ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 3 Issue 4 December 2023 / Page No: 188-197
Paper Id: JETA-V3I8P121 / Doi: 10.56472/25832646/JETA-V3I8P121

Original Article

Modern Software Development with Java, Spring Boot, and Python: A Survey of Frameworks and Best Practices

Vandana Chaturvedi
Independent Researcher

Received Date: 30 November 2023 Revised Date: 18 December 2023 Accepted Date: 28 December 2023

Abstract: *The software development is one of the most significant innovations of the modern digital economy, enabling organizations to build scalable, reliable, and user-friendly systems across industries. With the growing complexity of technology and changes in user expectations, recent trends in software engineering have moved to agile, automated and cloud-native software engineering models that focus on quick delivery, continuous-integration and operational-resiliency. The present work is the overall survey of contemporary software development practices through perspective of Java, Spring Boot and Python that are the main technologies in the modern system design. It looks at how Java and Spring Boot can be used to support enterprise-grade, micro services based and RESTful application development whereas Python assists in web development, automation, data analytics and AI based solutions. The paper also examines architectural designs like microservices and Model-View- Controller and best practices of Agile methodologies, DevOps integration, automated testing, security implementation and cloud-native deployment plans. Through the combination of existing paradigms and developmental approaches, this article shows how the incorporation of well-developed programming languages and well-organized engineering habits improves scalability, maintainability and quality of software in the fast-changing technological environment.*

Keywords: *Software Development, Python Frameworks, Spring Boot, Java, Modern Development, Automation.*

I. INTRODUCTION

Software development is a process of coming up with, designing, writing, testing, and supporting software systems [1]. It incorporates technical skills, problem-solving skills and Using creativity to transform needs and notions into a functional software program. The modern digital world is based on the power of software development, which is used in all industries, organizations and daily routines. This is aimed at creating quality, effective and convenient software that fulfills the requirements of the targeted clients. It includes a wide spectrum of activities, such as its acquaintance with user requirements, system architecture design, writing, debugging and testing of code, and putting software into production systems. Software development can be effectively done by people working together, emphasizing best practices, and concentrating on quality assurance [2]. It is essentially a highly dynamic area that is changing dynamically due to the advancement in technology, the changed user expectations and new trends in the sector. It also comprises the scope of techniques and approaches, such as Devops, Waterfall, and Agile, and Lean that possess their own principles and practices of the way to handle the process of the software development.

In a digital age, the modern software development sector is concerned with rapid delivery, scaled delivery, resilience and continuous integration. This ecosystem is between Java, Spring boot, and Python technologies [3]. Java is still regarded as a basis of the enterprise application development due to its platform independence, robustness, and a full ecosystem. The Spring boot is an enhancement of the Java framework since it makes configuring the Java framework simple to use in microservice architecture as well as the use of the RESTful web services that can accelerate the development cycles. They provide robust platform in creation of secure and extendable enterprise systems.

Meanwhile Python has also emerged as a highly diversified language and has been described as simple and flexible. It has widespread use in web development and computer automation, artificial intelligence and data science [4]. Python also offers a capability to add analytics and machine learning capabilities to contemporary systems easily, and frameworks such as Django and Flask assist developers to develop applications within a limited duration of time [5]. It can be read and customized thus becoming more desired by start-ups and large organizations.

This paper discuss the impact of Java, Spring Boot and Python on modern software development by examining their frameworks, design patterns and tactics of their application in practice. It determines the role of enterprise-grade Java applications, Spring boot-powered microservices, and Python-powered web and data solutions in developing scalable and resilient systems. The paper also explores the architectural patterns of microservices and Model-View- Controller (MVC), and development practices of Agile approaches, DevOps teamwork, Continuous Integration / Continuous Delivery automation, security consideration, and cloud-native implementation. Combining these technologies and best practices, the study shows how

their joint implementation promotes the efficiency of development, the ability to maintain the system, the optimization of its performance, and the overall quality of software in the modern computing environment.

The paper is structured in the following way. Section II deals with the key ideas of a modern software development, including Agile, DevOps, and cloud-native. Section III is what Java and Spring boot have to do with the creation of enterprise applications. In Section IV, Python frameworks and their application in modern and AI-based systems are identified. The literature review is located in Section V, and the conclusion to the paper with the future research directions is located in Section VI.

II. MODERN SOFTWARE DEVELOPMENT

The modern software development is a development to flexible, extensible, and highly automated processes which allow rapid innovation and unrelenting delivery. Unlike the more traditional linear models of development, the current-day models rely on an iterative approach, e.g., Agile, and DevOps practices, which enable the cooperation of the development and operations team [6]. Continuous Integration (CI) and Continuous Deployment (CD) pipelines are used to guarantee quicker testing, integration and release. Moreover, the implementation of microservices and cloud-native systems allows applications to be scalable, resilient and modular. With the focus on maintainability, security, performance, and resource efficient usage, the modern development relies on the power of the programming language and frameworks to create efficient, strong and quality software systems that can meet the changing business and user demands.

A. Evolution of Programming Paradigms

Table 1 : Comparative Overview of Procedural Programming Paradigms

Category	Aspect	Description / Key Points	Significance
Definition	Procedural Programming	Based on procedure/function calls executed sequentially	Foundation of structured programming
Core Concepts	Functions / Procedures	Code blocks that are reusable and carry out specific tasks	Improves modularity and organization
	Variables (Global & Local)	Program-wide access to global variables; function-specific local variables	Enables data management but may affect data integrity
	Control Structures	Conditionals (if, else) and loops (for, while)	Controls execution flow logically
	Top-Down Design	Breaking complex problems into smaller subproblems using functions	Simplifies development process
Historical Milestones	Assembly (1940s–50s)	Low-level, hardware-specific sequential instructions	Beginning of procedural logic
	FORTRAN (1957)	First high-level language for scientific computing	Increased accessibility of programming
	COBOL (1959)	Business-oriented language with readable syntax	Improved usability in commercial applications
	ALGOL (1960)	Introduced block structures and scope rules	Influenced modern languages
	C (1972)	Combined high-level and low-level capabilities	Widely adopted for system programming
Advantages	Simplicity & Readability	Logical and structured flow	Easier debugging and understanding
	Modularity	Programs divided into procedures	Easier testing and maintenance
	Efficiency	Direct hardware interaction (e.g., C)	Suitable for performance-critical systems
Limitations	Poor Scalability	Difficult to manage large systems with global data	Not ideal for large-scale applications
	Limited Code Reusability	No inheritance or object-based reuse	Less flexible compared to OOP
	Tight Coupling	Data often globally exposed	Risk of unintended modification
	Weak Real-World Modeling	Focus on actions rather than entities	Less intuitive for complex systems

The 1960s and 1970s were the decades of computing, and the complexity of software systems could be compared to the capabilities of the procedural programming. Although effective with smaller programs, procedural paradigms had problems with global state management, no modularity, and had difficulty modelling real world entities. It became obvious that there was

a necessity to have a new way that able to model data and other behaviours associated with it in a more coherent way [7]. The object-oriented paradigm was developed to meet such demands based on the ideas of simulation and modular design. Early pioneers tried to replicate real world systems by structuring software into interacting systems, each with its own data and behavior. The programming languages form the foundation of software development as they allow the developers to transform abstract concepts into instructions that can be executed using computers. The primitive programming paradigms emphasized on simple and sequential processes. But with more complex software systems, the desire to have a more structured, scalable and reusable strategy led to procedural programming. The object-oriented paradigm later came in to solve some of the weaknesses of the procedural approach, giving a new approach to data encapsulation, abstraction and modularity. The entire overview of the procedural programming paradigms is discussed in Table 1:

B. Agile, DevOps, and CI/CD Practices

In the contemporary fast paced change in the software world where there is a demand to be met in various sectors it is not enough to create software [8], there must be a strategy. All of the process must be organized, including backend and frontend, deployment and scaling. They require a special ecosystem that speeds up the delivery, provides a high quality of software and establishes profound cross-functional teamwork. The fundamental frameworks on which this integrated approach of CI/CD thrives in Agile and DevOps should be known before getting into the workflow of the latter. Benefits and comparison are deliberating below in Table 2 and the points:

Table 2 : Comparison of Agile, CI/CD, and DevOps Practices

Key Feature	Agile	CI/CD	DevOps
Aim	Helps teams build the right product by working in small increments and incorporating continuous user feedback.	Ensures fast and safe code updates through automated integration, testing, and deployment.	Addresses challenges between development and operations teams while improving collaboration and communication.
Core Process	Iterative development in sprint cycles including backlog management, grooming, daily stand-ups, sprint reviews, and retrospectives.	Automated pipelines for code integration, testing, and deployment to ensure continuous delivery.	Implements Infrastructure as Code (IaC), continuous monitoring, and configuration management to unify development and IT operations.
Tools	Uses frameworks like Scrum or Kanban and tools such as Trello and Jira for task management and tracking.	Powered by automation tools such as Jenkins, GitHub Actions, and other pipeline orchestration platforms.	Utilizes tools like Docker for containerization and monitoring tools to manage applications and infrastructure.

- Bigger Quality of Applications: Agile pays attention to constant integration and a regular test. This assists them in identifying problems prior to their release and the result is a more stable and dependable software.
- Reduced Time-to-Market: Agile divides the work into small units instead of having a big launch that may take months [9]. It implies that the users receive new features sooner, and the teams can travel quickly.
- Flexibility and Adaptability: The use of iterative cycles of Agile facilitates comfortable dedication to the alteration in requirements, priorities, as per the market circumstance at the prevailing time. This flexibility contributes towards having a final product that is close to user and business requirements.
- Improved Risk Management: Due to the delivery progression and frequent feedback loops of Agile, the teams detect and fix problems early enough to eliminate the possibility of significant failures. When problems are identified, they are early enough before they become serious problems.
- Greater Project Transparency and Control: The constant review and the ability to see the progress, the teams are better aware of the project status and can better control the project and make smarter decisions during the process.
- Continuous Improvement: Teams make time to self-assess their output, and review what is working and make corrections where necessary. This makes the process new and ever enhancing.

C. Microservices and Cloud-Native Architectures

Microservices and cloud-native architectures are a significant improvement in the current software design [10]. A microservices architecture comprises applications in the form of a set of small and loosely coupled services that fulfil a particular business task [11]. These services are autonomous and can be communicated with lightweight mechanisms, so that they can be developed, deployed, and scaled individually by teams. Cloud-native architecture takes this strategy further by using cloud

infrastructure, containerization, and automated administration of infrastructure to develop scalable resilient systems [12]. These architectures also increase flexibility, fault tolerance and sustained delivery of distributed applications.

Key Points:

- Autonomous services are concerned with some business functions.
- Developed and implemented individually, the services.
- Enhanced scalability through service scaling independence.
- The system's increased resilience and fault isolation.
- The usage of containerisation in the same deployment environment.
- The design was intended to take advantage of dynamic resource allocation and cloud infrastructure.
- Good adherence to CI/CD and DevOps procedures.

III. JAVA AND SPRING BOOT IN ENTERPRISE APPLICATION DEVELOPMENT

Java Spring Framework has still been a powerful framework in the Java world and allows programmer to build an enterprise application which is reliable, scalable and sustainable. Rod Johnson was the one who started the concept of simplifying Java EE development, with Spring in 2003, as a way of advancing the aspects-oriented programming paradigm (Figure 1), dependency injection and the inversion of control paradigm. The framework is a fairly basic dependency injection container that has since expanded to a massive ecosystem that supports contemporary development patterns like DevOps and microservices, and reactive programming, amongst others. Starting with the very start of the Spring Framework till the current date, the paper offers a broad analysis of the current development of the framework, taking into consideration such innovations as cloud computing [13], containerization, and event-driven design.

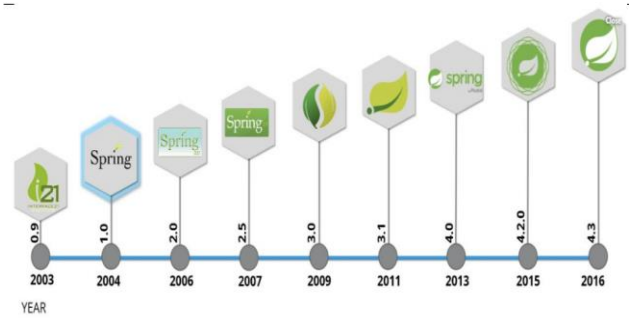

Figure 1 : Spring Framework Timeline

The enterprise systems are progressively rendering it more difficult, thus, the conditions were the structures that would improve the testability and maintainability and decrease the boilerplate code. This has been accomplished by Spring through the provision of flexibility, modularity and integration of third-party technology. The open source of Spring Cloud to distributed systems and Spring Boot to fast application development is noteworthy. In addition to rendering the development effective, these additions have rendered the framework to be in line with the existing software engineering practice. The paper critically evaluates these changes and evaluates their success in controlling the contemporary software issues and their future trends.

A. Limitations of Traditional Java Enterprise Edition Architecture

The heavy and inflexible model of architecture was the feature of enterprise development before the introduction of the Spring Framework on the Java 2 Platform Enterprise Edition (J2EE). Even though the J2EE had powerful enterprise capabilities, newer technologies such as Enterprise JavaBeans (EJB) were overly boilerplate-y and overly XML-configured [14]. This led to the tight integration of components, thus making applications more difficult to test, maintain and extend. The structural complexity of the framework tended to impose high learning curves on developers and increase development cycles.

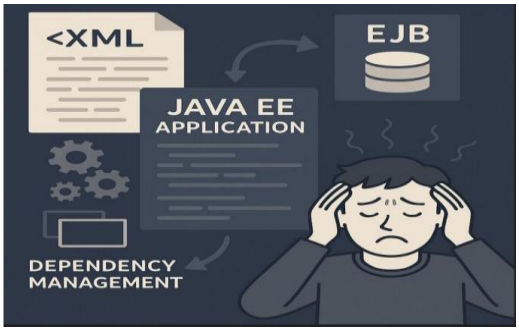

Figure 2 : Architectural Complexities of Java EE Before Spring

Flexibility and modularity were also crippled by the architecture. As the components highly relied on the container-managed service, it took a real overhaul of work before accommodating the new technology or the new business requirements. The processes of deployment were also at the same time complicated with big EAR/WAR files and environment-specific configurations that lowered the agility of operations.

Figure 2 has shown the significant limitations of the traditional Java EE development which includes voluminous XML configuration, complexity of EJB and component mis-coupling. These problems amplified maintenance work, development slowness and reduced scalability, which led to the necessity to have a lighter with more modular structure like Spring.

B. Building RESTful and Microservices-Based Applications

The construction of RESTful and microservices-based applications is one of the practices in the modern software engineering. The comparison between RESTful vs microservices architecture in Table III. The RESTful architecture provides interactions between systems by standard methods of communication like GET, POST, PUT and DELETE on the basis of standard HTTP. Data are exchanged between these services using lightweight formats such as JSON and hence integrating them across platforms is efficient and adaptable [15]. This method is further enhanced by microservices architecture, which breaks down applications into small and autonomous services, one having a mission of a certain business process. This enables standalone development, deployment and scaling of services enhancing agility, fault isolation and maintainability. Combined [16], RESTful APIs and micro services enable cloud-native applications, continuous delivery and massively scalable distributed systems.

Key Points:

- Standardized communication is achieved by using HTTP methods.
- Interactions without states enhance scalability.
- Information transfer is usually in JavaScript Object Notation.
- Application is separated into autonomous services.
- The services deal with one business capability.
- Scales and deploys independently.
- Increases fault tolerance and system resilience.
- Fits well in cloud and DevOps applications[16].

Table 3 : RESTful vs Microservices Architecture

Aspect	RESTful Applications	Microservices Architecture
Definition	Architectural style for designing networked APIs	Architectural approach for structuring applications as independent services
Communication	Uses HTTP methods like GET, POST, PUT, DELETE	Uses REST APIs or lightweight messaging between services
Structure	Resource-based endpoints	Collection of small, loosely coupled services
Scalability	Stateless design improves scalability	Each service can scale independently
Deployment	Typically, part of a larger system	Services are independently deployable
Use Case	API communication and integration	Large, complex, distributed systems

IV. PYTHON FRAMEWORKS AND ROLE IN MODERN SYSTEMS

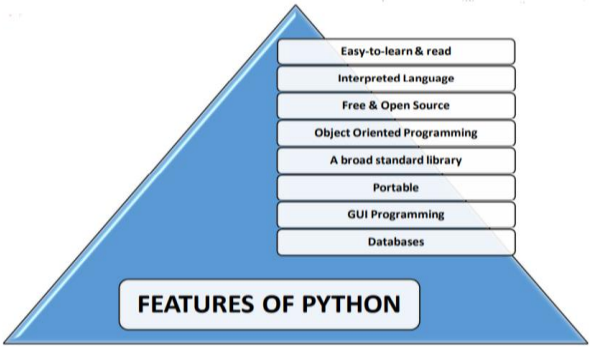

Figure 3 : Features of Python

The high-level, object-oriented, interpreted, general-purpose Python programming language was created by Guido van Rossum primarily in the late 1980s and put into use in the early 1990s. Its informative and easy to read language renders it very user friendly to the novice and at the same time powerful to its professional user developers. Python has understandable English keywords and less complicated syntactical regulations than most different programming languages, which increases clarity of code and maintainability. Python is widely applicable in the modern systems because of its automatic structure of memories and vast standard/third party libraries. Web development, cloud computing [17], automation, data analytics, AI, and ML applications extensively utilize it. Python is the language of choice when it comes to the development of scalable, data-driven, and AI-led modern software systems due to its robust ecosystem and ability to integrate smoothly. The python features are discussed above (Figure 3):

- Easy-to-learn & read: Python's minimal keyword use, simple structure, and clearly defined syntax make it incredibly easy to learn and comprehend [18]. This enables programmers to quickly pick up the language. Generally, Python's source code is maintainable.
- Interpreted Language: As with other interpreted languages such C, C++, Java, etc., Python executes code line by line at a certain moment.
- Free & Open Source: Python is a free software which is availed in official site because it is an open source.
- Object Oriented Programming: Python is an object-oriented language that uses concepts like inheritance, encapsulation, classes, and objects.
- A broad standard library: Large standard library Python has a lot of cross-platform, UNIX, Windows, and Macintosh standard libraries that are very useful.
- Portable Python: It is able to run on a very diverse range of hardware platforms and its interface is similar across all platforms.
- Windows Programming GUI applications: Python has the ability to support GUI applications that may be developed for a wide variety of system calls, libraries, and Windows systems, including Macintosh, Windows MFC, and the Unix X-Windows system.
- Databases: The interface for Python is compatible with all of the main commercial databases, including RDBMS and NoSQL databases.

A. Python for Web Development

Python is also accompanied by numerous web frameworks that are available to ease the life of a web developer. These web development frameworks are popular; some of them are Flask, Django, Pyramid, and Bottle [19]. Other additional advantages of using Python to develop a web are its Convenience, simplicity, and security in web development. Even more, Python has in-built support of other web-based protocols like HTML, XML, commonly used e-mail protocols, FTP. Python also has one of the largest libraries, which not only enhances the functionality of web applications but also makes their development simpler.

B. Python in Data-Driven and AI-Enabled Applications

Figure 4 : Common Applications of Python Programming Language

Python is central to data-driven and AI-enabled applications because it is simple, flexible, and has a large library ecosystem. Data analysis, ML, DL, and AI applications are currently popular due to the presence of robust frameworks like NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch [20]. Python can be easily used to perform data processing,

visualization, and model development, and thus it is a language of choice for creating intelligent systems, such as recommendation engines, predictive analytics platforms, and natural language processing applications. Its ability to be integrated with web systems and cloud solutions also enables AI models to be deployed as scalable services, enabling current data-driven and AI-driven software solutions.

The applications listed above (in Figure 4) are typical applications developed using the Python programming language [21]. Several domains in which the Python programming language makes it simple to develop an application, web development and many more discussed in this paper.

- Web Development: Python is associated with a large variety of different web frameworks, such as Django, Flask, Pyramid, and many others, that offer convenience to web developers.
- Machine Learning & Data Science: ML & DS are one such required subject in the present situation. To develop the ML and AI [22], SciPy, Panda, TensorFlow, Keras, etc., are some inbuilt libraries and tools provided by Python.
- Financial Applications: Python has built in libraries and applications such as Triton and Odoo, which provide a variety of for-profit apps with special security features and optimal business performance.
- Educational Development: Python is a user-friendly and the versatile program language that has taken the center stage in the classrooms across the globe. This is because of one reason: it provides the students with the starting point for learning computer science and coding. This, in its turn, is in line with the rapidly growing demands in the skills of coders within the contemporary workforce. With the constantly changing technological environment, code comprehension and manipulation has become a core competency, cutting across more than just the traditional computer science fields.

V. LITERATURE REVIEW

The literature highlights the relevance of modern frameworks and programming languages to enhance performance and scalability, as well as points out the issues with framework selection and developer adaptation. Table IV provides a comparative summary.

Choma, Chwaleba and Dzieńkowski (2023) This article compares the three most widely used server-side frameworks: Express, Spring Boot, and Django. Different programming languages are used in the development of the selected technologies. The frameworks were compared based on the time and reliability of requests. In the research conducted, three backend applications that received HTTP requests were developed and they all used the same database which was composed of the data of employees. The reliability and efficiency levels of developed apps to an array of virtual users accessing the server simultaneously were then determined through load tests. Five test scenarios were planned, with the following quantity of requests included: The server simultaneously processes 1000, 2000, 4000, 8000, and 16000 requests of each kind of HTTP request [23].

Leontevich (2023) offers an in-depth discussion of the tools and frameworks that would be instrumental in Java backend development. The entire development lifecycle is covered, along with the functions and benefits of various tool types, such as database integration tools, build automation tools, testing frameworks, version control systems, Integrated Development Environments (IDEs), containerization and orchestration tools, API documentation tools, and others. Additionally, the paper discusses the main Java backend frameworks, including Hibernate, Java Server Faces (JSF), and Spring Boot, and how they are uniquely strong and applicable in the application development [24].

Pandya and Tiwari (2023) developed two research questions (RQs) to comprehend the issues and questions posed and encountered by developers. Additionally, they identified the temporal trend of new questions posted on Stack Overflow by the programming languages (PLs) Python and Java. Their results showed that, as of 2015, Python was the most popular language after Java. In Stack Overflow, they examined 18,892 queries from Python and Java PLs, and 42,674 issues from 22 GitHub repositories, 11 for each PL. Their results indicate a correlation between Stack Overflow queries and problems developers describe during the real-time development of a corresponding PL [25].

Ollila, Mäkitalo and Mikkonen (2022) Explain the rendering techniques used by the following frameworks, which are among the most popular and prominent contemporary web frameworks: Angular, React, Vue, Svelte, and Blazor. They discover considerable variations in cost up-scaling in their rendering plans with, possibly, equally important practical performance consequences. To test these variations, they use a series of benchmarks that determine how the cost of rendering scales with an increasing complexity of an application. Their benchmark results confirm the fact that in some conditions, the difference in the framework performances can reach several orders of magnitude when they are doing the same jobs [26].

Transmissia Semiawan Muhammad Riza Alifi and Lieharyani (2021) research reveal that a framework usage does not go a long way in assisting software or application developers to develop productively and efficiently. The value between the development with and without tools do not appear to have significant differences in the success of the application that has been

measured on the level of completion, quality, and usability of the application. Moreover, according to the study, the software development skill that cannot be acquired using framework was the general or detailed system conceptualization [27].

Shetty, Dash and Joish (2020) involves the product requirements identification, design, code development and testing through frameworks and technologies. Classes and libraries that provide a rich set of functionalities are called frameworks. Applications are developed using frameworks so that the basic requirements for creating a web application are pre-established. Front-end and back-end development have a number of available frameworks that support other programming languages. It is where the web application stores the information that it uses in the back-end frameworks. Selecting the appropriate front-end framework, back-end framework, and database environment is the most significant section of a web development lifecycle [28]

Table 4 : Comparative Analysis of Related Studies Based on Modern Software Development Studies

Authors (Year)	Focus of Study	Frameworks	Evaluation Criteria	Key Findings	Contribution to Modern Development
Choma, Chwaleba & Dzieńkowski (2023)	Performance and reliability comparison of server-side frameworks	Django (Python), Spring Boot (Java), Express (Node.js)	Request processing time, scalability under load, reliability with increasing concurrent users	Performance and reliability vary across frameworks depending on request load; scalability differs significantly	Provides empirical insight for selecting backend frameworks based on performance requirements
Leontevich (2023)	Overview of Java backend development tools and frameworks	Spring Boot, JSF, Hibernate, IDEs, CI/CD tools	Development lifecycle support, integration capability, automation, maintainability	Spring Boot and related tools streamline backend development and improve maintainability	Highlights best practices and ecosystem strengths in Java backend development
Pandya & Tiwari (2023)	Trends and issues in Java and Python developer communities	Java, Python	Frequency of Stack Overflow questions, GitHub issues, temporal popularity trends	Shift in developer engagement from Java to Python since 2015; strong correlation between community questions and real-world issues	Demonstrates evolving developer preferences and real-time problem patterns
Ollila, Mäkitalo & Mikkonen (2022)	Techniques for rendering in contemporary web frameworks	Angular, React, Vue, Svelte, Blazor	Rendering cost scaling, performance benchmarking, application complexity growth	Significant differences in performance scalability; some frameworks outperform others under complex scenarios	Emphasizes importance of frontend framework choice for performance optimization
Alifi & Lieharyani (2021)	Impact of frameworks on development efficiency	General framework usage	Application quality, completion rate, usability, developer skill influence	Framework usage alone does not significantly improve development effectiveness; conceptual skills remain crucial	Suggests balanced focus between tools and developer expertise
Shetty, Dash & Joish (2020)	Role of frameworks in web development lifecycle	Front-end, back-end frameworks, databases	Framework functionality, integration with databases, lifecycle coverage	Proper selection of frontend, backend, and database combination is critical for success	Highlights importance of architectural decisions in full-stack development

VI. CONCLUSION AND FUTURE WORK

Software development has become a highly structured and technology-oriented field today, with the focus put on scalability, flexibility and continuous improvement. The combination of Java, Spring Boot, and Python shows the effectiveness

of various programming frameworks that can jointly meet the needs of the enterprise, prompt development of applications, and informative innovation. Java and Spring Boot can provide a stable and safe foundation to build the large scale and microservice-based systems, and Python can provide more flexibility in other areas, such as automation, analytics, and AI. The additional improvement of the development efficiency and the stability of its functioning are backed by the adoption of Agile approaches, DevOps, CI/CD pipelines, and cloud-native structures. Despite these being addressed, there are still some serious concerns, such as the complexity of systems, security vulnerabilities, integration overheads and skill shortages. The next work would be directed at the enhancement of cross-platform interoperability, improved automated security tools, and the usage of AI-based development tools to make the work more productive and to make decisions. Furthermore, a recent literature on sustainable software engineering, edge computing integration and intelligent monitoring system may also transform the present development practice so that it allows more resilient, efficient and adaptive software eco-systems.

VII. REFERENCES

[1] P. Nath, J. R. Mushahary, U. Roy, M. Brahma, and P. K. Singh, "AI and Blockchain-based source code vulnerability detection and prevention system for multiparty software development," Comput. Electr. Eng., vol. 106, p. 108607, Mar. 2023, doi: 10.1016/j.compeleceng.2023.108607.

[2] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," Comput. Commun., vol. 54, pp. 1–31, Dec. 2014, doi: 10.1016/j.comcom.2014.09.008.

[3] A. A. Khan et al., "Software Architecture for Quantum Computing Systems A Systematic Review," arXiv, Mar. 2023, doi: arXiv:2202.05505.

[4] A. Alam and A. Mohanty, "Discerning the Application of Virtual Laboratory in Curriculum Transaction of Software Engineering Lab Course from the Lens of Critical Pedagogy," in Sentiment Analysis and Deep Learning, 2023, pp. 53–68. doi: 10.1007/978-981-19-5443-6_5.

[5] S. Stradowski and L. Madeyski, "Exploring the challenges in software testing of the 5G system at Nokia: A survey," Inf. Softw. Technol., vol. 153, p. 107067, Jan. 2023, doi: 10.1016/j.infsof.2022.107067.

[6] A. Sheth, "Transforming Big Data into Smart Data: Deriving value via harnessing Volume, Variety, and Velocity using semantic techniques and technologies," in 2014 IEEE 30th International Conference on Data Engineering, IEEE, Mar. 2014, pp. 2–2. doi: 10.1109/ICDE.2014.6816634.

[7] M. Kumar, P. Goyal, R. Gandhi, and R. Yadav, "Evolution Of Programming Paradigms," Ind. Eng. J., vol. 51, no. 08, pp. 40–46, 2022, doi: 10.36893/IEJ.2022.V51I8.039-046.

[8] C. S. Maharao, "A Study On Impact Of Agile And Devops Practices On Software Project Management Success," ShodhKosh J. Vis. Perform. Arts, vol. 3, no. 1, Jun. 2022, doi: 10.29121/shodhkosh.v3.i1.2022.3397.

[9] R. Nirek, "Integrating Agile Methodologies with DevOps Practices in Linux Environments: A Comparative Study," Int. J. Sci. Res., vol. 7, no. 10, pp. 1824–1832, Oct. 2018, doi: 10.21275/SR24923125254.

[10] Anirudh Parupalli and Honie Kali, "An In-Depth Review of Cost Optimization Tactics in Multi-Cloud Frameworks," Int. J. Adv. Res. Sci. Commun. Technol., pp. 1043–1052, Jun. 2023, doi: 10.48175/IJARSCT-11937Q.

[11] B. M. Mweu, "The Impact of Microservices on Cloud-Native Application Development: A Review," Int. J. Nov. Res. Dev., vol. 8, no. 9, 2023.

[12] L. de A. Monteiro, W. H. C. Almeida, R. R. Hazin, C. de Anderson Lima, S. K. G. Silva, and F. S. Ferraz, "A Survey on Microservice Security–Trends in Architecture, Privacy and Standardization on Cloud Computing Environments," Int. J. Adv. Secur., vol. 11, no. 3, pp. 201–213, 2018.

[13] C. Patel, "A Review of Multi-Channel CRM Strategies Using Big Data and Cloud Integration," Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., vol. 8, no. 1, pp. 577–588, 2022.

[14] S. C. G. Varma, "The Role of Java in Modern Software Development: A Comparative Analysis with Emerging Programming Languages," Int. J. Emerg. Res. Eng. Technol., vol. 1, no. 2, pp. 28–36, 2020, doi: 10.63282/3050-922X/IJERET-V1I2P104.

[15] S. Knox, P. Meier, J. Yoon, and J. J. Harou, "A python framework for multi-agent simulation of networked resource systems," Environ. Model. Softw., vol. 103, pp. 16–28, May 2018, doi: 10.1016/j.envsoft.2018.01.019.

[16] S. Garg, "Predictive Analytics and Auto Remediation using Artificial Inteligence and Machine learning in Cloud Computing Operations," Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci., vol. 7, no. 2, 2019, doi: 10.5281/zenodo.15362327.

[17] V. Shah, "Analyzing Traffic Behavior in IoT-Cloud Systems : A Review of Analytical Frameworks," Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., vol. 9, no. 3, pp. 877–885, 2023.

[18] V. Cutting and N. Stephen, "A Review on using Python as a Preferred Programming Language for Beginners," Int. Res. J. Eng. Technol., vol. 8, no. 8, 2021, doi: 10.1109/ICCNEA57056.2022.00030.

[19] R. Chouhan, K. Singh, and R. Vashistha, "Review Paper on Python for Data Science & Web Development," World J. Res. Rev., vol. 14, no. 5, pp. 15–19, 2022.

[20] S. Raschka, J. Patterson, and C. Nolet, "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence," Information, vol. 11, no. 4, p. 193, Apr. 2020, doi: 10.3390/info11040193.

[21] M. Otieno, D. Odera, and J. E. Ounza, "Theory and practice in secure software development lifecycle: A comprehensive survey," World J. Adv. Res. Rev., vol. 18, no. 3, pp. 053–078, Jun. 2023, doi: 10.30574/wjarr.2023.18.3.0944.

[22] P. Siva, D. Yamaganti, D. Rohita, and U. sikharam, "A Review on Python for Data Science, Machine Learning and IOT," 2023. doi: 10.13140/RG.2.2.18708.48000.

[23] D. Choma, K. Chwaleba, and M. Dzieńkowski, “The Efficiency And Reliability Of Backend Technologies: Express, Django, And Spring Boot,” Inform. Autom. Pomiary w Gospod. i Ochr. Środowiska, vol. 13, no. 4, pp. 73–78, Dec. 2023, doi: 10.35784/iapgos.4279.
[24] Z. A. Leontevich, “Tools for Effective Java Backend Development,” Int. J. Latest Eng. Manag. Res., vol. 8, no. 8, pp. 50–60, 2023.
[25] N. Pandya and S. Tiwari, “Similarities in Challenges faced by Developers: Investigations on Stack Overflow and GitHub,” in 16th Innovations in Software Engineering Conference, 2023, pp. 1–11. doi: 10.1145/3578527.3578539.
[26] R. Ollila, N. Mäkitalo, and T. Mikkonen, “Modern Web Frameworks: A Comparison of Rendering Performance,” J. Web Eng., vol. 21, no. 3, pp. 789–814, Mar. 2022, doi: 10.13052/jwe1540-9589.21311.
[27] T. Semiawan, M. R. Alifi, H. Hayati, and D. C. U. Lieharyani, “Analysis of the Effectiveness and Efficiency of Software Development Tools,” in Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021), 2021, pp. 32–39. doi: 10.2991/aer.k.211106.006.
[28] J. Shetty, D. Dash, and A. K. Joish, “Review Paper on Web Frameworks, Databases and Web Stacks,” Int. Res. J. Eng. Technol., vol. 07, no. 04, pp. 5734–5738, 2020.