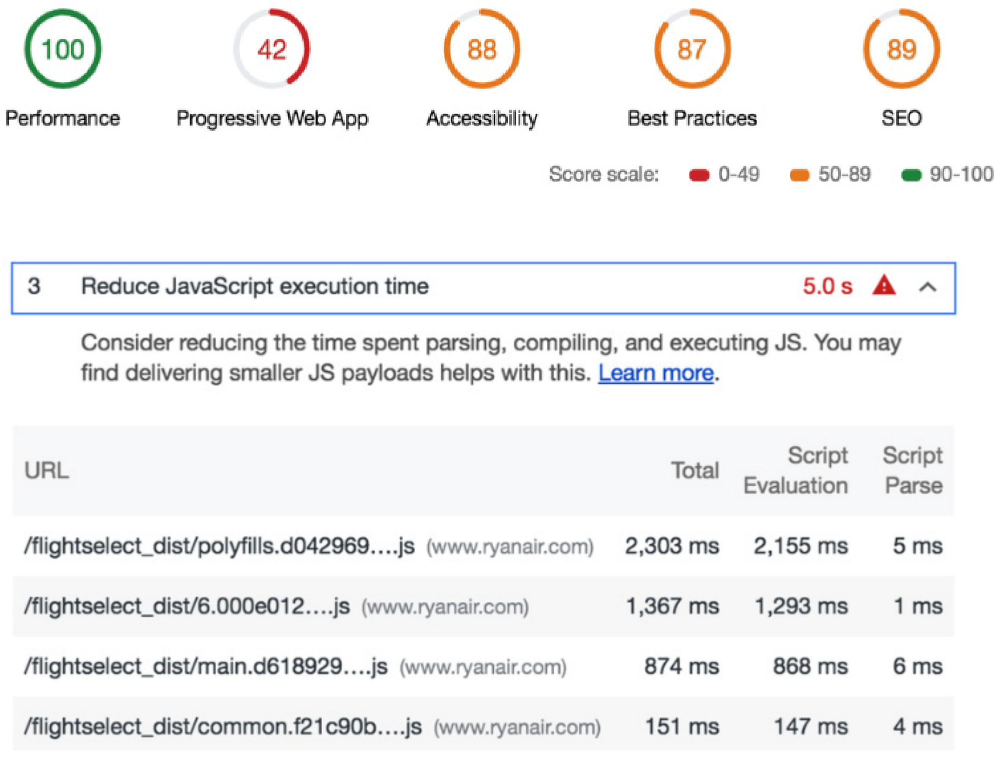
The research [4] investigates the process of storing and managing data in the client side of web applications, focusing on software methods for local state management, and demonstrates that using an atomic approach to data state management with the React Context API reduces data processing time by 17% compared to popular libraries such as Redux, MobXState-Tree, and Recoil, thereby optimizing state management and improving overall performance.

Existing software solutions analysis

Google Lighthouse is an open source software tool that evaluates performance, accessibility, SEO and other aspects of web applications. Google Lighthouse performs automated audits to evaluate various indicators, works on the basis of Chrome DevTools (Fig. 1). Google Lighthouse specializes in client-side performance analysis, providing detailed reports on page rendering, load times, image optimization, resource size, and overall page speed.

Google Lighthouse has a limited ability to analyze server performance, it can provide general recommendations for download speed but does not provide in-depth analysis [5].

Apache JMeter is a powerful web application load testing and performance measurement tool that uses the Java language to create load scripts and can test various protocols, including HTTP, HTTPS, SOAP, FTP. JMeter is less suitable for analyzing client-side performance. The main application of the tool is server load and performance testing. JMeter is used for load testing servers, including web servers, APIs, and databases. The tool can create a heavy load for



3 Reduce JavaScript execution time 5.0 s

Consider reducing the time spent parsing, compiling, and executing JS. You may find delivering smaller JS payloads helps with this. Learn more.

URL	Total	Script Evaluation	Script Parse
/flightselect_dist/polyfills.d042969….js (www.ryanair.com)	2,303 ms	2,155 ms	5 ms
/flightselect_dist/6.000e012….js (www.ryanair.com)	1,367 ms	1,293 ms	1 ms
/flightselect_dist/main.d618929….js (www.ryanair.com)	874 ms	868 ms	6 ms
/flightselect_dist/common.f21c90b….js (www.ryanair.com)	151 ms	147 ms	4 ms

Fig. 1. Google Lighthouse monitoring results

testing the scalability and performance of the backend of a web application [6].

New Relic is a cloud platform for monitoring the performance of web applications in real-time. Uses agents integrated with server-side programming languages (Java, Python, Node.js, Ruby, etc.) to collect data. New Relic offers front-end performance monitoring tools, including Real User Metrics (RUM), which show how real users interact with a web application. New Relic provides deep back-end monitoring, including server, database, and infrastructure performance. This allows the identification of bottlenecks and performance problems on the server side [7].

Dynatrace is a commercial application performance monitoring tool, including a detailed analysis of each transaction, using AI to analyze performance, and agents to collect data from various system components. Dynatrace offers comprehensive client-side monitoring, including analysis of page load times, user interactions, and JavaScript performance metrics [8].

Dynatrace provides deep back-end analysis, including monitoring of servers, databases, microservices, and containers. AI algorithms help to automatically identify problems and causes of productivity decline (Fig. 2).

GTmetrix is a tool for analyzing the speed of loading web pages and providing detailed reports, which uses an API to receive data from Google Lighthouse and other sources. GTmetrix focuses on client-side performance analysis, using Lighthouse and PageSpeed tools to provide detailed reports on page rendering, resource loading, and optimization [9].

Pingdom is a tool for monitoring the availability and performance of websites. The software system uses global servers for availability monitoring, and APIs for integration with other systems. Pingdom provides tools to monitor page load speed and client-side performance and uses real user data to evaluate user interaction with the web application. Pingdom does not specialize in backend analysis but only provides basic information about server availability and response time [10].

AppDynamics is a commercial application performance monitoring and user experience management platform that uses agents to collect metrics from various system components and analytical tools for data analysis. AppDynamics offers tools for monitoring client-side performance, including real-user metrics and JavaScript analysis. AppDynamics also provides deep back-end monitoring, including application, database, and infrastructure performance analysis. This allows for identifying bottlenecks and optimizing the client part of the software [11].

WebPageTest is an open-source tool for measuring web page performance, providing detailed reports, using browsers for testing and APIs for automation. WebPageTest specializes in analyzing the performance of the client side of a web application, offers detailed reports on page load times, resource optimization, and rendering, and has limited capabilities for analyzing server performance [12].

Sentry is a commercial platform that monitors errors and application performance. The advantage of this platform is integration with various programming languages and frameworks, and analysis of logs and metrics. Sentry provides front-end error and performance monitoring, including JavaScript analysis and real-time error tracking. Sentry also

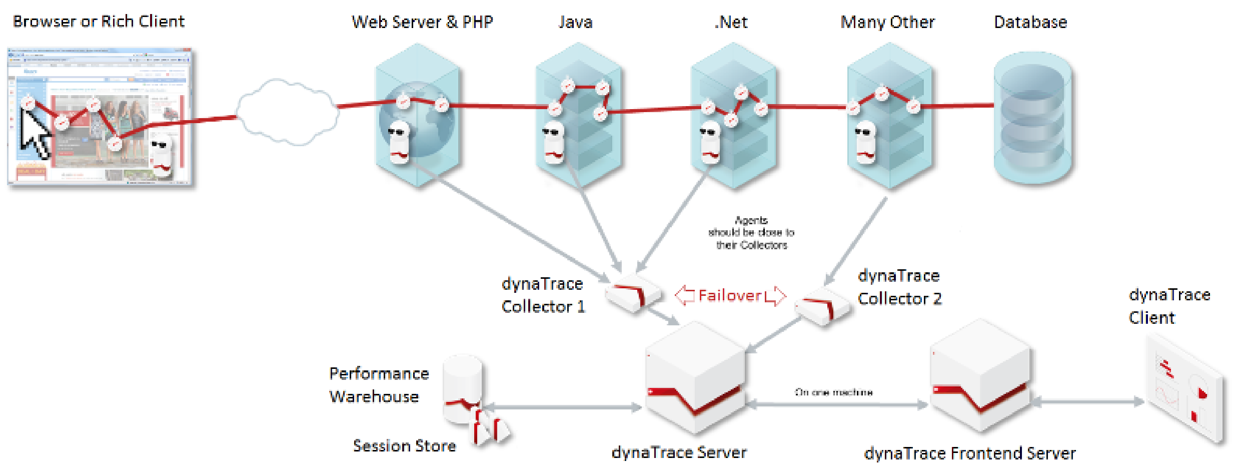


Fig. 2. Dynatrace monitoring process [8]

supports back-end monitoring, including server errors and back-end performance [13].

LoadRunner is a commercial load-testing tool created by Micro Focus that supports various protocols, and load scenarios based on real user actions. LoadRunner is less suitable for client-side performance analysis, and specializes in server load testing, providing scalable tests to determine server performance under heavy load [14].

Table 1 shows the capabilities of software systems to analyze the performance of client and server web applications written in JavaScript.

Google Lighthouse, GTmetrix, WebPageTest, and Pingdom are compatible with major browsers such as Chrome, Firefox, Safari, and Edge. Apache JMeter primarily tests web applications and APIs, not browser functionality.

New Relic and Dynatrace provide cross-browser compatibility for monitoring JavaScript errors and user interactions in Chrome, Firefox, Safari, and Edge. AppDynamics supports monitoring in all major browsers. Sentry integrates with most modern browsers, including Chrome, Firefox, Safari, and Edge, to catch JavaScript errors. LoadRunner focuses on performance testing and is compatible with web applications running in any browser, but does not provide browser-specific integration features.

Google Lighthouse relies on Chrome DevTools, limiting its integration flexibility with non-Chrome test environments. It may struggle with scalability for over 1,000 requests per minute and lacks support for large-scale distributed performance testing, problematic for web applications with more than 100,000 daily requests.

Apache JMeter requires significant configuration, is challenging for beginners, and its interface slows with large test plans. Though it supports distributed testing, managing multiple instances can be cumbersome, and handling over 10,000 concurrent users can bottleneck memory usage.

New Relic's setup is complex, with proprietary limitations and high costs for over 500,000 daily requests, potentially leading to data processing limits. Dynatrace's complex setup and reliance on AI for analysis can obscure problem causes, with scalability issues due to high costs and significant data storage needs. GTmetrix's reliance on third-party tools limits customization, making it less suitable for large-scale testing of web applications with over 1 million daily requests.

Pingdom lacks deep server performance analysis and struggles with over 10,000 requests per second. AppDynamics offers comprehensive monitoring, but its high cost and decreased performance with over 1 TB of data per month limit its appeal to small and medium enterprises.

WebPageTest lacks server-side tools and becomes resource-intensive with large sites having over 1,000 elements per page. Sentry detects errors but can delay with over 50,000 events per minute.

LoadRunner, while powerful, is complex and expensive, and its performance degrades with over 1 million virtual users. As data volumes exceed 10 GB per day, performance bottlenecks require advanced

Table 1

Capabilities of software systems for analyzing the performance of the client and server part of web applications and their main limitations

Software system	Analysis of the client part	Analysis of the server part	Disadvantages
Google Lighthouse	+	-	May require expertise to interpret the results
Apache JMeter	-	+	Difficulty setting up requires knowledge of Java programming language
New Relic	+	+	High cost for large teams (from $99/month for each host)
Dynatrace	+	+	High cost, complexity of configuration (from $69/month for each host)
GTmetrix	+	-	Limitations of the free version (Pro plans from $10/month)
Pingdom	+	-	High cost for advanced features (from $15/month)
AppDynamics	+	+	High cost, complexity of setup (from $330/month)
WebPageTest	+	-	Limited support for other protocols
Sentry	+	+	High cost for large teams (from $29/month), limited functionality
LoadRunner	-	+	High cost, complexity of setting (from $4000/year)

data architectures like NoSQL, in-memory data grids, or cloud storage. Managing distributed components is challenging, necessitating robust orchestration and monitoring tools, and addressing network latency and data consistency issues is critical for maintaining system performance.

Using machine learning methods and AI to evaluate the performance of web applications

Google Lighthouse uses AI to analyze various web performance metrics and suggest specific improvements for optimization. For example, the system allows to identify resources that block rendering and recommend ways to postpone them, thus speeding up page load time. Regression analysis is used to predict the performance of web pages based on historical data.

Google Lighthouse automatically tests web pages and collects various performance metrics such as First Contentful Paint (FCP), Time to Interactive (TTI), Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) and others. For example, over the past month, the Lighthouse system tested the page 100 times and collected data on load time (FCP, TTI), size of downloaded resources, number of requests, etc. After collecting the data, Lighthouse applies regression analysis to identify patterns. For example, an analysis might show that increasing the size of images on a page by 1MB results in an increase in FCP load time of 200ms.

A regression model based on the collected data, where performance metrics are the dependent variables and factors such as image sizes, number of requests, file types, etc. are the independent variables. Based on the model, it is possible to predict the performance of web pages when certain parameters are changed. If we reduce the size of the images by 500KB, the model can show that the FCP load time will decrease by 100ms.

Web developers can use these predictions to optimize their pages. For example, they can decide to optimize images or reduce the number of HTTP requests based on predictions from regression analysis to improve overall page performance. Using regression analysis, Google Lighthouse can find that every additional 100KB of images adds 50ms to FCP's load time. Web developers can predict that reducing the total image size by 500KB can reduce FCP load times by 250ms. This provides concrete numbers to base optimization decisions on.

The use of regression analysis in Google Lighthouse allows not only to evaluate the current performance of web pages but also to predict how certain changes in the code and content will affect these indicators, which makes the optimization process more scientific and justified.

Random Forest in Google Lighthouse is used to classify and evaluate the impact of various factors on page performance. A random forest is an ensemble ML method consisting of many decision trees. Each tree is trained on different subsets of data and subsets of features, which allows the model to be resistant to overtraining and more accurate in general cases. A random forest models the relationship between a dependent variable (performance measures such as First Contentful Paint, Time to Interactive) and independent variables (factors affecting performance such as image size, number of HTTP requests, and script loading time).

Lighthouse collects data about web page performance, including performance metrics and the factors that influence them. For example, data about FCP, image size, number of requests, and loading time of scripts for 1000 web pages is collected. The data are divided into training and test samples. Prepared data can look like a table, where rows represent web pages and columns represent performance metrics and factors.

A random forest is trained on the training sample. For example, 100 decision trees are created, where each tree is trained on a random subset of the data. A random forest model classifies and evaluates the impact of various factors on performance.

For example, the model might show that image size has the greatest impact on FCP, with an importance score of 0.45, number of requests at 0.30, and script load time at 0.25. The model uses a trained random forest to predict the performance of new web pages. For a web page with an image size of 1800 KB, a request count of 55, and a script load time of 1200 ms, the model predicts an FCP of 1600 ms.

Random Forest helps developers understand which factors most affect the performance of their web pages and predict how changes in those factors will affect performance metrics. This allows to make informed decisions to optimize and improve the speed of web applications.

GTmetrix uses ML algorithms to predict how changes to a website might affect its speed and user interaction.

Gradient boosting is used to more accurately determine the factors that affect the performance of web pages. Gradient boosting is an ensemble ML technique that builds a model by successively adding weak models (such as decision trees) in such a way that each successive model tries to correct the errors of the previous one. GTmetrix collects web page performance data, including performance metrics (eg Page Load Time, First Contentful Paint) and factors that affect them (eg image size, number of HTTP requests, script load time).

Gradient boosting in GTmetrix allows to accurately determine which factors have the greatest impact on web page performance. For example, if image size turns out to be the most important factor, developers can focus on optimizing images to improve page load times.

Classification is used to rank pages by performance and identify critical areas. GTmetrix uses various classification methods to analyze and rank web page performance, such as Decision Trees, Random Forest, and Support Vector Machine (SVM). Decision trees are used to identify the most important factors affecting web page performance. This helps to quickly identify bottlenecks and areas that need optimization. A random forest improves the accuracy of predictions by combining the results of many decision trees. This provides more reliable and stable performance analysis results.

WebPageTest uses artificial intelligence (AI) to simulate various network conditions and user interactions, providing a more comprehensive analysis. Neural networks are used for complex analysis of the influence of various parameters on the performance of web pages.

Pingdom uses AI to continuously monitor website uptime and performance. The system can automatically alert users to performance issues and suggest potential fixes based on historical data patterns. If the server is often slow at a certain time, Pingdom can predict this and notify the user in advance. Time series analysis is used to predict downtime and website availability issues. Using ML techniques such as autocorrelation and moving averages, the system can detect patterns and anomalies in time series data. This allows Pingdom to predict potential downtime and availability issues, providing users with warnings and recommendations to avoid them. Logistic regression is used to identify the probability of problems based on historical data. The system analyzes various parameters such as response time, number of requests, throughput, and other performance indicators to estimate the probability of failures or downtime. Logistic regression can be used to create models that predict the risk of problems based on input data.

Apache JMeter mainly focuses on load and performance testing of web applications and APIs. By integrating ML, JMeter can detect anomalies in performance data, such as unexpected spikes in response time, and automatically flag them for further investigation. K-means clustering is used to group similar queries or transactions based on their performance and behavioral characteristics and to identify atypical test results that may indicate performance issues. When testing a web application, we can collect response time data for various requests. Using the K-means algorithm, these queries can be grouped into several clusters, such as fast, medium and slow queries. This allows to identify groups of requests that have similar performance and identify critical areas that need optimization.

New Relic uses AI and ML for predictive analytics and performance monitoring. The system can automatically detect performance bottlenecks and provide recommendations to optimize web application performance. For example, New Relic can analyze transaction traces to pinpoint slow queries and suggest methods for indexing or optimizing queries. Linear regression is used in New Relic to analyze trends in performance data to predict future problems.

Dynatrace uses AI for root cause analysis and predictive analytics. Its Davis AI engine automatically correlates performance data across the entire application stack, identifying the root cause of problems without manual intervention. For example, if a specific microservice is causing slowdowns, Davis AI can highlight it and suggest steps to fix it. Recurrent neural networks (RNNs) are used to analyze time series and predict server loads. Reinforcement learning uses automatic changes in system configurations.

AppDynamics uses AI anomaly detection to monitor application performance, can automatically detect deviations from normal behavior, and perform detailed diagnostics. For example, if a certain transaction starts taking longer than usual, AppDynamics can flag this and offer insight into whether the problem is in the database, server, or network. A decision tree is used to identify critical paths in code that affect performance. Anomaly detection is used to automatically detect deviations in system performance.

Sentry integrates ML to prioritize and cluster bugs, helping developers focus on the most critical issues. For example, Sentry can automatically group similar bugs together and rank them based on their impact, allowing developers to prioritize the most pressing issues. Clustering is used to group similar errors and identify the root causes of problems. For this, clustering methods such as K-means or Density-Based Spatial Clustering of Applications with Noise (DBSCAN) are used. K-means clustering helps identify groups of similar errors based on their characteristics, such as error messages and call stacks. DBSCAN is used to detect high-density clusters in error data, allowing the detection of sporadic or rare errors that may have a significant impact on the system.

Deep learning in Sentry is used to automatically prioritize errors and their criticality. For example, RNNs can analyze a sequence of errors and identify patterns that allow predicting the probability of future failures. Convolutional Neural Networks (CNNs) can be applied to analyze logs and traces, automatically classifying errors according to their criticality. This allows developers quickly determine which errors have the greatest impact on the system.

LoadRunner uses AI to simulate realistic user behavior patterns and predict system performance under various loads. It can model complex user interactions and analyze how they affect system performance, helping to ensure that applications can handle real-world usage scenarios. Regression analysis is used to determine the effect of various factors on performance during stress tests. Anomaly detection is used to detect unexpected behaviors when testing a system under load.

Software system architecture designing for intelligent analysis of web application performance

In the research, we use regression analysis to predict the impact of changes on page performance. Regression analysis is used to predict the performance of web pages based on several independent variables.

The main independent variables include: **time to the first byte (Time to First Byte, TTFB)**, which is measured in milliseconds (ms) and indicates the time required by the server to respond; **the loading time of the first content image (First Contentful Paint, FCP)**, also measured in milliseconds (ms) and indicates when the first image or text appears on the screen; **speed index (Speed Index)**, which is measured in milliseconds (ms) and shows the speed of page content display; **the number of HTTP requests (Number of Requests)**, measured by the number of requests and indicating the total number of requests required to download all page resources; and **the total size of downloaded resources (Total Page Weight)**, measured in kilobytes (KB) or megabytes (MB) and indicate the total size of all downloaded resources. These variables are used to build a regression model that helps determine how each of these factors affects overall page performance, allowing for more accurate predictions and identifying underlying issues.

CNNs we use to analyze images and videos, which can be useful for evaluating visual aspects of web page performance, such as image and video load times. The CNN neural network allows to analyze screenshots of web pages at different stages of loading to discover which elements appear earlier or later and how this affects the user's perception.

LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit) are used to analyze data sequences. They are used for modeling user interaction, which is a sequence of actions. RNNs can model user click sequences and their site navigation, taking into account previous actions to predict next steps and assess how interface changes may affect overall performance.

Software system collects a large amount of data about user interaction with web pages, including loading times of various resources, clicks, page scrolling, etc. The data is cleaned and converted into a format suitable for training neural networks. For example, images can be normalized, and click sequences can be encoded.

CNNs are trained on images or videos to recognize patterns related to page load times and rendering. RNNs learn from sequences of clicks and other user interactions to understand behavioral patterns and predict future actions.

After training, models analyze new data about user interactions and network conditions. Simulating different scenarios (such as low network bandwidth) allows to evaluate how these conditions affect performance. Analysis results are used to generate performance optimization recommendations. If CNN finds that large images have a strong impact on load times, it can suggest optimizing the images or using responsive formats. According to the research, using neural networks, the performance of web applications can increase on average by 15-20%. Neural networks allow more accurate analysis of the impact of various parameters on the performance of web pages by modeling complex user interactions and network conditions.

Designing a software system architecture for evaluating the performance of web applications using regression analysis uses the integration of several components to provide data collection, processing, and analysis. The data obtained from such a model will be used in the sub calculations of the assessment of the impact of various factors on the system. Below is a list of such factors:

– server factors: include data on processor load time and the amount of RAM used;

– factors of the web application client: response time, request frequency, error frequency, session duration;

– data source factors: database query time, number of active connections;

– infrastructure factors: server configuration, percentage of cached data, load distribution system performance;

– user factors: location, browser, device type;

– time factors: time of day, day of the week.

A classification method should be used in the architecture of this system, this will help to divide the data into different categories or classes based on a set of input variables. Such separation will make it possible to provide system users with better recommendations for improving the performance of the web application. Decision trees are used to hierarchically divide data into subsets based on the values of the input variables. They work by building a hierarchical structure where each node represents a condition on one of the input variables and each branch is the result of that condition, eventually leading to leaves that contain the final solution or class.

In the context of web applications, decision trees can be used to classify requests into different performance categories, such as fast, medium, or slow responses. With the help of the classification of requests, we can determine which of them load the system the most, and take measures to optimize their processing. This may include changes to the web application architecture, code improvements, or server configuration (Fig. 3).

Research results. Classification methods provides more accurate predictions about web page performance and improve user experience, to 10-15% faster page load times and a better user experience, which in turn can lower bounce rates and increase web applications user satisfaction.

Clustering helps identify patterns in query behavior that can point to specific conditions or configurations that are causing performance degradation. This, in turn, allows for more targeted and detailed optimization, which can reduce response time by 15-20% for the slowest requests.

The use of time series analysis and logistic regression allows for significant improvement in the management of website performance and availability. This provides in some cases to 20-30% reduction in downtime, which in turn increases the reliability and stability of websites, improving the user experience and reducing the financial losses associated with site unavailability.

With the help of linear regression, we can analyze the response time of requests and determine the growth trends of this indicator. If the regression model shows a steady increase in response time, this may signal potential performance issues that may arise in the future. Identifying such trends allows to take preventive measures, such as optimizing the code or increasing server resources, to avoid a decrease in the performance of the web application. Such forecasting helps reduce downtime by 10-15% and ensures more stable system operation.

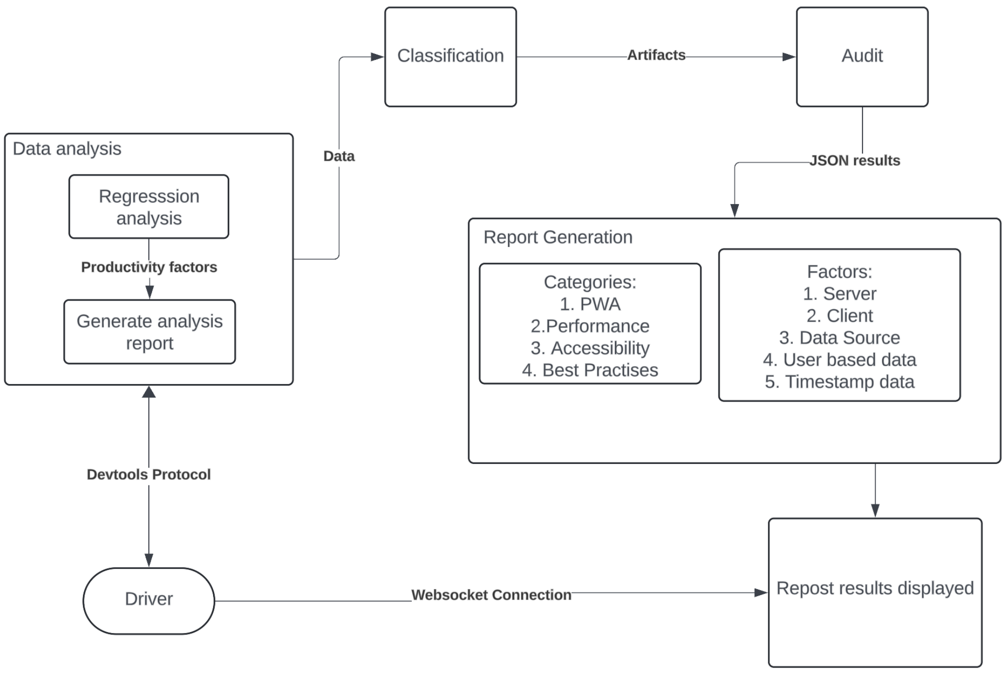


Fig. 3. The proposed architecture of the software system for intelligent analysis of the performance of web applications

Deep Neural Networks (DNNs) are used for complex analysis and predictions, including user behavior and system load. DNNs can analyze historical data about server loads during peak periods and predict when similar peak loads can be expected next time. This allows a configuration of additional resources in advance or optimize existing ones to cope with the expected increase in load. As a result of such measures, system performance is improved during peak loads, reducing the risk of downtime and ensuring continuity of service to users. This approach can increase the overall efficiency of the system in some cases by 20-25%, which significantly improves the user experience.

Conclusions and future work. The article discusses the analysis of available software solutions for evaluating the performance of web applications. Not all considered software systems are equally well suited for analyzing the performance of web applications written in JavaScript. Some have strengths on the client side, others on the server side, and some offer comprehensive analysis.

Google Lighthouse, GTmetrix, Pingdom, and WebPageTest are good for detailed client-side performance analysis but have limited capabilities for back-end analysis. Apache JMeter is a powerful tool for server load testing, but less suitable for client-side analysis. New Relic, Dynatrace, AppDynamics, and Sentry provide end-to-end monitoring, covering both the client and backend, offering deep analysis of application, database, and infrastructure performance. LoadRunner specializes in scalable server load testing and has limited capabilities for client-side analysis.

The architecture of a software system for intelligent analyzing the performance of web applications written in JavaScript is proposed, which allows to collection of numerical data on factors affecting the performance of a web application using regression analysis, perform regression analysis to determine the assessment of the influence of factors, clustering and classification of processed data.

The proposed software system provides recommendations for web developers, which must be implemented to improve the performance of the web application and reduce the load on the web server.

According to the conducted research, the use of regression analysis and classification methods allows for an increase in the productivity of web applications by an average of 20%.

Bibliography:

1. Shailesh S., Suresh P.V. A Survey and Analysis of Techniques and Tools for Web Performance Optimization. *Journal of Information Organization*. 2018. Vol. 8. № 2. P. 31–57. DOI: 10.6025/jio/2018/8/2/31-57.
2. Leone J. and Traini L. Enhancing Trace Visualizations for Microservices Performance Analysis. *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*. 2023. P. 283–287. https://doi.org/10.1145/3578245.3584729.
3. Willnecker Felix, Andreas Brunnert, Wolfgang Gottesheim and Helmut Krcmar. Using Dynatrace Monitoring Data for Generating Performance Models of Java EE Applications. *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. 2015. P. 103–104. DOI:10.1145/2668930.2688061.
4. Oleshchenko L., Burchak P. Web Application State Management Performance Optimization Methods. *Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. Lecture Notes on Data Engineering and Communications Technologies*. 2023. vol. 181. P. 59–74. Springer, Cham. https://doi.org/10.1007/978-3-031-36118-0_6.
5. Google Lighthouse. https://developer.chrome.com/docs/lighthouse/overview
6. Apache JMeter. https://jmeter.apache.org/
7. New Relic. https://newrelic.com/
8. Dynatrace. https://www.dynatrace.com/
9. GTmetrix. https://gtmetrix.com/
10. Pingdom. https://www.pingdom.com/
11. AppDynamics. https://www.appdynamics.com/
12. WebPageTest. https://www.webpagetest.org/
13. Sentry. https://sentry.io/welcome/
14. LoadRunner. https://www.opentext.com/en-gb/products/loadrunner-professional

Олещенко Л.М., Бурчак П.В. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ПОКАЗНИКІВ ПРОДУКТИВНОСТІ ВЕБЗАСТОСУНКІВ

Сьогодні зростає кількість вебзастосунків, які обробляють великі обсяги даних, що створює нові виклики для розробників і користувачів. Вебзастосунки, які працюють з великими даними, стають важливою частиною бізнесу, державного управління та повсякденного життя, сприяючи підвищенню *ефективності, точності та швидкості прийняття рішень. Однак з цим пов'язані численні проблеми, такі як забезпечення безпеки та конфіденційності даних, ефективна обробка та зберігання великих обсягів інформації, а також необхідність постійного моніторингу і оптимізації продуктивності вебзастосунків. Управління ресурсами та енергоефективність стають особливо актуальними в умовах економії енергоресурсів. Вирішення цих проблем потребує використання новітніх технологій, таких як інтелектуальні системи моніторингу та аналізу, що допомагають забезпечити стабільну та ефективну роботу вебзастосунків в умовах постійного зростання обсягів даних і складності задач.*

Вирішення проблем продуктивності вебзастосунків за допомогою інтелектуальних систем моніторингу дозволить розробникам швидко і точно виявляти вузькі місця та оптимізувати код, що, в свою чергу, забезпечить кращий користувацький досвід і зменшення витрат на обслуговування. Наразі існують невирішені питання, такі як складність інтеграції нових технологій у вже існуючі системи, необхідність врахування різноманітних факторів впливу на продуктивність, а також забезпечення високого рівня безпеки і конфіденційності даних під час моніторингу.

У статті проаналізовано наявні програмні рішення Google Lighthouse, Apache JMeter, New Relic, Dynatrace, GTmetrix, Pingdom, AppDynamics, WebPage Test, Sentry та LoadRunner, їх функціональні можливості, основні переваги та недоліки. Проаналізовано можливості використання технологій машинного навчання та штучного інтелекту в розглянутих програмних системах. На основі проведеного аналізу запропоновано архітектуру програмної системи для інтелектуального аналізу продуктивності вебзастосунків, написаних мовою JavaScript, яка дозволяє виконувати збір числових даних про фактори, що впливають на продуктивність вебзастосунку, виконувати регресійний аналіз для визначення оцінки впливу факторів, кластеризацію та класифікацію оброблених даних для коректності виділення рекомендацій для розробників, які необхідно вжити задля покращення продуктивності вебзастосунку та зменшення навантаження на вебсервер. Згідно проведених досліджень, використання методів машинного навчання в програмних системах аналітики показників вебзастосунків дозволяє збільшити продуктивність вебзастосунків в середньому на 20 %.

Ключові слова: *архітектура програмної системи, оцінка продуктивності вебзастосунків, JavaScript, машинне навчання, AI, регресійний аналіз, кластеризація, класифікація, нейронні мережі.*