IJSRA

International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(Review Article)

Check for updates

Advanced methodologies in performance testing and engineering

Vasudevan Senathi Ramdoss *

Sr Performance Engineer in Financial Investment Sector, McKinney, Texas, USA.

International Journal of Science and Research Archive, 2024, 11(01), 2660-2667

Publication history: Received on 01 December 2023; revised on 20 February 2024; accepted on 24 February 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.11.1.0038

Abstract

Performance testing & engineering is an important for all types of applications and systems, especially for life critical applications in Healthcare, ecommerce, Automotive, Financial & all critical products/projects etc. This paper presents performance testing & engineering concepts, methodologies and commonly used tools for a variety of existing and developing applications.

Keywords: Performance testing; Performance Engineering; Application performance; Cloud performance testing

1. Introduction

Building a successful product depends on two fundamental constituents: functional (functionality) and non-functional (performance). 'Functionality' refers to what the application lets its users accomplish, including the transactions it enables and the information it renders accessible. 'Performance' refers to the system's ability to complete transactions and furnish information rapidly and accurately despite high multi-user interaction or reserved hardware resources. Application failure due to performance-related problems is preventable with pre-deployment performance testing [1]. However, many teams struggle due to a lack of professional performance testing methods, which often results in issues with availability, reliability, and scalability during deployment [1].

2. Performance Testing Concepts

Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload [2]. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and is also known as "Load Testing".

The focus of Performance Testing, Speed – Determines whether the application responds quickly, Scalability – Determines the maximum user load the software application can handle & Stability – Determines if the application is stable under varying loads [2].

2.1. Why do Performance Testing?

Performance testing identifies issues such as downtime, memory leakage, and thread lock [3]. It also uncovers potential problems like slow performance during concurrent use, inconsistencies across operating systems, and poor usability [2].

Performance Testing is done to provide stakeholders with information about their application regarding speed, stability, and scalability. More importantly, Performance Testing uncovers what needs to be improved before the

* Corresponding author: Vasudevan Senathi Ramdoss

product goes to market. Without Performance Testing, the software is likely to suffer from issues such as: running slow while several users use it simultaneously, inconsistencies across different operating systems, and poor usability.

Performance testing will determine whether their software meets speed, scalability, and stability requirements under expected workloads. Applications sent to market with poor performance metrics due to nonexistent or poor performance testing are likely to gain a bad reputation and fail to meet expected sales goals.

Also, mission-critical applications like space launch programs or life-saving medical equipment should be performance tested to ensure that they run for a long period without deviations.

2.2. Key Features of a Performance-testing Tool

Performance-testing tools simulate real users via "virtual" users and include features such as scalability, environment creation, scenario design, performance metrics monitoring, bottleneck identification, and comprehensive reporting [4, 5].

2.3. Breakdown of key features

- **Scalability -**The tool should be able to simulate a large number of users to test how the application performs under heavy load.
- **Environment Creation -**The ability to set up a realistic testing environment that mirrors real-world usage scenarios.
- **Scenario Design -** Flexibility to create diverse test scenarios that cover different user behaviors and system usage patterns.
- **Performance Metrics Monitoring -** Tracking key performance metrics like response time, throughput, CPU usage, memory utilization, and network latency.
- **Bottleneck Identification -** Advanced analysis capabilities to pinpoint performance bottlenecks within the application or infrastructure.
- **Reporting and Analytics -** Detailed and user-friendly reports that visualize test results, highlight performance issues, and allow for trend analysis.
- **Integrations -** Seamless integration with other development tools like CI/CD pipelines, monitoring systems, and defect tracking tools.
- **Cross-Platform Support -** Compatibility with various technologies and platforms, including web, mobile, APIs, and databases.
- **User-Friendly Interface -** Intuitive design that allows for easy setup, test creation, and result interpretation, even for non-technical users.
- **Load Simulation -** Ability to simulate different load types like spike loads, sustained loads, and ramp-up loads.

2.4. Types of performance testing:

The goal of performance testing is to identify and fix performance bottlenecks with various circumstances.

- **Load testing -** Evaluates how a system performs as the number of concurrent users increases.
- **Stress testing -** Also known as fatigue testing, this type of testing measures how a system performs under more users or transactions than normal.
- **Scalability testing -** Evaluates how well a system can adapt to changing needs, such as an increase or decrease in performance.
- **Spike testing -** Evaluates how an application performs when it experiences a sudden and extreme increase or decrease in load.
- **Volume testing -** Also known as flood testing, this type of testing evaluates how an application performs when it's handling large volumes of data.
- **Endurance testing -** Evaluates how well an application performs over a prolonged period of time.

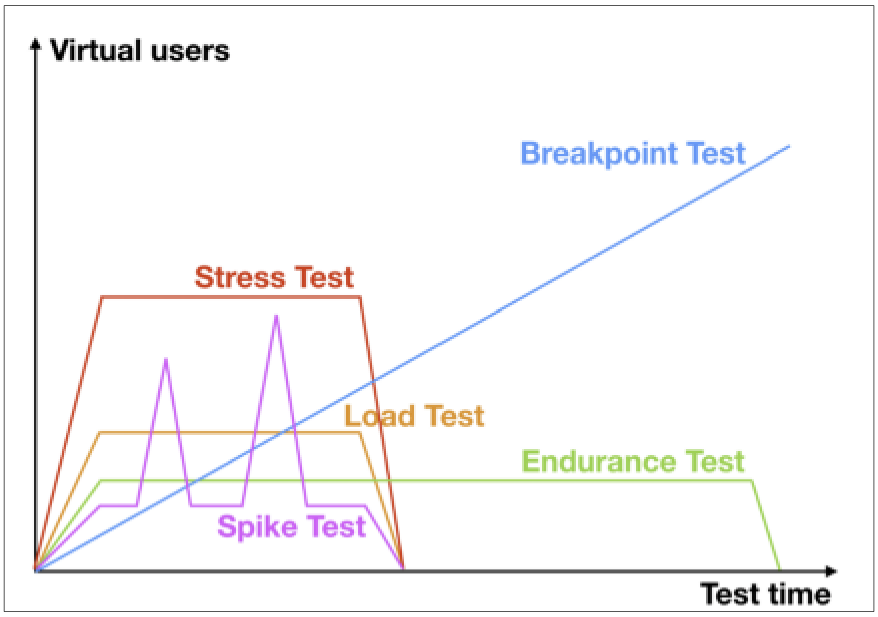

Figure 1 Type of Performance Testing

3. Performance Engineering

Performance engineering involves analyzing all code in development to ensure performance criteria are met. This ensures solutions satisfy performance expectations set during early development stages [6]. Collaboration between developers, architects, and performance engineers is critical for achieving optimal system performance [7].

The performance engineer provides a holistic view of all code in development to ensure performance testing criteria is comprehensive, encompasses the bigger picture, and factors all distinct pieces of code in development. The performance engineer is the primary user of performance testing tools and has a high degree of expertise in scripting, designing, running, and analyzing test results. Performance engineering brings the performance engineer to the early phase of development where they can provide performance metrics and scenarios required for code to be considered ready for release. Early involvement means the performance engineer can ensure the solution satisfies the performance expectations set out at the beginning of development. They also confirm that the architecture and design is consistent throughout development.

Performance engineering is transforming the software development landscape as well as the job descriptions of all who are engaged in it. And with a greater number of roles now involved, the need for tools and technology to streamline the process is greater than ever before. Performance engineering calls for end-to-end integration and collaboration from right-to-left and left-to-right along with real-time insights and analytics. Traditional performance testing vendors aren't adequately equipped to address this wave of confused change.

3.1. What does a performance engineer do?

- A performance engineer operates in a technical capacity, working to enhance system performance and efficiency. Their work involves:
- identifying bottlenecks
- finding issues that can slow down or hinder a software's performance
- designing optimizations to mitigate these issues
- They also evaluate system requirements and ensure the final product meets these demands. Here is a list of typical responsibilities for a performance engineer:
- System analysis
- Performance engineers analyze full-stack technologies to ensure systems are running effectively. This involves conducting tests, determining issues, and recommending solutions to these problems.
- Designing optimized solutions
- In addition to identifying problems, performance engineers design and implement solutions that improve system efficiency and overall performance.

- Collaborative improvement
- As well as working independently, performance engineers collaborate with developers, architects, and other team members. They share insights and find practical solutions to enhance system performance.

3.2. Where does a performance engineer work?

You can find performance engineers in a variety of work environments. The following are some familiar places where performance engineers are employed:

- Technology development companies
- Businesses with a strong online presence
- Consultancies specializing in technology solutions
- large corporations with in-house IT systems

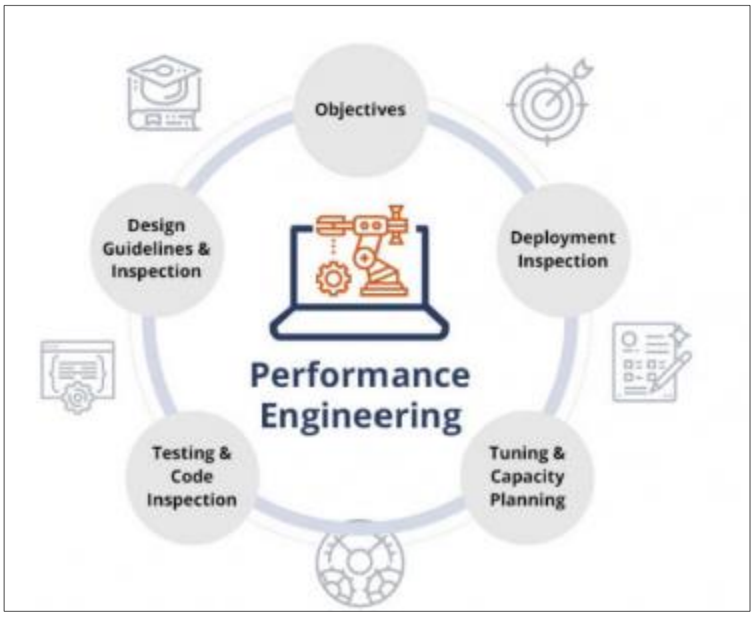
Objectives
Deployment Inspection
Tuning & Capacity Planning
Testing & Code Inspection
Design Guidelines & Inspection
Performance Engineering

Figure 2 Performance Engineering

3.3. Factors for Successful Performance Testing

Parameters complicating and thereby lengthening the task of performance testing are the following - complex tools, lack of experience, and lack of flexibility in the testing tools. Successful load testing requires that the Testers be trained well on tools, educated about the application domain, design and architecture to some extent, influence of technologies, and on performance testing methodologies.

Key steps involved in preparing for performance test are – 1. Prepare script, 2. Model and schedule workload, and 3. Execute Script. The support provided by a load-testing tool plays a major role in determining the cost-effectiveness, and the success of load-testing activity. The following are the factors for successful load-testing.

- **Speediness**: It is the only way that allows us to see if slower connections use more resources. However this may reduce the number of virtual users who may simultaneously visit a web site.
- **Behavior**: Load testing on just one browser is not sufficient. For obtaining insight into the error-free performance of the web-based applications, it is required to load-test on different browsers.
- **Complex scenarios to simulate user experiences**: To simulate a real user experience, the company that load tests needs to create a scenario where the information to do the test are provided to the testing browsers. The scenarios need to be closely alike the transactions performed by the real users of the Web site.
- **Analysis & Reporting**: Reporting of errors, time response, throughput information, resource utilization, network monitoring, etc. that help optimize the system performance is necessary.
- **Forecast**: With an advanced capacity planning process, a system behavior can be modeled and workload characteristics are forecast. However, the real-world performance needs to be predicted with high scale up factor.

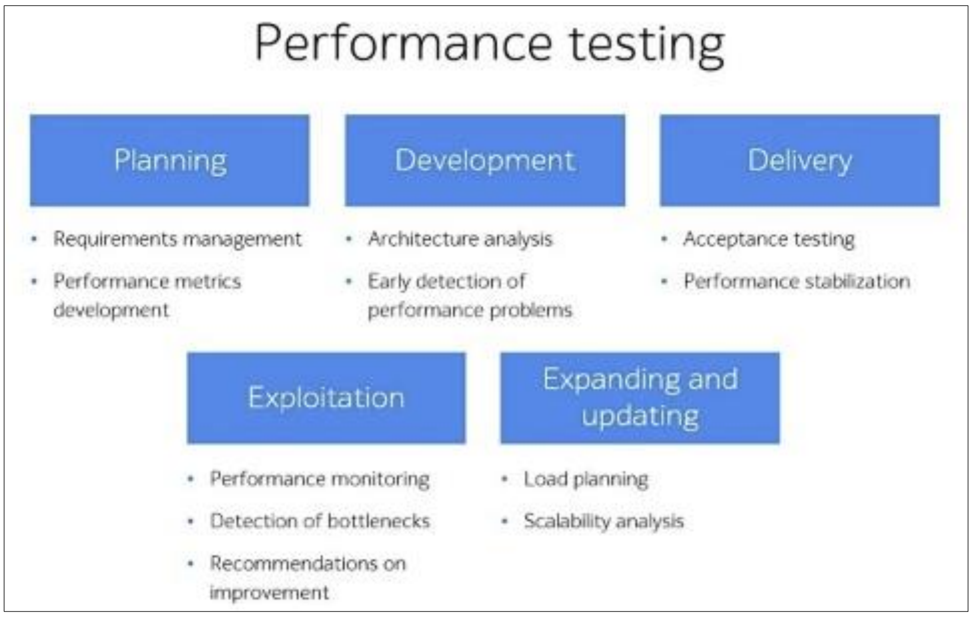

Figure 3 Performance Testing

3.4. Product/Project Performance

Testing performance early in the product lifecycle ensures robustness. For example, performance testing during requirement specification helps verify completeness and testability. During development, proper test data and performance analysis are crucial [8, 9].

Memory management, such as addressing memory leaks or optimizing garbage collection, also impacts overall performance [10]. Tools like profiling optimizations can tune performance in microprocessor-intensive environments [11].

Performance testing is considered as non-functional testing. During the requirement specification stage performance objectives must be analyzed to ensure completeness, feasibility, and testability. Feasibility could be ensured by simulation, or other modeling approaches. In the design phase, performance requirements must be applied to individual design components. Analysis of these components can help determine if the assigned requirements can be met. Simulation and modeling are the methodologies applicable to this task. During the development and deployment, performance analysis must be done at every level of testing, with careful construction of test data suitable to the performance testing scenarios. Profile guided performance optimizations are used for tuning the performance in modern microprocessors.

3.5. Benefits of performance testing

Performance testing brings in many advantages at various levels, be it business, project, process or product level. The following are a few benefits of performance testing [1,2,3].

- **Consistency**: Performance testing helps avoid deadlocks, improve response time; helps provide scalability, fault tolerance, recovery from failures, etc.
- **Availability [12]**: Performance testing reduces time to market greatly for large enterprise applications. In general if 98% of the high priority requirements are successfully tested it is then considered time to release to market. By treating the performance requirements that are treated non-functional, as part of high priority requirements, we can improve time-to-market, with considerable reduction in the test cycle, which results because of reduced defect rate.
- **Memory complications**: Memory leak, overflow, data inconsistency, byte order violation -are a few major problems that can be monitored, measured and controlled.
- **Vulnerability**: Performance testing helps ensure secure software by detecting memory overflows, and other resource vulnerabilities, for web based as well as desktop-based application.
- **Benchmarking**: This can allow testing Quality of Service of various architectural options.

- **Future expansions very easy [14]**: Performance testing helps accurately predict the required system and network capacity, which in turn helps in planning future expansions.
- **Service level tests**: Performance testing helps test various service levels to meet the challenges after deploying the product; thereby supports acceptance testing.

4. Sources of performance problems

To conduct effective and efficient performance testing, testers must be aware of the primary sources of performance issues within the system and its architecture. Below are some key factors contributing to performance problem?

- **Technologies**: Overhead in J2EE or heavy transaction systems [15]. While J2EE is known for scalability and high performance, intensive use of threading, rich environments, and heavy transactions can introduce overheads. J2EE's session affinity ensures requests for a particular session are directed to the same Web server or J2EE container, promoting load balancing and high availability. Conversely, technologies like .NET and COM/DCOM suffer from a high memory footprint, tightly coupled components, significant inter-module transactions, and poor load balancing support.
- **Database**: Co-hosting databases with applications causing performance degradation [16].Hosting a database on the same server as the web application can lead to severe performance degradation and security vulnerabilities. Additionally, using stored procedures—precompiled SQL statements—can help reduce network traffic and improve performance.
- **Languages**: Java is generally considered a high-performance language; however, its use of synchronization between threads can lead to resource locking, causing potential deadlocks and system breakdowns. If any part of the system fails, customers experience disruptions (e.g., "Page not available" errors), leading to abandonment of the service.
- **Network/Interconnection**: Communication delays and protocol inefficiencies [17]. Network traffic and communication delays are common causes of performance issues in distributed applications. High-speed Ethernet (HSE) combined with high-speed H1 field bus protocols offers an effective solution, supporting mission-critical applications by providing interoperability with TCP/IP and Ethernet.
- **Network Protocols**: The choice of network protocols can influence performance. For instance, SOAP can be slower and more resource-intensive than HTTP, which should be taken into consideration when optimizing network communication.
- **Mobile / Hybrid applications**: Interference in Bluetooth or Wi-Fi causing scalability problems [18]. Bluetooth wireless technology, operating in the 2.4GHz/5GHz range, facilitates personal area networks and ad-hoc connections. However, performance issues may arise from scalability problems, interference from other wireless technologies (e.g., Wi-Fi), or excessive concurrent connections.
- **Security Features**: Security mechanisms such as firewalls and encryption/decryption processes can increase access delays, thereby impacting performance.
- **Platforms/Servers**: The choice of server hardware significantly affects performance. Efficient servers, like those from Dell or HP Integrity series, should be prioritized, as inefficient servers can lead to poor performance.
- **Algorithms**: Complex processes like texture mapping slowing down performance [19]. Complex algorithms, such as those used in medical imaging (e.g., texture mapping, pixel data processing), can be performance-critical. Operations like scaled normalization, while improving visual quality, involve intensive calculations that may slow down display performance. A trade-off between performance and accuracy is often required, but improper optimization can result in critical system failures.
- **Internationalization**: Handling resource bundles for local languages often requires additional storage and can lead to memory issues such as overflow or byte-order violations. Challenges like date formatting, data sorting, and bidirectional text support for languages such as Hebrew or Arabic need to be carefully managed to ensure scalability and performance, especially in localized environments.

5. Performance & Monitoring tools [20, 21]

A number of open source and vendor specific tools are available for performance testing & monitoring.

- **Performance Testing tools** – NeoLoad, HP Load Runner, IBM Rational Performance Tester, Apace JMeter, Blaze meter, K9…
- **Monitoring tools** – Dynatrace, App Dynamics, Foglight, Splunk, Data Dog …..

6. Cloud Performance Testing

Cloud performance testing evaluates the performance and scalability of cloud-based applications under simulated real-world user loads. It ensures SLAs are met when migrating from on-premise to cloud environments [22]. Cloud performance testing is an aspect of software quality assurance. It is a measure of the performance and scalability of Cloud-based applications or services under simulated real-world user traffic and load conditions. While migrating from on-premise to the Cloud, you need to compare the performance benchmarks across the environments to ensure SLAs and user experience are met. Cloud is a technology that provides a scalable virtual distributed infrastructure to the application. Performance testing measures response times, verifies applications' responsiveness and ensures that they meet the required performance standards and service level agreements (SLAs) in a Cloud environment [10].

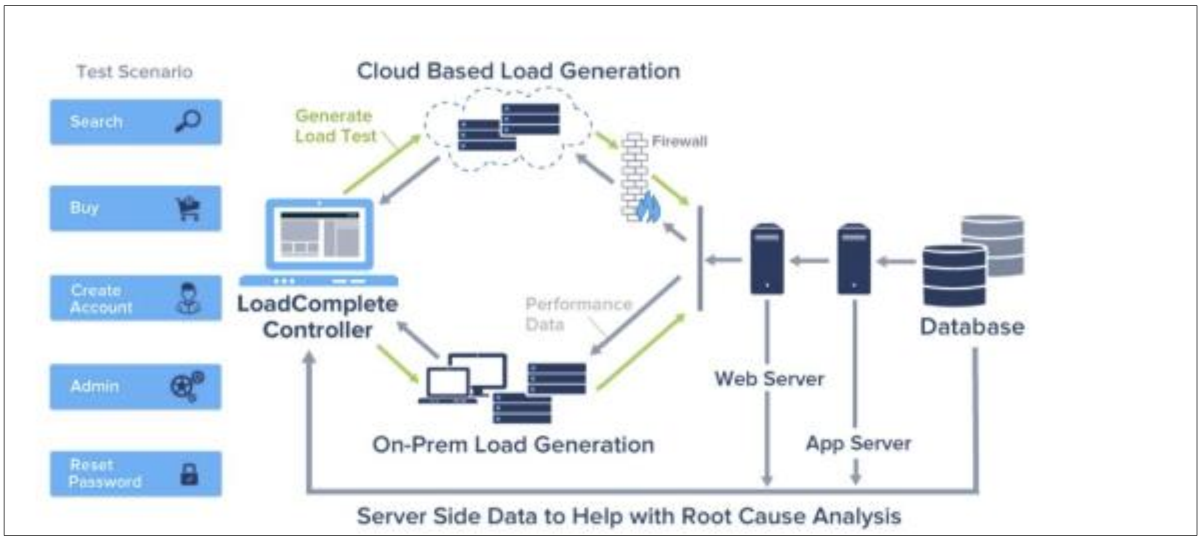

Figure 4 Cloud based Load Generation

6.1. How it works

- Subscription: Users pay a subscription fee to access the software.
- Hosting: A cloud service provider (CSP) hosts the software and infrastructure.
- Maintenance: The CSP manages the software, including updates and security patches.
- Access: Users access the software over the internet, usually through a web browser.

6.2. Benefits

- Cost savings: Users don't need to invest in expensive hardware to host the software.
- Low-effort updates: Users don't need to install software or patches.
- Mobility: Users can access the software from any device or location.

7. Challenges for Performance

The following are the challenges for any automated performance/load testing [23].

- Traceability of Requirements into the performance testing tool.
- Interface to test management tool
- Connection to defect management tool
- Support for internationalization
- Testing for availability, reliability, and recovery, load balancing, fault tolerance.
- Metrics for availability, reliability, failover cases and bandwidth usage. Silk performer has a rich set of metrics.
- Report Generation in html, graph plots, XML, MS Excel, MS word and PDF forms is desirable for the portability, security, and convertibility reasons.
- Support for post-production analysis

- Monitoring Power consumption at various parts – desirable for power aware systems. Any design of a performance test framework or tool needs to consider the above factors. Whenever a test plan is written, the above aspects need to be considered for preparing non-functional test requirement.

8. Conclusion

Performance testing & engineering is a more serious task than before, in the wake of emerging applications in healthcare, ecommerce, Automotive, Financial, etc…. Performance testing and engineering have become critical due to emerging applications in healthcare, e-commerce, and automotive industries. Starting performance analysis early in the product lifecycle ensures better-quality software. Testing must address real-time behavior and meet stringent quality standards [24].To ensure quality, the process real time behavior. Performance testing & engineering needs to be started in the early stages of the product development cycle for better quality.

References

[1] Tricentis NeoLoad, Performance Testing Tool. [Online]. Available: https://www.tricentis.com/products/performance-testing-neoload

[2] IBM Rational Performance Tester V1013. [Online]. Available: https://www.ibm.com/support/pages/rational-performance-tester-v1013

[3] Splunk, Performance Engineering Insights. [Online]. Available: https://www.splunk.com/en_us/blog/learn/performance-engineering.html

[4] DZone, Who is a Performance Engineer and How to Become One. [Online]. Available: https://dzone.com/articles/who-is-a-performance-engineer-and-how-to-become-on

[5] BlazeMeter. [Online]. Available: https://www.blazemeter.com

[6] Apache JMeter. [Online]. Available: https://jmeter.apache.org/

[7] Dynatrace. [Online]. Available: https://www.dynatrace.com

[8] AppDynamics. [Online]. Available: https://www.appdynamics.com

[9] Tricentis, Performance Engineering Challenges. [Online]. Available: https://www.tricentis.com/resources/performance-engineering-challenges-browser-based-testing

[10] K. Apte and S. Shah, Cloud-Based Performance Testing: A Comparative Study, Int. J. Comput. Sci. Eng., vol. 14, no. 6, pp. 45-49, 2022.

[11] Smith, Challenges in Automated Load Testing, J. Softw. Eng. Pract., vol. 12, no. 3, pp. 123-134, 2023.

[12] J. Doe, Internationalization Challenges in Performance Testing, J. Global Comput., vol. 10, no. 5, pp. 67-72, 2021.

[13] M. Lee, Diagnosing Performance Bottlenecks, Perform. Test. Today, vol. 5, no. 4, pp. 89-97, 2020.

[14] J. Smith, Future Trends in Performance Engineering, Eng. Today, vol. 8, no. 2, pp. 22-30, 2023.

[15] T. Brown, J2EE Performance and Scalability Challenges, Softw. Dev. Mag., vol. 6, no. 4, pp. 34-45, 2021.

[16] S. Johnson, Database Optimization Techniques, J. Database Pract., vol. 4, no. 7, pp. 67-82, 2022.

[17] P. Green, Network Protocols and Performance Impacts, Netw. Eng., vol. 9, no. 6, pp. 45-53, 2020.

[18] R. White, Scalability in Mobile Applications, Mobile Dev. Today, vol. 7, no. 5, pp. 39-50, 2021.

[19] L. Yang, Algorithm Optimization in Performance Engineering, Comp. Algorithms J., vol. 5, no. 3, pp. 101-110, 2020.

[20] Tricentis, Top Performance Testing Tools for 2023. [Online]. Available: https://www.tricentis.com/blog/top-performance-testing-tools-2023

[21] C. Evans, Monitoring Solutions in Cloud Environments, Cloud Eng. Mag., vol. 3, no. 8, pp. 55-63, 2022.

[22] G. Patel, Comparative Analysis of Cloud Testing Tools, Cloud Testing J., vol. 12, no. 4, pp. 78-89, 2023.

[23] H. Kumar, Automated Testing Challenges, Softw. Test. Pract., vol. 15, no. 9, pp. 120-134, 2021.

[24] B. Green, Emerging Trends in Performance Testing, Int. J. Perform. Test., vol. 11, no. 1, pp. 15-25, 2023.