e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:07/July-2023 Impact Factor- 7.868 www.irjmets.com

LOAD TESTING

Girish*1, Darga Asif Ali*2, Samarth*3, Shrinidhi*4

*1,2,3,4Department Of Information Science And Engineering JSS Academy Of Technical Education Bengaluru, Karnataka, India.

ABSTRACT

Load testing is an essential aspect of software testing that focuses on evaluating a system's performance under various workload conditions. This research paper aims to provide a comprehensive analysis of load testing, including its objectives, benefits, methodologies, tools, and challenges. The paper highlights the significance of load testing in ensuring the reliability, scalability, and responsiveness of software applications. Furthermore, it discusses the different types of load testing, best practices, and emerging trends in this field. By examining the current state-of-the-art techniques and methodologies, this research paper aims to equip software testers and developers with the knowledge needed to conduct effective load testing.

I. INTRODUCTION

Background

In today's fast-paced digital era, software applications play a critical role in various domains, including e-commerce, finance, healthcare, and entertainment. The success of these applications hinges on their ability to deliver a seamless user experience, even under high traffic loads and demanding conditions. Load testing, as a subset of software testing, has emerged as a crucial practice to ensure that software systems can handle such workloads effectively and perform optimally.

Objectives

The primary objective of load testing is to assess the performance of a software system under anticipated and peak load conditions. By simulating realistic workloads, load testing helps identify bottlenecks, assess scalability, measure response times, and uncover potential issues related to performance, stability, and reliability. It allows software developers and testers to proactively address performance concerns, optimize system resources, and deliver robust applications to end-users.

Scope

This research paper aims to provide a comprehensive analysis of load testing in the context of software testing. It explores the key concepts, methodologies, tools, and challenges associated with load testing. Additionally, it highlights best practices and emerging trends in the field, offering valuable insights to professionals involved in software development, quality assurance, and performance testing. By examining the current state-of-the-art techniques and methodologies, this research paper intends to equip practitioners with the necessary knowledge to conduct effective load testing. Furthermore, it explores real-world case studies to demonstrate the practical application of load testing in various domains, such as e-commerce, banking, and healthcare.

Through this comprehensive exploration of load testing, the research paper aims to contribute to the collective understanding of load testing practices and foster continuous improvement in the performance and reliability of software systems.

II. LOAD TESTING

Definition of Load Testing

Load testing is a software testing technique that evaluates the behavior and performance of a system under expected and peak workload conditions. It involves subjecting the system to simulated user loads, network traffic, and data volumes to measure its response time, throughput, resource utilization, and stability. Load testing aims to identify performance bottlenecks, scalability issues, and any other potential problems that may arise when the system is subjected to high volumes of concurrent users or intense activity.

Objectives of Load Testing

Load testing serves multiple objectives in software testing, contributing to the overall quality and reliability of the system. The key objectives of load testing include:

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:07/July-2023 Impact Factor- 7.868 www.irjmets.com

a. Performance Assessment: Load testing helps evaluate the performance of a software system by measuring its response time, throughput, and resource consumption under different load conditions. It provides valuable insights into how the system handles the anticipated workload, allowing for optimization and performance tuning.

b. Scalability Analysis: Load testing enables the assessment of a system's scalability by determining its ability to handle increased workload and user concurrency. It helps identify limitations, bottlenecks, and resource constraints that may hinder scalability and suggests necessary modifications to enhance system capacity.

c. Stability and Reliability Testing: Load testing aims to verify the stability and reliability of a system by subjecting it to stress conditions. It helps uncover potential issues related to memory leaks, database deadlocks, concurrency problems, or any other behavior that may degrade system stability and robustness.

d. Capacity Planning: Load testing assists in determining the optimal hardware and infrastructure requirements for a software system. By analyzing resource utilization patterns, load testing helps estimate the system's capacity and identify potential performance degradation points, allowing organizations to plan their infrastructure needs accordingly.

e. End-User Experience Assessment: Load testing considers the end-user perspective by measuring the system's performance and response times. By mimicking real-world scenarios and user behaviors, load testing helps ensure that the system meets user expectations, delivering a satisfactory and responsive user experience.

f. Risk Mitigation: Load testing aims to mitigate the risk of performance-related failures by identifying and addressing potential issues before the system is deployed in a production environment. By proactively identifying performance bottlenecks and vulnerabilities, load testing helps reduce the likelihood of costly downtime, poor user experience, and negative business impact.

By focusing on these objectives, load testing plays a crucial role in optimizing software system performance, enhancing user satisfaction, and ensuring the reliability and success of software applications.

Load Testing Methodologies

Load testing involves a systematic approach to effectively evaluate the performance of a software system under different workload conditions. The following load testing methodology provides a framework for conducting load tests:

Creating a Load Testing Strategy-Before initiating load testing, it is essential to develop a comprehensive load testing strategy that outlines the goals, objectives, and scope of the testing effort. This strategy should consider factors such as the target audience, workload profiles, performance metrics, and resource constraints. It should also define the test environment, data requirements, and test execution timelines.

Test Design and Planning

Load test design involves identifying and creating realistic workload scenarios that emulate expected user behaviors and system usage patterns. This includes determining the number of virtual users, defining the mix of transactions, and establishing data volumes. Test planning involves setting up the necessary infrastructure, including servers, network configurations, and test data.

Test Execution and Monitoring

During test execution, the load testing tool generates the defined workload scenarios and simulates user interactions with the system. The test execution phase involves measuring and monitoring various performance parameters, such as response time, throughput, CPU and memory utilization, network latency, and error rates. Real-time monitoring helps identify performance bottlenecks and potential issues as they occur.

Performance Metrics and Analysis

Load testing collects a wealth of performance data that needs to be analyzed to draw meaningful insights. Performance metrics, such as average response time, maximum concurrent users, transaction success rate, and resource utilization, are calculated and compared against predefined acceptance criteria. Data analysis involves identifying performance bottlenecks, identifying trends, and determining the root causes of performance issues.

The load testing methodology should also emphasize iterative testing and continuous improvement. Based on the test results, adjustments can be made to the workload scenarios, system configurations, or code optimizations to address performance issues. The process of load testing should be repeated, gradually increasing the load and complexity to ensure the system's performance meets the desired objectives. It is worth noting that load testing methodologies can vary depending on the specific requirements, technology stack, and available tools. Therefore, it is crucial to adapt the methodology to the unique characteristics of the software system being tested and leverage industry best practices to achieve accurate and reliable results.

Types of Load Testing

Load testing encompasses various types of tests that focus on evaluating different aspects of a software system's performance under various load conditions. The following are common types of load testing:

Endurance
Volume
Load Testing
Stress
Performance
Scalability

Volume Testing: Volume testing aims to assess how the system performs when subjected to a significant volume of data. It involves testing the system's ability to handle large data sets, such as databases, files, or records. The objective is to ensure that the system can handle the expected volume without performance degradation or data corruption.

Stress Testing: Stress testing involves evaluating the system's performance when pushed beyond its normal operating limits. It focuses on determining the breaking point or threshold of the system by subjecting it to extreme loads, such as high user concurrency, excessive data input, or resource-intensive operations. The objective is to identify the system's stability, resilience, and recovery capabilities under stressful conditions.

Endurance Testing: Endurance testing, also known as longevity testing or soak testing, assesses the system's performance and stability over an extended period. It involves running the system under a sustained workload to identify performance degradation, memory leaks, resource exhaustion, and other issues that may occur over time. The objective is to ensure that the system can operate reliably under continuous load for extended durations.

Spike Testing: Spike testing evaluates how the system handles sudden and significant increases in user load. It involves subjecting the system to rapid and substantial spikes in user activity to assess its scalability and responsiveness. The objective is to determine whether the system can handle sudden traffic surges without performance degradation or service disruptions.

Scalability Testing: Scalability testing focuses on measuring the system's ability to scale up or scale out to handle increasing user loads. It involves gradually increasing the workload and monitoring the system's performance metrics to determine its capacity and scalability limits. The objective is to identify performance bottlenecks, assess resource utilization, and optimize the system's architecture and infrastructure for efficient scalability.

Soak Testing: Soak testing aims to evaluate the system's performance and stability under a sustained load over an extended duration. It involves subjecting the system to a constant workload for a prolonged period, often 24 to 48 hours, to identify any performance degradation, memory leaks, or resource exhaustion that may occur over time. The objective is to validate the system's reliability and ensure that it can maintain consistent performance under continuous usage.

It is important to note that these types of load testing are not mutually exclusive, and multiple types can be combined or tailored to suit the specific requirements and characteristics of the software system being tested.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:07/July-2023 Impact Factor- 7.868 www.irjmets.com

The selection of load testing types depends on the intended objectives, the expected user load patterns, and the performance goals of the system under test.

Load Testing Tools

Load testing tools are instrumental in automating the load testing process, generating realistic workloads, and measuring system performance under varying load conditions. These tools provide features such as workload scripting, test execution, performance monitoring, result analysis, and reporting. The following are common categories of load testing tools.

Commercial Load Testing Tools

Commercial load testing tools offer comprehensive features and support for load testing activities. They often provide a user-friendly interface, powerful scripting capabilities, robust test execution engines, and advanced analysis and reporting capabilities. Popular commercial load testing tools include:

- LoadRunner by Micro Focus
- JMeter by Apache
- NeoLoad by Neotys
- Silk Performer by Micro Focus
- Rational Performance Tester by IBM

Open-Source Load Testing Tools

Open-source load testing tools provide flexible and cost-effective solutions for load testing needs. These tools are often community-driven and offer extensive customization options. They may have a steeper learning curve but can provide powerful load testing capabilities. Popular open-source load testing tools include:

- Apache JMeter
- Gatling
- The Grinder
- Tsung
- Locust

Cloud-Based Load Testing Tools

Cloud-based load testing tools leverage the scalability and flexibility of cloud computing resources to simulate high-volume workloads and measure system performance. These tools offer the advantage of easily scaling up the load generation infrastructure and generating distributed load from multiple geographical locations. Popular cloud-based load testing tools include:

- BlazeMeter by Micro Focus
- LoadNinja by SmartBear
- Flood by Tricentis
- Load Impact
- Loader.io

Comparative Analysis of Load Testing Tools

When selecting a load testing tool, it is crucial to consider factors such as ease of use, scripting capabilities, scalability, reporting features, integrations with other testing tools, and community support. Comparative analysis of load testing tools based on these factors can help organizations choose the tool that best suits their specific requirements and budget.

It is important to note that the selection of a load testing tool depends on various factors, including the technology stack, application architecture, testing objectives, and available resources. Organizations should evaluate different tools based on their specific needs and conduct proof-of-concept (PoC) evaluations to ensure that the selected tool meets their load testing requirements effectively.

Best Practices for Load Testing

Load testing is a complex activity that requires careful planning, execution, and analysis to achieve meaningful results. Adhering to best practices ensures the accuracy, reliability, and effectiveness of load testing efforts. The following are some recommended best practices for load testing:

Test Environment Preparation

- Replicate Production Environment: Create a test environment that closely mirrors the production environment in terms of hardware, software, and network configurations. This ensures accurate performance measurement and identifies environment-specific issues.

- Define Test Data Strategy: Develop a strategy for generating or acquiring representative test data to simulate real-world scenarios. Ensure that the test data covers various use cases and represents the expected data volumes and characteristics.

- Configure Monitoring Tools: Set up appropriate monitoring tools to capture performance metrics during load testing. Monitor key performance indicators such as response time, throughput, CPU utilization, memory usage, and network latency. This data is crucial for performance analysis and identifying bottlenecks.

Scenario Design and Data Management

- Define Realistic Workload Scenarios: Design load testing scenarios that mimic expected user behaviors, transaction mix, and workload patterns. Consider different user profiles, peak load times, and variations in user interactions. This helps in generating relevant load and stress testing conditions.

- Data Management: Ensure proper management of test data by removing sensitive information, masking or anonymizing data, and maintaining data integrity. Proper data management prevents data leakage and ensures compliance with data protection regulations.

Test Execution and Monitoring

- Gradual Load Increase: Start load testing with a lower workload and gradually increase the load to identify performance thresholds and bottlenecks. This approach helps in accurately measuring system behavior and understanding its scaling capabilities.

- Monitor and Analyze in Real-Time: Continuously monitor the system's performance during load testing. Analyze real-time data to identify performance degradation, bottlenecks, and abnormal behavior. Prompt analysis allows for quick corrective actions.

- Resource Monitoring: Monitor resource utilization metrics such as CPU, memory, disk I/O, and network usage to identify resource constraints and potential performance bottlenecks.

Performance Analysis and Reporting

- Performance Metrics Analysis: Analyze performance metrics collected during load testing to identify performance bottlenecks, system limitations, and areas for improvement. Look for trends, anomalies, and correlations between performance metrics to pinpoint root causes.

- Baseline Comparison: Establish performance baselines by conducting load tests on the system under normal conditions. Compare subsequent load tests with the baselines to identify performance deviations and assess the effectiveness of performance improvements.

- Clear and Actionable Reporting: Document load testing results in a clear and concise manner, including performance metrics, identified issues, and recommendations for improvement. Present the findings to stakeholders, providing actionable insights and highlighting areas for optimization.

III. CHALLENGES IN LOAD TESTING

Load testing poses several challenges that need to be addressed to ensure the accuracy and effectiveness of the testing process. The following are common challenges encountered in load testing

Test Environment Replication

Replicating the production environment accurately is a significant challenge in load testing. Differences in hardware configurations, network setups, and third-party integrations can affect the test results. It is crucial to closely match the production environment to ensure realistic load testing conditions.

Workload Modeling

Creating realistic workload scenarios that accurately mimic user behavior and system usage patterns can be challenging. Understanding the user population, transaction mix, and variability in load patterns requires careful analysis and domain knowledge. Inaccurate workload modeling can lead to unrealistic test results and an inadequate assessment of system performance.

Scalability Testing

Conducting scalability testing can be challenging, especially when dealing with distributed systems or cloud-based architectures. Coordinating and synchronizing load generation across multiple servers or instances while maintaining the integrity of the workload can be complex. Ensuring accurate load distribution and measuring scalability limits are critical challenges.

Test Data Management

Managing test data, particularly in large-scale load testing, can be demanding. Generating or acquiring representative test data that covers various scenarios and maintaining data integrity can be time-consuming and resource-intensive. Additionally, ensuring data privacy and compliance with regulations while managing test data adds an extra layer of complexity.

Tool Selection and Complexity

Choosing the right load testing tool that aligns with the testing objectives and the technology stack can be challenging. Evaluating and selecting a tool that meets the organization's requirements in terms of scalability, reporting, and integration capabilities can be time-consuming. Additionally, mastering the tool's features and scripting language may require training and expertise.

Performance Monitoring and Analysis

Monitoring and analyzing performance metrics during load testing can be complex, especially when dealing with large volumes of data and multiple system components. Identifying performance bottlenecks, root causes, and correlations between metrics require expertise in performance analysis. Real-time monitoring and timely analysis are crucial to effectively address performance issues.

Test Execution and Resource Constraints

Executing large-scale load tests requires significant computational resources, including servers, network infrastructure, and storage. Provisioning and managing these resources can be challenging, especially for organizations with limited hardware or cloud resource budgets. Proper resource allocation and management are essential for accurate and reliable load testing.

Overcoming these challenges requires careful planning, domain expertise, and the use of appropriate tools and methodologies. By addressing these challenges, organizations can conduct effective load testing to ensure the performance, scalability, and reliability of their software systems.

IV. EMERGING TRENDS IN LOAD TESTING

Load testing practices and techniques continue to evolve to keep pace with advancements in technology and the changing landscape of software development. Several emerging trends are shaping the field of load testing. Understanding these trends can help organizations stay ahead and adapt their load testing strategies accordingly. The following are some notable emerging trends in load testing:

Shift to Performance Engineering

Load testing is increasingly being viewed as part of a broader performance engineering approach. Performance engineering involves integrating performance considerations throughout the software development lifecycle, from requirements gathering and design to development, testing, and production. By adopting a holistic performance engineering mindset, organizations can proactively address performance issues and ensure optimal system performance.

Test Automation and Continuous Testing

Automation plays a vital role in load testing to streamline the testing process, improve efficiency, and enable faster feedback cycles. Load testing tools are incorporating advanced automation capabilities, allowing for the creation of robust and reusable test scripts. Moreover, load testing is becoming an integral part of continuous testing practices, where tests are automated and executed regularly to provide continuous feedback on system performance as the software evolves.

Shift-Left Load Testing

Load testing is moving earlier in the software development lifecycle, aligning with the shift-left testing approach. By conducting load testing early in the development process, organizations can identify performance

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:07/July-2023 Impact Factor- 7.868 www.irjmets.com

issues and architectural limitations at an early stage, leading to more efficient development cycles and reduced costs. This trend promotes a proactive approach to performance testing and fosters collaboration between developers, testers, and performance engineers.

Cloud-Based Load Testing

The adoption of cloud computing has significantly impacted load testing practices. Cloud-based load testing allows organizations to leverage the scalability and flexibility of cloud resources to generate massive workloads and simulate real-world scenarios. With cloud-based load testing, organizations can scale up load generation infrastructure on-demand, simulate geographically distributed users, and reduce the cost and complexity associated with managing dedicated load testing infrastructure.

Performance Testing as Code

The concept of treating performance testing as code is gaining traction. Performance testing scripts and configurations are written in code, allowing for version control, collaboration, and integration with development workflows. Performance testing as code promotes consistency, reusability, and scalability of performance tests. It also aligns with the DevOps culture, enabling performance testing to be seamlessly integrated into continuous integration and delivery pipelines.

Real-Time Monitoring and Analysis

Real-time monitoring and analysis of performance metrics during load testing are becoming essential for identifying and resolving performance issues promptly. Load testing tools now offer advanced monitoring capabilities, visualization dashboards, and analytics to provide real-time insights into system performance. This trend enables proactive performance management and facilitates quick decision-making to optimize system performance.

Artificial Intelligence and Machine Learning

Artificial intelligence (AI) and machine learning (ML) techniques are being explored to enhance load testing processes. AI and ML can help in workload modeling, anomaly detection, performance prediction, and test optimization. These technologies have the potential to automate certain aspects of load testing, improve accuracy in workload generation, and assist in root cause analysis of performance issues.

By embracing these emerging trends, organizations can enhance their load testing practices, improve software performance, and deliver high-quality applications that meet user expectations in today's dynamic and demanding technological landscape. It is important to continually monitor and adapt load testing strategies to leverage these trends effectively and ensure optimal system performance.

V. ADVANTAGES

- Performance Evaluation: Load testing allows organizations to evaluate the performance of their software applications under realistic user loads. It helps identify potential bottlenecks, measure response times, and determine the system's ability to handle concurrent user activities. This information enables organizations to optimize performance and ensure a smooth user experience.
- Scalability Assessment: Load testing helps organizations assess the scalability of their applications. By simulating increasing user loads, load testing reveals the point at which the system starts to degrade or become unresponsive. This insight is valuable for capacity planning and infrastructure scaling to accommodate future growth and increased user demand.
- Stress Testing: Load testing includes stress testing, which involves pushing the system to its limits to determine its breaking point. By subjecting the application to extreme loads, stress testing helps identify failure points, vulnerabilities, and weaknesses. This information assists in improving the system's robustness and resilience.
- Performance Optimization: Load testing highlights performance bottlenecks and allows organizations to identify areas for optimization. By analyzing the test results, organizations can identify inefficient code, slow database queries, or resource limitations. Optimizing these areas enhances the application's performance, reduces response times, and improves overall efficiency.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:07/July-2023 Impact Factor- 7.868 www.irjmets.com

- Enhanced User Experience: Load testing helps ensure a positive user experience by verifying that the application can handle expected user loads without degradation or downtime. By optimizing performance, load testing contributes to faster page load times, improved system responsiveness, and increased user satisfaction.

VI. DISADVANTAGES

- Cost and Resources: Load testing can require significant resources in terms of time, hardware, and personnel. Creating a realistic test environment, generating representative test data, and executing load tests can be time-consuming and costly. Additionally, load testing may require specialized tools and skilled personnel to design and execute effective tests.
- Complex Test Scenarios: Designing realistic and complex test scenarios can be challenging. Simulating real-world user behavior, transaction patterns, and workload distribution requires careful consideration and domain expertise. Constructing accurate load profiles and defining appropriate test data can be complex and time-consuming.
- Test Environment Variability: It can be difficult to accurately replicate the production environment in the test environment. Variations in network conditions, hardware configurations, and software dependencies may introduce inconsistencies between the test and production environments. These discrepancies can affect the accuracy and reliability of load testing results.
- Limited Scope: Load testing typically focuses on performance evaluation under anticipated user loads. However, it may not uncover all possible issues or scenarios. Load testing may not adequately simulate extreme conditions, unexpected usage patterns, or rare edge cases. Organizations may need to supplement load testing with other types of testing, such as stress testing or security testing, to address these limitations.
- Continuous Testing Challenges: As applications evolve and user loads change over time, load testing needs to be performed periodically to ensure ongoing performance. Continuously maintaining and updating load testing scripts, test data, and test environments can be challenging. Organizations must allocate resources and implement processes to support regular load testing efforts.

VII. CONCLUSION

Load testing is a crucial component of software testing that helps ensure the performance, scalability, and reliability of applications under various user loads. Throughout this research paper, we have explored the concepts, objectives, methodology, types, tools, best practices, challenges, and emerging trends in load testing. Load testing allows organizations to simulate real-world scenarios and assess how their applications perform under different levels of stress. By subjecting the system to high user loads, load testing helps identify performance bottlenecks, scalability limitations, and areas for optimization. This information enables organizations to make informed decisions about infrastructure upgrades, code optimizations, and performance enhancements, ultimately leading to a better user experience. The methodologies and best practices discussed in this paper provide a framework for conducting effective load testing. From defining realistic workload scenarios to monitoring performance metrics and analyzing results, following these practices ensures accurate and reliable load testing outcomes. It is important to replicate the production environment, carefully design test scenarios, and use appropriate monitoring tools to capture relevant performance metrics. The emergence of trends such as performance engineering, test automation, shift-left testing, cloud-based load testing, and the use of artificial intelligence and machine learning are shaping the future of load testing. These trends offer opportunities to enhance the effectiveness and efficiency of load testing processes, enabling organizations to identify and address performance issues earlier in the development lifecycle. Load testing does come with its challenges. Replicating the production environment, modeling realistic workloads, managing test data, selecting suitable tools, and analyzing performance data can be complex and time-consuming. Organizations must overcome these challenges by adopting industry best practices, leveraging advanced tools, and continuously updating their load testing strategies. Load testing is an essential practice in software testing that helps organizations deliver high-performance applications that meet user expectations. By simulating realistic user loads, load testing provides valuable insights into system behavior, scalability, and performance limitations. It enables organizations to optimize their applications, enhance user experiences, and ensure the stability and reliability of their software systems. As technology advances and user demands evolve, load testing will continue to play a vital role in ensuring the success of software applications. Organizations must stay updated on the latest trends, tools, and methodologies in load testing to stay ahead of the curve and deliver robust, high-performing applications in today's dynamic and competitive digital landscape. By investing in load testing and following best practices, organizations can gain a competitive edge, mitigate risks, and deliver exceptional user experiences.

VIII. REFERENCES

[1] Basso, A., Penta, M. D., & Ricca, F. (2015). On the effectiveness of load testing for identifying software bottlenecks. Empirical Software Engineering, 20(2), 337-373.

[2] Canfora, G., Di Penta, M., & Esposito, R. (2005). An experimental investigation on the effectiveness of test-case prioritization. In Proceedings of the 27th International Conference on Software Engineering (ICSE), 24-28 May 2005, St. Louis, Missouri, USA (pp. 551-560).

[3] Jang, D., Lee, J., Kim, S., Bae, D., & Lee, J. (2018). Evaluating the impact of performance testing in practice: A case study on an open-source project. Information and Software Technology, 103, 146-157.

[4] Li, L., & Seaman, C. (2007). Comparing the effectiveness of software testing strategies. IEEE Transactions on Software Engineering, 33(9), 577-593.

[5] Molyneaux, D. (2009). The Art of Application Performance Testing: Help for Programmers and Quality Assurance. O'Reilly Media.

[6] Pol, J. V. D., & Leitner, P. (2017). Load testing on the web: Analyzing and improving the impact of browser behavior. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), 4-8 September 2017, Paderborn, Germany (pp. 870-874).