Evaluating The Performance, Scalability, and Adoption of JavaScript, PHP, and Ruby

Dr. Binod Kumar Acharya

Assistant Professor, Department of Software Development,
Sikkim Government Vocational College Dentam, Sikkim.

ABSTRACT

Programming languages have a significant impact on contemporary web application development in terms of execution speed, scalability, resource utilisation, and developer productivity. In this study, we compare and contrast three popular languages—JavaScript, PHP, and Ruby—using standardised computational benchmark tests such as merge Sort, Prime Number Generation, and Fibonacci Series execution. The results show that JavaScript, which is powered by the V8 engine in Node.js, consistently outperforms the other two languages in computationally intensive tasks, with lower execution times and superior scalability. Besides performance, the study delves into language adoption trends, ecosystem maturity, framework support, scalability, and developer productivity, offering a comprehensive overview of the subject.

Keywords: *JavaScript, Ruby, Web Application, Benchmark, Developer, Runtime*.

I. INTRODUCTION

Programming languages provide the cornerstone of contemporary software development by offering a systematic framework for developers to create algorithms, execute business logic, and construct applications in various computer settings. Each software system, whether online applications, mobile platforms, business solutions, cloud services, or embedded systems, depends on one or several programming languages to convert computational needs into runnable programs. In recent decades, programming languages have seen considerable evolution, with refined syntax, advanced execution models, superior memory management, fortified security protocols, and comprehensive standard libraries. These innovations have empowered developers to produce software that is swifter, more dependable, scalable, and easier to maintain.

The swift advancement of information technology has resulted in progressively intricate software systems that need superior performance, adaptability, maintainability, and interoperability. Historically, software programs were created utilising a singular programming language for the complete development process. Although this unified methodology streamlined both creation and upkeep, it frequently constrained developers from leveraging specialised functionalities present in alternative programming languages. As software architectures have evolved to be more distributed and component-based, organisations are increasingly aware that no single programming language can adequately fulfil both functional and non-functional demands of contemporary applications.

Every programming language has distinct traits, advantages, and constraints that render it appropriate for specific types of applications. JavaScript, for example, prevails in client-side web development and has notably extended into server-side programming via runtime environments like Node.js. PHP continues to be one of the most prevalent scripting languages for dynamic web application development due to its ease of use, vast ecosystem, and smooth integration with web servers and relational databases. Ruby, especially when utilised with the Ruby on Rails framework, prioritises developer efficiency via convention-over-configuration methodologies, refined syntax, and swift application development. As a result, developers are progressively choosing programming languages tailored to certain project needs instead of depending on a singular, all-encompassing answer.

The increasing intricacy of corporate software has hastened the embrace of multi-language software development, also known as polyglot programming. Polyglot programming involves the use of many programming languages inside a singular software framework to leverage the advantages of each language while mitigating their respective drawbacks. Rather of constructing every application layer with a uniform programming language, developers choose the best suitable language for each subsystem, including frontend interfaces, backend services, database interactions, analytics engines, and machine learning components.

This development has led to the notion of multi-language compatibility, which denotes the capacity of a software platform, framework, runtime environment, or application architecture to accommodate and include many programming languages inside a single project. Multilingual compatibility allows software components developed in many languages to interact efficiently, share data effortlessly, and operate cohesively as a singular system. This compatibility affords software programmers enhanced latitude in choosing technologies according to performance criteria, development efficacy, existing knowledge, and the availability of libraries.

The significance of multi-language compatibility has grown considerably with the advent of distributed computing, cloud-native apps, containerised deployments, microservices architecture, serverless computing, and Application Programming Interfaces (APIs). Contemporary corporate applications seldom function as standalone programs; rather, they are composed of several interlinked services created by autonomous teams utilising various programming languages. A frontend interface constructed using JavaScript can interact with backend services programmed in PHP or Ruby via RESTful APIs, GraphQL interfaces, or message queues. Such diverse designs necessitate strong interoperability frameworks to provide effective communication among various language ecosystems.

Numerous technologies and approaches enable multi-language compatibility in contemporary software systems. A prevalent method is using runtime environments that offer uniform execution platforms for many programming languages. Node.js allows JavaScript to run outside web browsers, whereas language-specific virtual machines and interpreters provide uniform execution across various operating systems. These execution environments frequently include interfaces that enable interaction with components created in different programming languages.

Factors Influencing Language Adoption

A multitude of reasons propels the utilisation of a programming language in the industrial sector, often reliant on the distinct demands of the company and the area of application.

- **Performance:** Languages like C, C++, and Rust are preferred in system-critical and high-performance applications due to their efficient memory management and speed. While languages like Ruby, PHP, and JavaScript may not be the most performant choices for web development, their developer productivity makes them quite popular.
- **Ease of Learning:** Because of its easy-to-understand syntax, which novices can pick up quickly and which companies can use to teach new engineers, Python has witnessed broad acceptance across many sectors. The higher learning curves of more powerful languages, such as C++ and Rust, might prevent their adoption in projects with limited resources or short timeframes, despite their superior capabilities.
- **Community Support and Ecosystem:** A language's adoption rate is greatly affected by its development community and the number of libraries or frameworks it has. For example, JavaScript is great for both front-end and back-end development because of its extensive ecosystem of libraries and frameworks, such as React and Node.js. One reason Python is so popular in data science is because of its extensive ecosystem, which includes libraries like Pandas, NumPy, and TensorFlow.

- **Legacy Systems:** Legacy systems developed in earlier languages such as COBOL, Java, or C are still in use by many sectors, including healthcare and the financial sector. More recent languages are available, but the expense and risk of redesigning these systems typically make them unusable.
- **Industry-Specific Needs:** Because of the specialised nature of their work, several sectors have a preference for a certain language. When it comes to data science and machine learning, Python is king, but when it comes to game creation and HPC, C++ is queen. Because of its reliability, scalability, and long-term support, Java is still widely used in business contexts.

Language Adoption Across Industries

- **Web Development:** JavaScript, especially when utilised with frameworks such as React and Node.js, is the preeminent language in online development. This facilitates the creation of comprehensive applications authored in one language. PHP and Ruby are commonly employed for server-side scripting because of their user-friendliness and swift development processes.
- **Data Science and Machine Learning:** Python dominates these domains because to its vast array of scientific libraries and tools. R is extensively utilised, especially in academic settings and sectors that depend significantly on statistical analysis.
- **Finance:** Java and C# continue to be the predominant programming languages in the finance sector, especially for developing high-frequency trading systems and extensive corporate applications. Python is progressively utilised for data examination and automation inside this industry.
- **Embedded Systems and IoT:** C and C++ continue to be the favoured programming languages for embedded systems because of their low-level management and effectiveness. Rust is attracting interest in this domain because to its safety attributes, especially in essential applications such as automotive software.
- **Game Development:** C++ and C# (especially inside Unity) prevail in the realm of game development, providing the requisite performance and control for graphics-intensive, real-time applications.

II. REVIEW OF LITERATURE

Shukla, Abhishek. (2023). Since its inception as a scripting language for use on the client side, JavaScript has grown into a vital programming language with applications throughout the whole software stack. This article takes a look at JavaScript's complicated history, following it from its beginnings to its current significance. This study looks at the language's evolution, highlighting how ECMAScript standardisation affected it and how ES6 significantly improved its capabilities and made it more maintainable. The introduction of ES6 modules has promoted modularity and code reuse, drastically changing the structure of code. Modern JavaScript frameworks like Vue.js, Angular, and React are discussed in this article as they pertain to modern web development. By providing a wide range of capabilities for creating interactive and dynamic user interfaces, these frameworks have revolutionised front-end development. Improved usability is a result of React's virtual DOM and component-oriented architecture. At the same time, the whole framework of Angular and Vue.A wide variety of developer preferences may be accommodated by JS's gradual process. The study confirms the innovations' broad approval, which is largely attributable to their developer-friendliness and strong community backing.

Islam, Nawroze et al., (2023). Choosing a programming language is vital since it affects the project's end result, especially considering how fast software development is changing. Two prominent languages in modern software development are Python and PHP, which are both extensively utilised. The investigation begins with a thorough and enlightening comparison of the two languages. Finding out how PHP and Python stack up in terms of performance is the main focus of this study. Legislators and programmers alike will find this data to be quite useful. The end goal is to empower participants with a thorough guide that helps them navigate the complex decision-making process and pick the programming language that fits their specific project needs.

This study goes further than previous ones in its analysis of performance. When assessing the efficacy of software development, quantitative and qualitative factors are also considered. The capacity to make decisions that go beyond the scope of the current endeavour is one of the primary goals of this study. This will make it easier to scale, adapt, and achieve long-term success. As software environments change, this comparison helps people navigate the intricacies of choosing a programming language. Beyond technical details, the study provides a strategic understanding of how PHP and Python function in software development environments as a whole. Pros, cons, and best uses of PHP and Python will be illuminated for developers and decision-makers by this investigation. The choice of programming language will be more thoughtful and careful as a consequence of this.

Laaziri, Majida et al., (2019). A PHP framework is a crucial element in the arsenal of every web developer, providing efficiency, reliability, maintainability, and scalability—qualities that are more sought after in the growing field of online development. PHP frameworks are increasingly favoured in web projects for their capacity to enhance development efficiency, save time expenditure, and comply with coding conventions. These frameworks are intended to facilitate and expedite software development. Code that is both resilient and simple to modify. Utilising the PHP framework will provide more secure and reliable online applications. A multitude of PHP frameworks exist in contemporary times. Despite the plethora of outstanding PHP frameworks available, it is crucial for developers to possess a comprehensive understanding of each to make a well-informed choice. To aid developers in making an educated choice about a PHP framework for upcoming projects, this research analysed three prominent frameworks and conducted assessments and performance evaluations.

Verdú, Javier & Pajuelo, Alex. (2016). Through the use of novel standard-based technologies, web applications are increasingly resembling native applications. A feature of the updated HTML5 standard is the Web Workers API, which facilitates the execution of JavaScript applications over many threads, or workers. Nonetheless, the internal mechanisms of the browser's JavaScript virtual engine obscure any direct relationship between the quantity of workers and threads functioning inside the browser and the utilisation of the processor's logical cores. As a result, developers remain uncertain about the optimal quantity of workers for concurrent JavaScript scripts and the actual performance enhancement across different environments. This paper is the first investigation into the scalability of performance for web applications operating concurrently with several workers. This research primarily focuses on two case studies that exemplify different worker execution approaches. Our investigation indicates that performance varies across three widely used web browsers and many parallel processor microarchitectures. Furthermore, we examine the impact of application co-execution on the performance of online applications. These results carry significant consequences for forthcoming strategies designed to autonomously ascertain the ideal workforce size to preserve system responsiveness and user experience amid unexpected fluctuations in system activity, while simultaneously enhancing resource efficiency.

Abdullah, Mahmood. (2010). This research is centred on comparing web server resources. It was mentioned that two of the most well-known web programming languages are PHP and JavaScript. A method for finding one's age that may be done online. Our comparisons covered a wide range of metrics, including response time and CPU load. It was decided that PHP would deliver the best results in some cases, even while JavaScript is more appropriate in others.

III. MATERIALS AND METHODS

The version of PHP utilised in this investigation is 7.1.3. The Zend Engine is the core execution engine utilised by PHP versions later than 5. Zend Engine 3.0, which is used by PHP 7, reduces memory use and improves execution performance compared to earlier versions. In this study, the web server Apache 2.4.25 is used to run the PHP environment.

The version of NodeJS 9.5.0 was utilised in this research. Using Just-In-Time (JIT) compilation, Node.js is built on top of Google's V8 JavaScript Engine. This engine converts JavaScript into native machine code. You may manage packages and install dependencies with the help of the Node Package Manager (NPM) 5.6.0.

Another server-side technology that has been provided for comparison is Ruby 2.5.1. The Ruby programs were deployed using the Puma 3.11 application server and ran using the Ruby Interpreter (MRI 2.5.1). To install the necessary libraries and dependencies, RubyGems was utilised as the package management system.

A recommended configuration for testing purposes is a system with 16 GB of RAM and an Intel Core i7-7700HQ CPU running at 3.8 GHz. To make sure we were comparing PHP, NodeJS, and Ruby fairly, all of the benchmark applications were run on the same hardware and operating system. Table 1 displays the technical details of the software and server infrastructures.

Table 1. Technical Details of Server and Software Environments

Environment	Feature
Server	CPU: Intel Core i7-7700HQ 3.8 GHz RAM: 16 GB OS: Windows 10 (64-bit) HDD: 1 TB SSD: 128 GB
PHP	Apache: 2.4.25 PHP: 7.1.3 MySQL: 5.7.17 PhpMyAdmin: 4.7.0
NodeJS	NodeJS: 9.5.0 Node Package Manager (NPM): 5.6.0
Ruby	Ruby: 2.5.1 Puma: 3.11 RubyGems: 2.7.6

To evaluate each setting, three distinct benchmarks were used. An O(N log N) divide-and-conquer sorting algorithm called Merge Sort was the first benchmark. The benchmark tested how long it took for PHP, Node.js, and Ruby to sort randomly generated arrays of varying sizes. Running a program to produce prime numbers inside certain ranges was the subject of the second benchmark. All three settings used the same technique and computational logic to guarantee that variations in execution time were due only to the programming languages' runtime performance. Thirdly, we used an iterative method to generate the Fibonacci sequence up to a given number. When it came time to do repeated numerical computations, the benchmark determined which environment was the most efficient.

The other benchmarks comprised further assessments of computational and web-service performance using the same implementations in Ruby, Node.js, and PHP. To reduce experimental variance and provide accurate performance comparisons across the three technologies, we ran each benchmark 10 times and averaged the execution times.

IV. EXPERIMENTAL RESULTS

Four separate benchmark tests were run in each of the three environments to compare PHP, NodeJS, and Ruby. For the sake of objectivity, all of the benchmark programs used the same algorithms when they were launched. We averaged the execution time of each experiment after repeating it 10 times.

Merge Sort Algorithm

The first benchmark measured the execution time of the Merge Sort algorithm. Arrays including 10,000, 100,000, 1,000,000, and 2,000,000 randomly generated numbers were organised using an identical divide-and-conquer algorithm in PHP, NodeJS, and Ruby.

The experimental findings demonstrate that NodeJS attained the minimal execution duration, followed by Ruby, whilst PHP necessitated the maximal execution duration. Ruby shown considerable advancement compared to PHP due to enhancements in the Ruby 3.x interpreter and refined memory management, however NodeJS retained its superiority thanks to Google's V8 JavaScript engine.

Table 2. Execution Time of Merge Sort Algorithm

Array Length	Number Range	NodeJS (s)	PHP (s)	Ruby (s)
0–10,000	0–10,000	0.0196	0.0304	0.0248
0–100,000	0–100,000	0.1005	0.3515	0.2124
0–1,000,000	0–1,000,000	1.0104	4.2118	2.1435
0–2,000,000	0–2,000,000	1.2415	8.7378	2.8752
0–10,000	Fixed (0–10,000,000)	0.0099	0.0297	0.0185
0–100,000	Fixed	0.0611	0.3607	0.1438
0–1,000,000	Fixed	0.6006	4.3138	1.0124
0–2,000,000	Fixed	1.2408	9.1579	1.9637

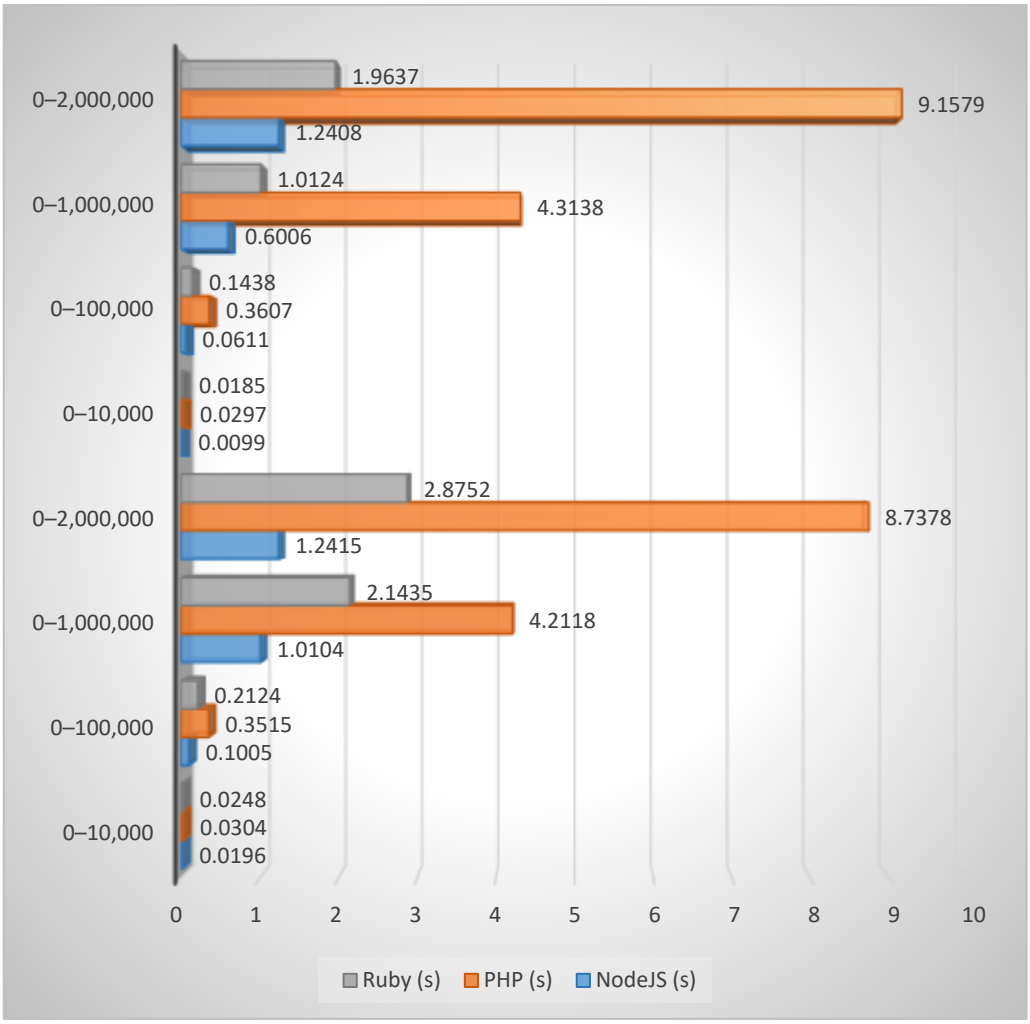

Figure 1. Execution Time of Merge Sort Algorithm

The findings indicate that NodeJS consistently recorded the minimal execution time across all array dimensions. Ruby had much superior performance compared to PHP, achieving a reduction in execution time of around 45–60%, although NodeJS outperformed Ruby by 35–50% for extensive datasets.

Prime Number Generation

The second benchmark assessed the time taken to calculate prime numbers over various numerical intervals using the same techniques. NodeJS once again shown superior computing efficiency, although Ruby executed the computations more swiftly than PHP, however not as rapidly as NodeJS.

Table 3. Execution Time of Prime Number Program

Prime Number Range	NodeJS (s)	PHP (s)	Ruby (s)
0–500,000	4.00	35.88	15.64
0–1,000,000	11.49	89.27	41.38
0–2,000,000	33.22	279.95	116.73

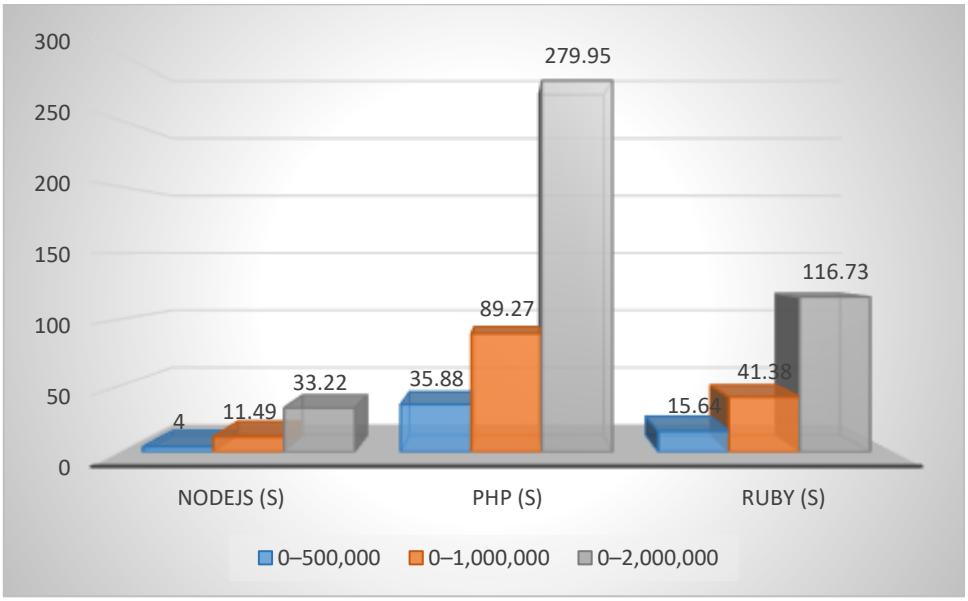

Figure 2. Execution Time of Prime Number Program

The benchmark results show that compared to the other settings, Node.js generated prime numbers far more quickly. While Node.js continued to be three to four times quicker than Ruby for bigger datasets, Ruby shortened execution time by around 58% relative to PHP.

Fibonacci Series

The third standard used an iterative process to calculate the initial 1000 Fibonacci numbers. Due to the minimization of runtime cost caused by the iterative loop, PHP outperformed the prior benchmarks in terms of execution time. Due to JavaScript runtime cost, Node.js displayed somewhat longer execution time, whereas Ruby needed marginally more processing time than PHP.

Table 4. Fibonacci Program Execution Time

Environment	Execution Time (Second)
NodeJS	0.003618
PHP	0.000113
Ruby	0.000247

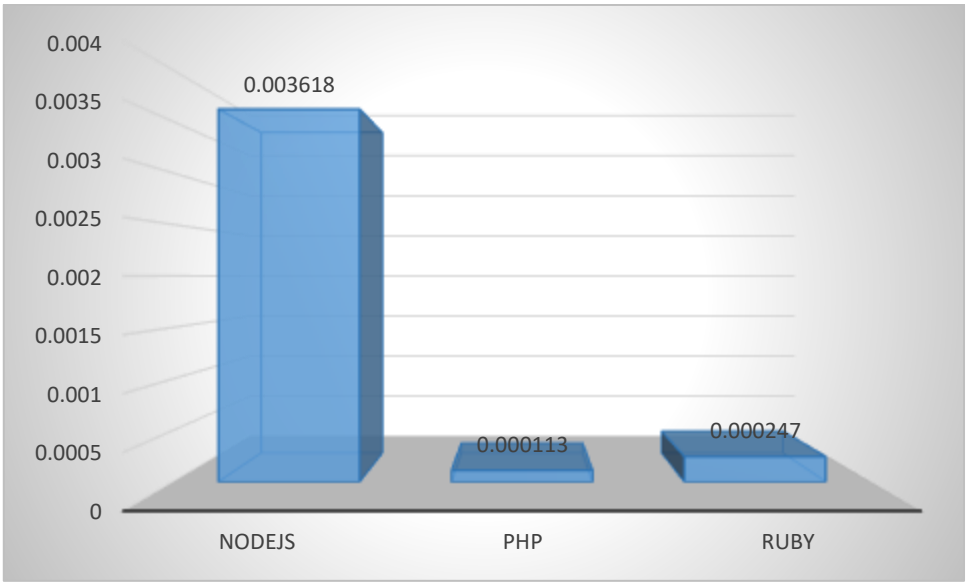

Figure 3. Fibonacci Program Execution Time

In this benchmark, PHP performed the best, with Ruby remaining around 2.2 times slower than PHP but significantly quicker than NodeJS.

V. CONCLUSION

By analysing their performance, scalability, and industry acceptance using standardised benchmark testing, this study provided a thorough comparison of JavaScript (Node.js), PHP, and Ruby. Since the V8 engine is so efficient, the trial findings showed that JavaScript routinely performed better than other languages on computationally expensive tasks. This makes JavaScript an excellent choice for contemporary, scalable online applications. With its balanced performance, simple grammar, and fast application development capabilities, Ruby is a top choice for developers. Despite doing somewhat worse on some benchmarks, PHP's stable ecosystem, significant framework support, and simplicity of deployment make it a reliable and popular server-side language. Project demands, anticipated workload, scalability requirements, maintainability, and development efficiency should be considered while choosing a programming language, according to the results. No single language is suitable for all applications.

REFERENCES

1. M. Szewczyk and M. Skublewska-Paszkowska, “Performance comparison of development frameworks in selected environments in REST API architecture,” *Journal of Computer Sciences Institute*, vol. 35, no. 1, pp. 121–128, 2025.
2. H. Alnuhait, W. Al-Zyadat, A. Thunibat, H. Kahtan, B. Zaqaibeh, and H. Al-Khawaja, “Web application performance assessment: A study of responsiveness, throughput, and scalability,” *International Journal of Advanced and Applied Sciences*, vol. 11, no. 9, pp. 214–226, 2024.
3. D. Đurđev, “Popularity of programming languages,” AIDASCO Reviews, vol. 2, no. 2, pp. 24–29, 2024.
4. Shukla, “Modern JavaScript frameworks and JavaScript's future as a full-stack programming language,” *Journal of Artificial Intelligence & Cloud Computing*, vol. 2, no. 2, pp. 1–8, 2023.
5. N. Islam, M. R. Chowdhury, and S. Islam, “A comparative analysis of PHP and Python programming languages for optimal software development,” *International Journal of Information Technology*, vol. 8, no. 1, pp. 1–13, 2023.

6. S. Tenzin, "PHP framework for web application development," *International Advanced Research Journal in Science, Engineering and Technology (IARJSET)*, vol. 9, no. 2, pp. 110–113, 2022.
7. R. M. Lerner, "Design patterns in modern JavaScript development," *Communications of the ACM*, vol. 63, no. 5, pp. 42–47, 2020.
8. M. Laaziri, K. Benmoussa, S. Khoulji, and L. Kerkeb, "A comparative study of PHP frameworks performance," *Procedia Manufacturing*, vol. 32, no. 1, pp. 864–871, 2019.
9. M. Koushik and R. Selvarani, "A study on web application development using ReactJS framework," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6, pp. 1456–1460, 2019.
10. J. Verdú and A. Pajuelo, "Performance scalability analysis of JavaScript applications with Web Workers," *IEEE Computer Architecture Letters*, vol. 15, no. 2, pp. 105–108, 2016.
11. D. Yoseph and W. Raharjo, "Performance and scalability analysis of Node.js and PHP/Nginx web application," *Jurnal Informatika*, vol. 9, no. 2, pp. 379–386, 2014.
12. J. Lundar, T.-M. Grønli, and G. Ghinea, "Performance evaluation of a modern web architecture," *International Journal of Information Technology and Web Engineering*, vol. 8, no. 1, pp. 36–50, 2013.
13. M. Abdullah, "Evaluating and scaling web programming languages: A comparison between PHP and JavaScript," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 7, no. 2, pp. 169–176, 2010.
14. Y. Chen, R. Dios, A. Mili, L. Wu, and K. Wang, "An empirical study of programming language trends," IEEE Software, vol. 22, no. 3, pp. 72–78, May–Jun. 2005.