Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 1, January 2025
DOI : https://doi.org/10.33395/sinkron.v9i1.14333
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Comparative Analysis of Express and Hono Framework Performance in Simple Registration Application

Anjar Tiyo Saputro*[1)], Mega Novita [2)]
[1)2)] Program Studi Informatika, Universitas PGRI Semarang
[1)]anjar.jog@gmail.com

Submitted :Dec 13, 2024 | **Accepted** : Jan 24, 2025 | **Published** : Jan 29, 2025

Abstract: This research evaluates the performance of two Node.js frameworks, Express and Hono, in developing a simple registration application. This application serves as a backend to store user registration data into a PostgreSQL database using the pg client of the node package manager (npm). The purpose of this performance comparison is to identify the framework that is superior in executing 1 million requests in this scenario. The analysis shows that Express has an average execution time of 26.85% faster than Hono. However, it is inversely proportional to the resource usage, where Hono shows better efficiency with lower CPU and memory usage of 29.29% and 19.97%. These findings provide important insights for developers in choosing a suitable framework based on performance and resource efficiency requirements.

Keywords: Express, Hono, PostgreSQL, JMeter, Docker

INTRODUCTION

Web application development has become an essential element in various industry sectors, ranging from business to education and healthcare. Web frameworks play a crucial role in accelerating the development process by providing tools and libraries that make it easier for developers to build functional and reliable applications. However, with so many web frameworks available, choosing the right one can be challenging for developers, especially when considering performance, scalability, and resource efficiency.

In this research, two web frameworks built for Node.js, Express and Hono, are the main focus of evaluation. Express has long been recognized as a popular, minimalist, and flexible framework for building web applications and Application Programming Interfaces (APIs). It is supported by a large community, a broad ecosystem, and comprehensive documentation. On the other hand, Hono is a newcomer designed for high efficiency by using standard web APIs and emphasizing speed and simplicity. The reason we chose Express and Hono is because they are both popular web development frameworks from the most widely used programming language, Javascript (Ahmod, 2023). The focus of this research is to evaluate the effectiveness of both frameworks in creating a simple registration application.

The main objective of this research is to evaluate the performance of the two frameworks in handling a simple registration application that stores user data in a PostgreSQL database. By testing the ability of each framework to handle one million requests, this research is expected to provide in-depth insight into the effectiveness and efficiency of the two frameworks. The main issue raised is the performance difference between an established and proven framework, such as Express, and a new framework such as Hono, which offers claims of superiority in speed. This research also aims to answer the important question of how developers can choose the optimal framework for their particular application needs (Muhammed et al., 2020). By conducting empirical data-based testing, the results of this research are expected to make a significant contribution to web development literature as well as industry practice (Diebold et al., 2018).

LITERATURE REVIEW

Express Framework

Express, also known as Express.js, is an open-source web framework developed using the JavaScript programming language. Express' main advantages lie in its flexibility and minimalist nature, making it a popular choice for building real-time, scalable, and efficient web and server-side applications. As one of the most widely used frameworks in the Node.js ecosystem, Express offers practicality and ease of application development. It extends the capabilities of Node.js' built-in HTTP module by providing additional functions to handle common

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

tasks, such as request management, routing, and response. Thus, developers do not need to implement these basic features from scratch, which speeds up the application development process.

One of the key features of Express is its support for middleware, which is a modular component that allows developers to add various additional functionalities to their applications. This middleware is very flexible and can be used for various purposes, including authentication, authorization, session management, and data request processing (Chandra & Tan, 2024). This broad middleware support gives developers the freedom to design applications according to their specific needs. In addition, the extensive documentation and large user community are a significant plus for Express. A wide range of tutorials, extended libraries, and discussion forums are available to help developers overcome technical challenges. Its high popularity means that Express is often the first choice for both novice developers new to Node.js, as well as experienced developers who need a reliable solution for large-scale projects. With its versatility, Express can be used for a wide range of applications, from simple APIs to complex applications with high scalability requirements.

Hono Framework

HONO is the latest open-source, full-stack web framework designed to make it easy to develop modern web applications quickly and efficiently. With a minimalist approach, the HONO Framework prioritizes the developer experience, making it suitable for beginners who want to learn modern web frameworks. One of Hono's main advantages is its blazing speed. This is possible thanks to the use of RegExpRouter, a router designed to process routes efficiently. With hono/tiny presets that are less than 14 kB in size, this framework is ideal for applications that require high performance with a minimal footprint. In addition, the framework is supported with a variety of built-in middleware while allowing developers to use custom or third-party middleware. This flexibility makes it easy to integrate additional features without having to start development from scratch.

Hono also features a clean and modern API, with full support for TypeScript. This provides a more structured, convenient, and error-free development experience. For developers who value productivity, Hono offers a solution that is intuitive and easy to adopt. In addition, the comprehensive documentation and active user community are significant pluses, providing adequate support for developers using the framework. Overall, Hono offers many advantages that make it an attractive choice in web development. With a combination of speed, efficiency, and modern feature support, Hono provides an innovative solution suitable for developers looking for a reliable framework for their web projects.

Previous Studies

Various previous studies have examined the performance of various web frameworks in application development. For example, a study conducted by Carmo et al. (2024) revealed that .NET has relatively high resource consumption compared to other frameworks such as Node.js and Django Rest Framework (DRF), which exhibit lower resource consumption with similar performance. However, in terms of response time, .NET stands out as the most effective, followed by Node.js and DRF.

Another study conducted by Yu et al. (2020) analyzed the impact of response time on mobile app user experience. The study also considered moderating factors such as gender and network conditions. The results show that slow response time negatively impacts user experience, affecting the dimensions of tolerance, acceptance, and satisfaction. Interestingly, this negative effect of response time was more significant in male users than female users. In addition, the combined influence of gender and network conditions resulted in a partially significant impact on user experience.

Smith (2012) also evaluated the resource efficiency of several web frameworks and found that frameworks with lightweight designs tend to be more efficient in CPU and memory usage. These findings underscore the importance of choosing an efficient framework for applications that require high performance and minimal resource consumption.

While many studies have evaluated the performance of various frameworks, research comparing established frameworks such as Express with new frameworks such as Hono is limited. This study aims to fill the gap by conducting a data-driven evaluation of the capabilities of both frameworks in handling simple registration applications. The emphasis on performance in scenarios with 1 million requests aims to provide a deeper insight into the resource efficiency and scalability capabilities of each framework.

METHOD

Research Flow

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 1, January 2025
DOI : https://doi.org/10.33395/sinkron.v9i1.14333
e-ISSN : 2541-2019
p-ISSN : 2541-044X

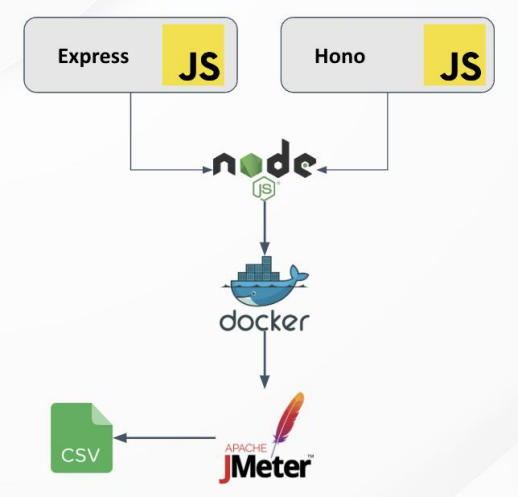

Figure 1. Testing Flowchart

The research method to be used in this study is comparative. Figure 1 shows the testing flow that will be performed on a local computer, with the test runner and the tested application running on the same device. The runtime to be used is Node with version 20.13.1. The database used is PostgreSQL version 16, which is a relational database management system that uses SQL as the main query language (Martins et al., 2021). The database will be run in a Docker container. The framework used in this test is Express version 4.19.2 and Hono version 4.4.6, which are installed via npm (Hafner et al., 2021).

The testing flow starts with writing server scripts using the Express and Hono frameworks in Javascript language. The database client used is *'pg'* as the client pool. The port used for the application is 3001. The application is then containerized with container names *infest-{nama_framework}-app* for the application and *infest-{nama_framework}-pg* for the database container. This container is run in Docker version 24.0.6 with a configuration, using 8 CPUs, 4 Giga Byte (GB) RAM, 1 GB Swap, and 64 GB virtual storage. After the application runs in Docker, testing is done using Apache JMeter (Nevedrov, 2006). The configuration is with 100 threads (concurrent users), 10,000 loops for a total of 1,000,000 requests. This scenario simulates 100 users accessing the application simultaneously and making 10,000 attempts so that there will be 1,000,000 requests by users (Jiang & Hassan, 2015). The test results will be recorded in a csv extension.

Implementation

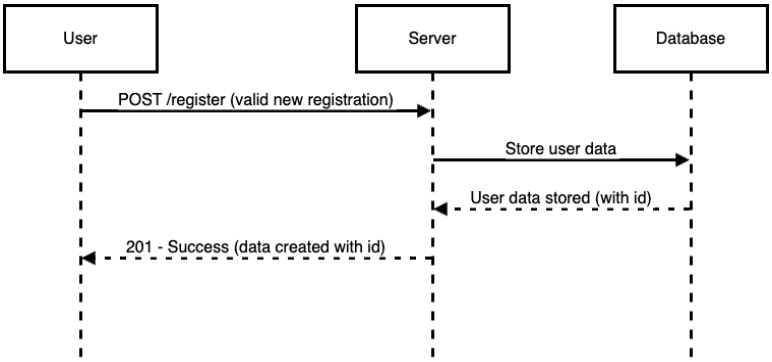

Figure 2. User Request Process by Server

Figure 2 explains how the server will execute the registration process. First, the user sends a POST request to the '/register' endpoint with the name, email and four-digit personal identification code (PIN). Then the server will process and store the data into the database. If the data is successfully stored, the server will respond with a unique id (Qin et al., 2009). This table is designed so that data is stored properly and uniquely, with the id column as the primary key (Katz et al., 1994).

Evaluation Criteria

This study compared the performance and resource usage between Hono and Express frameworks using metrics such as minimum response time, 10th percentile, first quartile, mean, median, 90th percentile, and maximum. These metrics help understand how fast the two frameworks respond to user requests, from the fastest

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 1, January 2025
DOI : https://doi.org/10.33395/sinkron.v9i1.14333
e-ISSN : 2541-2019
p-ISSN : 2541-044X

to the slowest (Glantz & Hurtig, 2022). In addition, this study compares resource usage such as average CPU and memory usage during testing, measured in percent (%) and Mega Byte (MB), as resource usage efficiency is crucial for an application to run well without consuming too much computing power or memory (Abouelyazid, 2022). In addition, the percentage of failed responses is also calculated (Shah & Pujara, 2020). With these metrics, this study provides a comprehensive overview of the performance and resource efficiency of both frameworks (Taley & Pathak, 2020).

Limitation

As with any study, there are some limitations to consider. One limitation is that performance results may vary depending on the specific hardware and server configuration used for testing (Fett et al., 2014). Also, the performance of simple registry applications may not reflect the performance of more complex applications or those used in the real world. In some cases, real-world applications require complex authorization or transaction workflows (Wieber, 2020). This research is still representative of real-world applications despite the use of simple tests because it uses the most commonly used programming languages.

RESULT

The testing flow started with writing a server script using the Express and Hono frameworks in Javascript. The server script was developed to handle the registration process, where the user sends a POST request to the '/register' endpoint with the parameters name, email, and four-digit PIN. The server then processes and stores this data into a PostgreSQL database using the 'pg' database client. Each entry is assigned a unique ID as the primary key to ensure data integrity. The application is packaged in a Docker container with the name infest-{framework name}-app for the application and infest-{framework name}-pg for the database. Testing is done in a Docker environment version 24.0.6, with a configuration of 8 CPUs, 4 GB RAM, 1 GB Swap, and 64 GB virtual storage.

Once the application was running in the container, testing was performed using Apache JMeter to assess the performance of both frameworks. The test configuration consisted of 100 threads each running 10,000 loops, totaling 1,000,000 requests. This scenario simulated 100 users accessing the application simultaneously, and each user performed 10,000 attempts to measure the framework's robustness and efficiency. The log data generated during the test was recorded in CSV format, including the response time and status of each request.

The test results were analyzed based on several key metrics, namely response time, resource usage, and failure rate. The scatterplot shown in Figure 3 shows the results of the one million request response time comparison test between the two frameworks, each represented in blue for Express and red for Hono. The x-axis represents the thread name. Thread is the unique naming of each request. While the y-axis shows the response time in milliseconds.

From Figure 3, it can be seen that the blue dots representing Express are mostly clustered at the bottom of the graph. This shows that Express has lower response time values in general. On the other hand, the red dots representing Hono are spread across a higher range of response times, indicating that Hono tends to have larger response times.

Express shows better consistency, with most of its response times being below about 50 milliseconds. In contrast, Hono shows a wider spread of response times, with some outlier values, up to 250 milliseconds. Express, on the other hand, has fewer outliers and these values do not reach as high as the Hono outliers. This suggests that Express is more stable in performance compared to Hono. Overall, Figure 3 shows that Express tends to give lower and more consistent response times, while Hono shows higher and more variable response times. This indicates that under the same conditions, Hono's performance is less stable than Express.

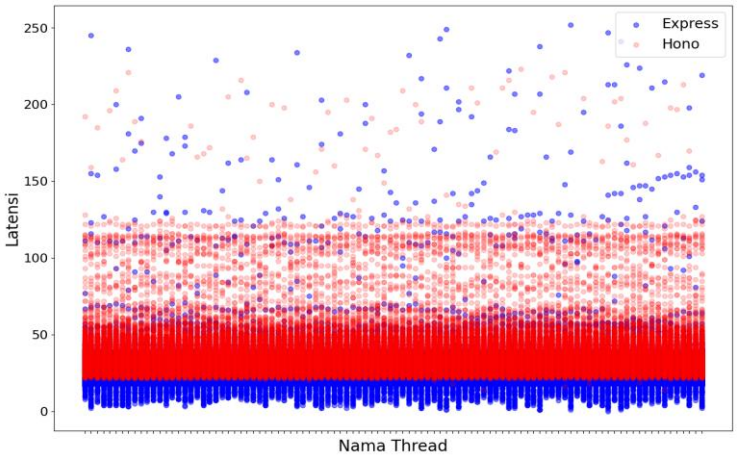

Figure 3. Scatter plot of response time distribution per thread

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 1, January 2025
DOI : https://doi.org/10.33395/sinkron.v9i1.14333
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Figure 4 shows the CPU usage over time for two frameworks, Express and Hono, with an application (app) and a PostgreSQL database container (pg), respectively. The x-axis represents the time series, while the y-axis shows the percentage of CPU utilization. The blue line representing infest-express-app consistently shows the highest CPU utilization, being in the range of 110-120%. The orange line representing infest-express-pg has a fluctuating CPU usage pattern, mainly between 50% to 80%. The green line representing infest-hono-app shows relatively high CPU utilization, averaging around 70-80% with some fluctuations. The red line representing infest-hono-pg shows low CPU utilization, mostly between 30% to 50%. In terms of stability and fluctuation, both infest-express-app and infest-hono-app show consistent CPU utilization patterns with small fluctuations. The infest-express-app container uses higher CPU than infest-hono-app.

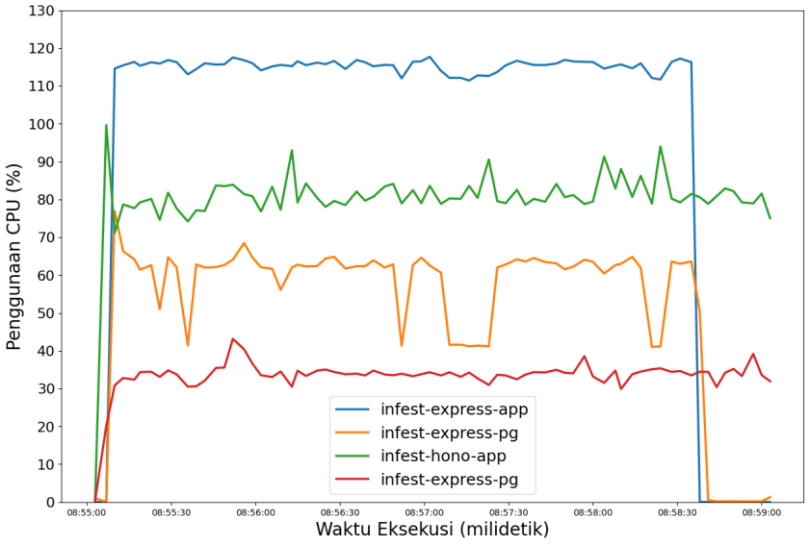

Figure 4. CPU utilization graph over time

From Figure 5, it can be seen that infest-express-pg with the orange line shows the highest memory usage, consistently around 160 MB, while infest-hono-pg with the red line is slightly lower but stable around 140 MB. The infest-express-app container with the blue line uses about 100 MB while infest-hono-app with the green line uses about 60 MB of memory. This shows that Hono uses less memory than Express in handling this scenario. The memory usage of both containers is also relatively stable over time with minor fluctuations. Overall, the Hono container (both application and database) consistently uses less memory compared to the Express container, showing better memory efficiency.

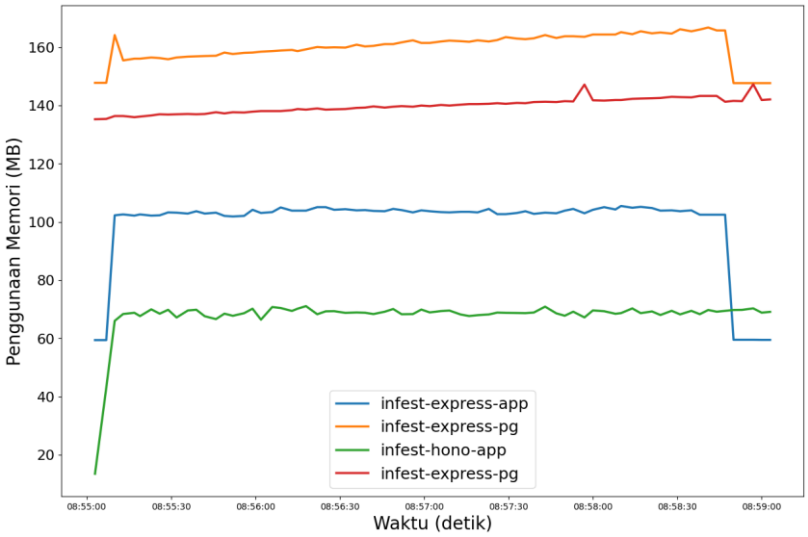

Figure 5. Memory usage graph over time

DISCUSSIONS

Formula (1) is used to calculate the margin of victory between the two samples presented in Table 1, namely Express represented by A and Hono represented by B. The margin of victory is calculated by taking the absolute difference between A and B (written as |A-B|), then dividing it by the highest value between A or B (written as

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 1, January 2025
DOI : https://doi.org/10.33395/sinkron.v9i1.14333
e-ISSN : 2541-2019
p-ISSN : 2541-044X

max(A,B)). The result of this calculation is then multiplied by 100 to get the value in percentage form. This formula gives an idea of how big the difference between the two samples is in the form of a percentage of the highest value between them.

Table 1. Performance and Resource Usage Test Results

Measurement	Express	Hono	Win	Winning Margin
Total time (seconds)	210,52	287,42	Express	26,75%
Requests per second (requests/second)	47,60	34,82	Express	26,85%
Lowest response time (milliseconds)	4,00	14,00	Express	71,43%
10 Percentile response time (milliseconds)	18,00	25,00	Express	28,00%
First quartile of response time (milliseconds)	19,00	26,00	Express	26,92%
Average response time (milliseconds)	21,01	28,72	Express	26,85%
Median response time (milliseconds)	20,00	28,00	Express	28,57%
90 Percentile of response time (milliseconds)	25,00	34,00	Express	26,47%
Highest response time (milliseconds)	252,00	223,00	Hono	11,41%
CPU utilization (%)	149,52	105,73	Hono	29,29%
Memory usage (MB)	259,33	207,54	Hono	19,97%
Request error (%)	0,00	0,00	Balanced	0,00%

$$Margin\ Kemenangan = \frac{|A-B|}{max(A,B)}\ \% \qquad (1)$$

From Table 1, it can be seen that Express shows superior performance with a total execution time of 210.52 seconds, compared to Hono which requires 287.42 seconds. This means that Express is 26.75% faster in completing 1 million requests. In addition, Express is also able to handle 47.60 requests per second, while Hono can only handle 34.80 requests per second, showing higher efficiency in handling simultaneous requests. In terms of response time, Express also shows a significant advantage. Express' lowest response time is 1 millisecond, much faster than Hono which has a lowest response time of 14 milliseconds. At various response time percentiles, Express is consistently faster, such as at the 10th percentile (8 milliseconds vs 24 milliseconds), first quartile (14 milliseconds vs 29 milliseconds), mean (21.01 milliseconds vs 28.75 milliseconds), median (19 milliseconds vs 25 milliseconds), third quartile (26 milliseconds vs 32 milliseconds), and 90th percentile (35 milliseconds vs 43 milliseconds). In terms of request error, both frameworks show a balanced performance with 0% error percentage for both. This shows that both Express and Hono were able to handle requests without errors in this test scenario.

Overall, the test results show that Express is faster in terms of execution time and response time than Hono. However, it should be noted that the higher execution speed of Express is due to its greater resource usage. Hono, although slower, proved to be more efficient in CPU and memory usage, which can be an important consideration in resource-constrained environments. These findings provide important insights for developers in choosing the appropriate framework based on their specific needs. If execution speed is a top priority, Express may be a better choice. However, if resource efficiency is more important, Hono could be a more suitable alternative.

CONCLUSION

This research has evaluated the performance of two Node.js frameworks, Express and Hono, in developing a simple registration application. The results show that Express has an average execution time of 26.85% faster than Hono. However, Hono is 29.29% more efficient in CPU usage and 19.97% in memory usage. These results provide important insights for developers in choosing a framework based on their specific needs. However, other things such as infrastructure, updates, and different use cases may affect the performance and efficiency of the tested frameworks. Future research is expected to extend this analysis by evaluating aspects such as developer support, documentation, update frequency, community support, and issue resolution to provide a more complete picture of which frameworks are better in various development contexts. Thus, this research can continue to contribute to the advancement of knowledge in the field of web development and help developers make more informed decisions.

REFERENCES

Abouelyazid, M. (2022). Forecasting Resource Usage in Cloud Environments Using Temporal Convolutional Networks. *Applied Research in Artificial Intelligence and Cloud Computing*, *5*(1), 179–194.

Ahmod, M. F. (2023). *Javascript runtime performance analysis: Node and Bun*.

Carmo, K. X., Ferreira, F., & Figueiredo, E. (2024). Performance Evaluation of Back-End Frameworks: A Comparative Study. *Proceedings of the 20th Brazilian Symposium on Information Systems*, 1–9. https://doi.org/10.1145/3658271.3658314

Diebold, P., Schmitt, A., & Theobald, S. (2018). Scaling agile. *Proceedings of the 19th International Conference*

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 1, January 2025
DOI : https://doi.org/10.33395/sinkron.v9i1.14333
e-ISSN : 2541-2019
p-ISSN : 2541-044X

on Agile Software Development: Companion, 1–4. https://doi.org/10.1145/3234152.3234177

Fett, D., Kusters, R., & Schmitz, G. (2014). An Expressive Model for the Web Infrastructure: Definition and Application to the Browser ID SSO System. *2014 IEEE Symposium on Security and Privacy*, 673–688. https://doi.org/10.1109/SP.2014.49

Glantz, I., & Hurtig, H. (2022). *Express. js and Ktor web serverperformance: A comparative study*.

Hafner, A., Mur, A., & Bernard, J. (2021). Node package manager's dependency network robustness. *ArXiv Preprint ArXiv:2110.11695*. https://doi.org/10.48550/arXiv.2110.11695

Jiang, Z. M., & Hassan, A. E. (2015). A Survey on Load Testing of Large-Scale Software Systems. *IEEE Transactions on Software Engineering*, *41*(11), 1091–1118. https://doi.org/10.1109/TSE.2015.2445340

Katz, E. D., Butler, M., & McGrath, R. (1994). A scalable HTTP server: The NCSA prototype. *Computer Networks and ISDN Systems*, *27*(2), 155–164. https://doi.org/10.1016/0169-7552(94)90129-5

Martins, P., Tomé, P., Wanzeller, C., Sá, F., & Abbasi, M. (2021). *Comparing Oracle and PostgreSQL, Performance and Optimization BT - Trends and Applications in Information Systems and Technologies* (Á. Rocha, H. Adeli, G. Dzemyda, F. Moreira, & A. M. Ramalho Correia (eds.); pp. 481–490). Springer International Publishing.

Muhammed, T., Mehmood, R., Abozinadah, E., & Sharaf, S. (2020). *SelecWeb: A Software Tool for Automatic Selection of Web Frameworks* (pp. 329–346). https://doi.org/10.1007/978-3-030-13705-2_14

Nevedrov, D. (2006). Using jmeter to performance test web services. *Published on Dev2dev*, 1–11.

Qin, L., Yu, J. X., & Chang, L. (2009). Keyword search in databases. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, 681–694. https://doi.org/10.1145/1559845.1559917

Shah, M., & Pujara, N. (2020). A review on software defects prediction methods. *ArXiv Preprint ArXiv:2011.00998*. https://doi.org/10.48550/arXiv.2011.00998

Smith, P. G. (2012). *Professional website performance: optimizing the front-end and back-end*. John Wiley & Sons.

Taley, D. S., & Pathak, B. (2020). Comprehensive study of software testing techniques and strategies: a review. *Int. J. Eng. Res*, *9*(08), 817–822.

Wieber, N. (2020). Automated generation of client-specific backends utilizing existing microservices and architectural knowledge. *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 1158–1160. https://doi.org/10.1145/3324884.3415283

Yu, M., Zhou, R., Cai, Z., Tan, C.-W., & Wang, H. (2020). Unravelling the relationship between response time and user experience in mobile applications. *Internet Research*, *30*(5), 1353–1382. https://doi.org/10.1108/INTR-05-2019-0223

*name of corresponding author