iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** XI **Month of publication:** November 2023

DOI: https://doi.org/10.22214/ijraset.2023.56617

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XI Nov 2023- Available at www.ijraset.com

Efficient Backend Development with Spring Boot: A Comprehensive Overview

Sushmita C. Hubli[1], Dr. R.C. Jaiswal[2]
[1]*Department of Electronics and Telecommunication, Pune Institute of Computer Technology, Pune, India*
[2]*Associate Professor, Department of Electronics and Telecommunication, Pune Institute of Computer Technology, Pune, India*

Abstract: ***This journal paper provides a comprehensive exploration of Spring Boot, a highly sought-after framework in contemporary web application development. It begins by introducing Spring Boot's core concepts, architectural principles, and its capacity to reduce redundant coding efforts. The paper delves into various facets of backend development, focusing on data storage through Spring Data JPA, as well as its adaptability to relational databases. Spring Boot's proficiency in simplifying data source integration and providing efficient data access tools is emphasized. Serving as an invaluable resource, this paper offers developers insights into Spring Boot's pivotal features, benefits, and best practices for crafting scalable, maintainable, and robust backend systems in the ever-evolving landscape of web applications.***
Keywords: Spring Boot, backend development, Spring Data JPA, database

I. INTRODUCTION

In today's rapidly evolving digital landscape, the efficiency and reliability of a web application heavily rely on its backend infrastructure. Backend development has undergone a significant transformation, with the emergence of powerful frameworks designed to streamline the development process and optimize performance. Spring Boot, in particular, has gained widespread recognition for its ability to simplify the creation of robust, enterprise-grade backend applications [2]. This paper embarks on an exploratory journey into efficient backend development with Spring Boot, offering a comprehensive understanding of its core principles, features, and best practices [8]. Whether you're an experienced developer looking to enhance your skills or a newcomer eager to grasp backend intricacies, this overview serves as a valuable resource to equip you with the knowledge and tools needed to craft high-performing and maintainable backend systems. We navigate through Spring Boot project architecture, data persistence, security, microservices, code structuring, testing, and CI/CD practices, culminating with insights into performance optimization and monitoring, ensuring you are well-prepared to tackle the challenges of modern backend development [6].

A. *Objectives*

1) To provide a comprehensive understanding of Spring Boot as a framework for backend development, including its core principles, architecture, and key features [2].
2) To explore the advantages of using Spring Boot in backend development, emphasizing its ability to reduce development effort, minimize boilerplate code, and enhance developer productivity [2][8].
3) To elucidate the process of setting up a Spring Boot project, including essential components, configurations, and best practices, enabling developers to initiate projects with ease [2].
4) To delve into data persistence strategies with Spring Boot, covering the integration of Spring Data JPA, relational databases, and NoSQL options, and showcasing how Spring Boot simplifies data access [2][6][7].
5) To address security considerations within Spring Boot applications, with a focus on authentication and authorization mechanisms, highlighting the integration of Spring Security to protect RESTful APIs and web applications [5].

B. *Scope*

1) *Framework Exploration:* This journal paper will delve into the theoretical underpinnings of Spring Boot, examining its core principles, architectural components, and key features within a scholarly context [2][8].
2) *Data Persistence Strategies:* The scope extends to academic insights into data persistence strategies, incorporating principles of data modeling and database design, grounded in academic theory [6].
3) *Security Framework:* This paper will academically explore security considerations, including the theoretical aspects of authentication, authorization, and the integration of Spring Security, informed by academic models and security paradigms [5].

4) *Microservices and Best Practices:* The scope will encompass an academic discussion of Spring Boot's alignment with microservices architecture, along with best practices and methodologies rooted in academic software engineering principles [4].

II. LITERATURE SURVEY

The dynamic landscape of technology and software development is underpinned by a multifaceted array of research endeavours, each addressing unique challenges and opportunities [2]. The onset of the COVID-19 pandemic redefined global economic dynamics, particularly affecting the trade sector and Small and Medium Enterprises (SMEs). In response to these unprecedented challenges, enterprises like Hanura Takeaway (Haway) emerged, specializing in goods and food delivery. To simplify transaction processes in this evolving landscape, web services and RESTful API web services became vital, enabling efficient data exchange through standard internet protocols [1]. The integration of RESTful APIs, while relatively underexplored in existing literature, has proven instrumental in software development, facilitating seamless interactions across different programming languages and platforms, with advanced features like URI optimization through parameters.

In the realm of backend development, the role of databases is paramount. Databases form the foundational infrastructure of software applications, storing and managing data critical to application functionality [2]. The choice of a database system can significantly impact the performance, scalability, and overall reliability of a backend system [4]. Furthermore, efficient database design plays a pivotal role in shaping the performance of the entire application. The paper on MySQL's optimization underscores the importance of a well-designed database system to ensure that data access does not compromise server performance [6]. MySQL, as a trusted and widely used open-source database platform, exemplifies the significance of a robust database system in backend development.

Meanwhile, Spring Security, celebrated for its ease of use in securing enterprise applications, takes center stage in addressing application framework misconfiguration vulnerabilities, identifying security anti-patterns and insecure defaults. The security of backend systems is of paramount importance, especially when handling sensitive data. Spring Security provides a comprehensive framework for implementing security features in Spring-based applications [5]. It offers authentication and authorization mechanisms, protecting the application from various security threats, including unauthorized access, data breaches, and more. The paper's exploration of Spring Security underscores its significance in safeguarding backend systems and the data they manage, thereby ensuring the integrity and confidentiality of data in modern applications.

As governments increasingly turn to information and communication technologies (ICT) for governance, the concept of E-government gains prominence. The development of public complaint service applications based on Spring Boot's microservices architecture exemplifies the modernization of public service delivery, enhancing accessibility and transparency [4]. Spring Boot's role in creating a Service-Oriented Architecture (SOA)-based REST service API at the Atmospheric Radiation Measurement (ARM) Data Center showcases its utility in bridging the divide between frontend user interfaces and backend databases [2]. In this context, databases play a critical role in storing and managing data critical to public service operations. Spring Boot's microservices architecture enhances the scalability and maintainability of these applications, allowing for the seamless addition of microservices to meet evolving service requirements [4]. Also, the role of advanced tools required using ML and ESPs [11-24] are becoming important in recent applications, recognition, and control.

These diverse research strands, encompassing database systems, security frameworks like Spring Security, and backend development paradigms, culminate in an enriched understanding of technology, software development, and their multifaceted impact on our digital world. They emphasize the symbiotic relationship between robust databases, secure frameworks, and efficient backend development, collectively contributing to the reliability and functionality of modern software applications.

III.METHODOLOGIES

The objective of this paper is to design and assess the backend component of a web application, and this will be achieved through the utilization of specific methodologies. The primary methodology entails the utilization of Spring Boot, a Java-based open source microservices framework that empowers the independent creation and deployment of services, each with its dedicated process, thereby fostering a lightweight business application support paradigm [2]. Complementing this, the Spring Web MVC framework, also Java-based, offers a Model-View-Controller (MVC) architecture known for its flexibility and loose coupling, effectively isolating the input logic, business logic, and user interface logic within the application. The Model layer encapsulates web application data, typically in the form of Plain Old Java Objects (POJO), while the View layer is responsible for presenting the model data and generating client-interpretable HTML output. The Controller layer manages user requests, generating a model subsequently rendered by the View layer.

In conjunction with these, Spring Data JPA, a component of the broader Spring Data family, simplifies the development of JPA-based repositories and the data access layer, streamlining tasks such as simple searches, pagination, and audits [2]. It reduces the need for extensive boilerplate code by allowing developers to define repository interfaces and custom locator methods while Spring handles the implementation. The entire development process is facilitated by the Spring Tool Suite IDEA, an integrated development environment tailored for creating Spring Boot microservices, offering a pre-configured workspace and extensibility through a plugin framework.

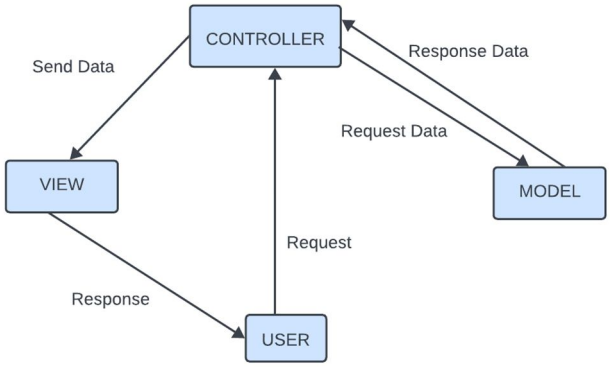


Fig.1: Pre-configured workspace and extensibility through a plugin framework

1) *Model:* The Model in the Model-View-Controller (MVC) architecture represents the application's data and logic. It encapsulates information, processes, and business rules, typically in Plain Old Java Objects (POJOs) [2].
2) *View:* The View is responsible for the presentation of data to users. It generates the HTML or user interface that a web browser can display. It doesn't contain application logic but focuses on rendering data in a user-friendly format [2].
3) *Controller:* The Controller acts as an intermediary between the Model and the View. It handles user requests, processes input, interacts with the Model to retrieve or modify data, and then sends the appropriate data to the View for presentation. It ensures that the Model and View remain loosely coupled, allowing for modular development and maintainability of web applications [2].

IV. IMPLEMENTATION

In this paper, the focus lies on the development of a backend system using the Spring Boot framework with complete configuration of the JPA persistence layer.

A. *Key Steps to Implement a Spring Boot project*

1) Set up your development environment (JDK, IDE, build tool).
2) Create a Spring Boot project using Spring Initializer or your IDE.
3) Define project dependencies and structure.
4) Write application code, including controllers, services, and repositories.
5) Configure application properties.
6) Write unit tests and use testing frameworks.
7) Build and package your application.
8) Test it locally.
9) Configure database and security (if needed).
10) Implement logging, error handling, and API documentation.
11) Test, debug, and prepare for production.
12) Deploy your application.
13) Monitor and scale (if necessary).
14) Maintain and update your project.
15) Document your application.
16) Consider setting up CI/CD for automation.

B. Spring Boot Application Architecture

In a Spring Boot application, the components like Postman, Controller, Service, DAO (Data Access Object), and the database play essential roles in achieving a well-structured and functional backend.

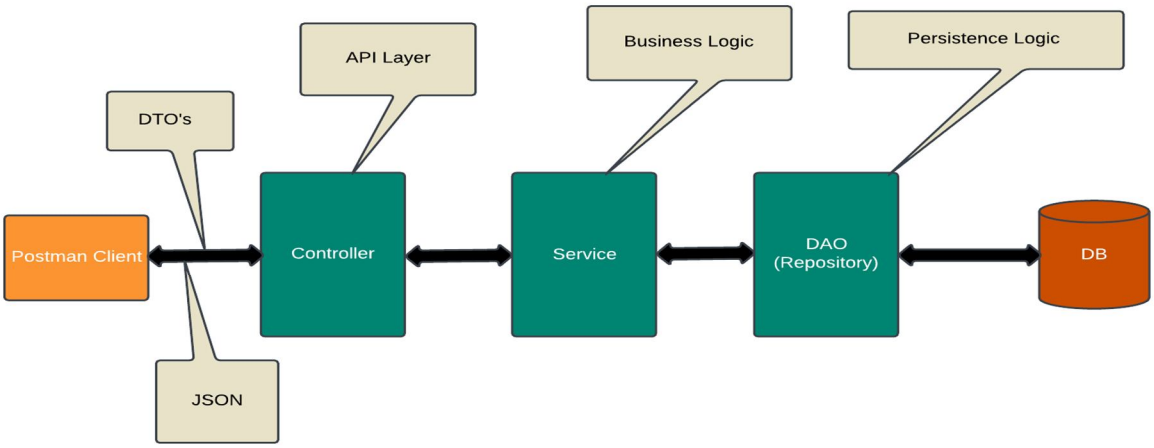


Fig. 2: Spring Boot Application Architecture

C. Postman

Postman is not an integral part of the Spring Boot application but serves as a valuable external tool for testing and interacting with the application's RESTful API. Developers use Postman to send HTTP requests to the application's API endpoints, making it easier to validate the API's functionality. Postman allows users to test various HTTP methods (GET, POST, PUT, DELETE) and inspect the responses, making it an indispensable tool for API development and debugging [1].

D. Controller

The Controller layer in a Spring Boot application is responsible for handling incoming HTTP requests and orchestrating the flow of data and actions within the application. Controllers are annotated with Spring's @Controller or @RestController annotations, which allow them to receive requests and return responses. They define request mappings, specifying which methods should be executed for specific URI endpoints and HTTP methods. Controllers interact with the Service layer to retrieve data, process requests, and return results to the client.

E. Service

The Service layer is an intermediate layer between the Controller and the DAO (Data Access Object) layer. It contains business logic, application-specific rules, and service operations. Services are annotated with @Service to indicate that they are Spring-managed components. They help maintain a clear separation of concerns, ensuring that business logic is not tightly coupled with the web layer. The Service layer typically involves aggregating and processing data before passing it to the DAO for database operations.

F. DAO (Data Access Object)

The DAO layer is responsible for database interactions, including data retrieval, storage, and manipulation. DAOs are annotated with @Repository to indicate that they are Spring-managed components responsible for database operations. They encapsulate data access logic, abstracting away the complexities of database connections and queries. The DAO layer uses Spring Data JPA, Hibernate, or other data access frameworks to interact with the database. By separating database-related code into the DAO layer, it becomes easier to switch between different data sources or database technologies.

G. Database (DB)

The database is where the application stores and retrieves data. In a Spring Boot application, this can be a relational database (e.g., MySQL, PostgreSQL) or a NoSQL database (e.g., MongoDB, Cassandra), depending on the application's requirements. Spring Boot simplifies database interactions with features like auto-configuration, data source management, and support for various database technologies. The database stores the application's data, and the DAO layer is responsible for translating high-level data operations into database-specific queries. In summary, the components in a Spring Boot application work together to create a robust and well-structured backend. Postman aids in testing and validating the API, the Controller handles HTTP requests and responses, the Service layer contains business logic, the DAO layer manages database interactions, and the database stores and retrieves data. This layered architecture ensures a separation of concerns, making the application more maintainable, testable, and scalable [6][7][9].

The data storage is facilitated by H2, a relational database, and the implementation is primarily in Java. The process begins with project creation in start.spring.io and the addition of necessary dependencies such as JPA, H2, and JDBC [1][2]. The configuration of Spring JPA is handled in the application.properties file. A DataModel class is created under the Entity model, marked with the @Entity annotation, signifying its entity status [1]. For example, a class like "Employee" with attributes like first name, last name and emailId results in the generation of a corresponding "Employee" table with id, first_name, last_name, and emailId columns. Additional annotations, such as @TableName, @Id, @GeneratedValue, and @Column, are used to specify mapping and primary key details. An interface for the repository package allows CRUD operations on the DataModel class, extending CrudRepository and supporting repository and custom finder functions [2].

In the Controller layer, REST API call types and endpoints are specified using annotations like @GetMapping, @PostMapping, @PutMapping, and @DeleteMapping. These methods handle client requests, data retrieval from the repository, and result returns, with common methods like save, findAll, findById, and findByFirstName. The project is executed, and testing involves sending HTTP requests with specified methods, URIs, and request bodies, receiving JSON responses with accompanying HTTP status codes. The four primary HTTP methods used are GET (read-only resource access), POST (new resource creation), DELETE (resource deletion), and PUT (resource update) [10].

V. RESULTS AND DISCUSSION

It's important to understand that receiving server responses in JSON format is like speaking a common language between computers. It makes data exchange smooth and easy, which is great for web applications. JSON is like a simple and organized way to share information. However, just like in any language, you need to be careful with security. JSON can sometimes be vulnerable to data-related problems, so it's crucial to have strong security measures to protect against potential issues. In summary, using JSON for server responses is like having a universal language for computers, making data exchange efficient, but safety precautions are vital to keep everything secure [8].

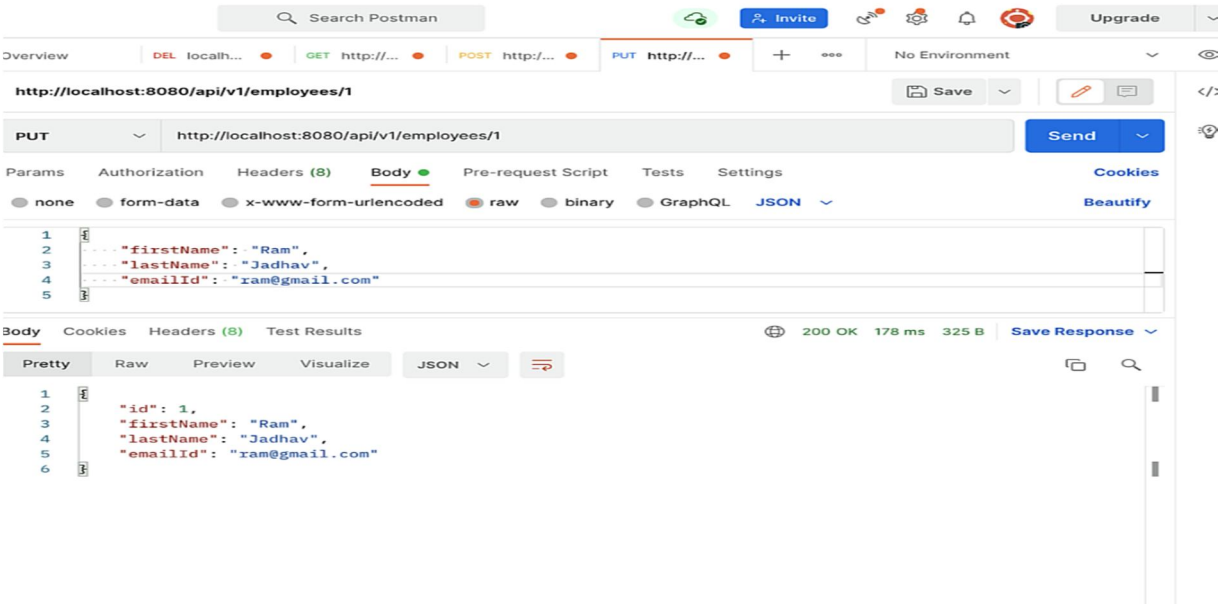


Fig. 3: Testing and validation process

VI. CONCLUSION

In conclusion, our exploration of backend development using Spring Boot reveals a powerful framework that simplifies application creation while promoting efficiency and best practices. Spring Boot's convention-over-configuration approach minimizes complexity, allowing developers to focus on business logic. Its adaptability suits diverse project requirements, from monolithic systems to microservices. Robust security integration and standardized practices ensure application safety.

VII. ACKNOWLEDGMENT

I would like to express my sincere gratitude to Dr. R.C. Jaiswal sir for affording me the invaluable opportunity to work on this journal paper under his expert guidance and mentorship. His unwavering support, insightful feedback, and wealth of knowledge have been instrumental in shaping this research endeavor. I am deeply appreciative of the trust he placed in me and the opportunity to learn and grow under his tutelage. This experience has not only enriched my academic journey but has also instilled in me a profound sense of inspiration and purpose.

REFERENCES

[1] Ahmad, E. Suwarni, R. I. Borman, Asmawati, F. Rossi and Y. Jusman, "Implementation of RESTful API Web Services Architecture in Takeaway Application Development," 2021 1st International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS), Yogyakarta, Indonesia, 2021, pp. 132-137, doi: 10.1109/ICE3IS54102.2021.9649679.

[2] K. Guntupally, R. Devarakonda and K. Kehoe, "Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 5328-5329, doi: 10.1109/BigData.2018.8621924.

[3] M. C. Calzarossa and L. Massari, "Analysis of Header Usage Patterns of HTTP Request Messages," 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS), Paris, France, 2014, pp. 847-853, doi: 10.1109/HPCC.2014.146.

[4] Suryotrisongko, Hatma & Jayanto, Dedy & Tjahyanto, Aris. (2017). Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot. Procedia Computer Science. 124. 736-743. 10.1016/j.procs.2017.12.212.

[5] M. Islam, S. Rahaman, N. Meng, B. Hassanshahi, P. Krishnan and D. D. Yao, "Coding Practices and Recommendations of Spring Security for Enterprise Applications," 2020 IEEE Secure Development (SecDev), Atlanta, GA, USA, 2020, pp. 49-57, doi: 10.1109/SecDev45635.2020.00024.

[6] Kisman and S. M. Isa, "Hibernate ORM query simplication using hibernate criteria extension (HCE)," 2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), Danang, Vietnam, 2016, pp. 23-28, doi: 10.1109/NICS.2016.7725656.

[7] K. I. Satoto, R. R. Isnanto, R. Kridalukmana and K. T. Martono, "Optimizing MySQL database system on information systems research, publications and community service," 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, Indonesia, 2016, pp. 1-5, doi: 10.1109/ICITACEE.2016.7892476.

[8] M. Mythily, A. Samson Arun Raj and I. Thanakumar Joseph, "An Analysis of the Significance of Spring Boot in The Market," 2022 International Conference on Inventive Computation Technologies (ICICT), Nepal, 2022, pp. 1277-1281, doi: 10.1109/ICICT54344.2022.9850910.

[9] Ying Bai; Satish Bhalla, "Introduction to Databases," in SQL Server Database Programming with Visual Basic.NET: Concepts, Designs and Implementations , IEEE, 2020, pp.9-65, doi: 10.1002/9781119608493.ch2.

[10] J. Yang, C. Yan, C. Wan, S. Lu and A. Cheung, "View-Centric Performance Optimization for Database-Backed Web Applications," 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), Montreal, QC, Canada, 2019, pp. 994-1004, doi: 10.1109/ICSE.2019.00104.

[11] Shreyas Purkar, Jaiswal R.C., M.V. Munot "The Role of Management Information Systems in Organizations", International Journal "Gradiva Review Journal" (GRJ), UGC Care group-II journal, Open Access, Peer Reviewed, refereed and multidisciplinary Journal, Google Scholar, Scribd, ResearchGate, Scopus indexed, ISSN: 0363-8057; SJR Impact Factor:0.101, Volume 9, Issue IX, pp. 506-516, September 2023.,

[12] Jaiswal R.C. and Shahul Patil, "Small Businesses Need Project Management", International Journal for Research in Applied Science & Engineering Technology (IJRASET), Open Access, Peer Reviewed and refereed Journal, ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:7.538, Volume 10, Issue XII, pp. 1532-1536, December 2022.

[13] Aarya Harkare, R. C. Jaiswal, "Object Fetching UAV using Autonomous Flight and Object Detection Algorithms", International Journal for Research in Applied Science & Engineering Technology (IJRASET), Open Access, Peer Reviewed and refereed Journal, Google Scholar, Mendeley : reference manager, Cite-Factor, Index Copernicus, ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:7.538, Volume 11, Issue IX, pp. 602-610, September 2023.

[14] Jaiswal R.C. and Shivani Pande, " Microservices in Cloud Native Development of Application", International Journal of Creative Research Thoughts (IJCRT), Open Access, Peer Reviewed and refereed Journal, Indexed in Google Scholar, Microsoft Academic, CiteSeerX, Thomson Reuters, Mendeley : reference manager, ISSN: 2320-2882; SJ Impact Factor:7.97, Volume 10 Issue X, pp. d170-d183, October 2022.

[15] Jaiswal R. C. and Atharva Agashe, " A Survey Paper on Cloud Computing and Migration to the Cloud", Journal of Emerging Technologies and Innovative Research (JETIR), Open Access, Peer Reviewed and refereed Journal, Indexed in Google Scholar, Microsoft Academic, CiteSeerX, Thomson Reuters, Mendeley : reference manager, ISSN-2349-5162, Impact Factor:7.95, Volume 9, Issue 10 pp. a258-a265, October 2022.

[16] Jaiswal R. C. and Apoorva Ushire, " Real Time Water Monitoring System Using NodeMCU ESMP8266 ", Journal of Emerging Technologies and Innovative Research (JETIR), Open Access, Peer Reviewed and refereed Journal, Indexed in Google Scholar, Microsoft Academic, CiteSeerX, Thomson Reuters, Mendeley : reference manager, ISSN-2349-5162, Impact Factor:7.95, Volume 9, Issue 9 pp. c1-c8, September 2022.

[17] Jaiswal R. C. and Akshat Kaushik, "Automated Attendance Monitoring system using discriminative Local Binary Histograms and PostgreSQL", Journal of Emerging Technologies and Innovative Research (JETIR), Open Access, Peer Reviewed and refereed Journal, ISSN-2349-5162, Impact Factor:5.87, Volume 7, Issue 11, pp. 80-86, November 2020.

[18] Jaiswal R.C. and Deepali Kasture, "Pillars of Object-Oriented System", International Journal for Research in Applied Science & Engineering Technology (IJRASET), Open Access, Peer Reviewed and refereed Journal , ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:7.177, Volume 7 Issue XI, pp. 589-591, Nov 2019.

[19] Jaiswal R.C. and Sakshi Jain, "Text Search Engine",', Journal of Emerging Technologies and Innovative Research (JETIR), UGC approved Journal ISSN-2349-5162, Volume 5, Issue 11, November 2018.

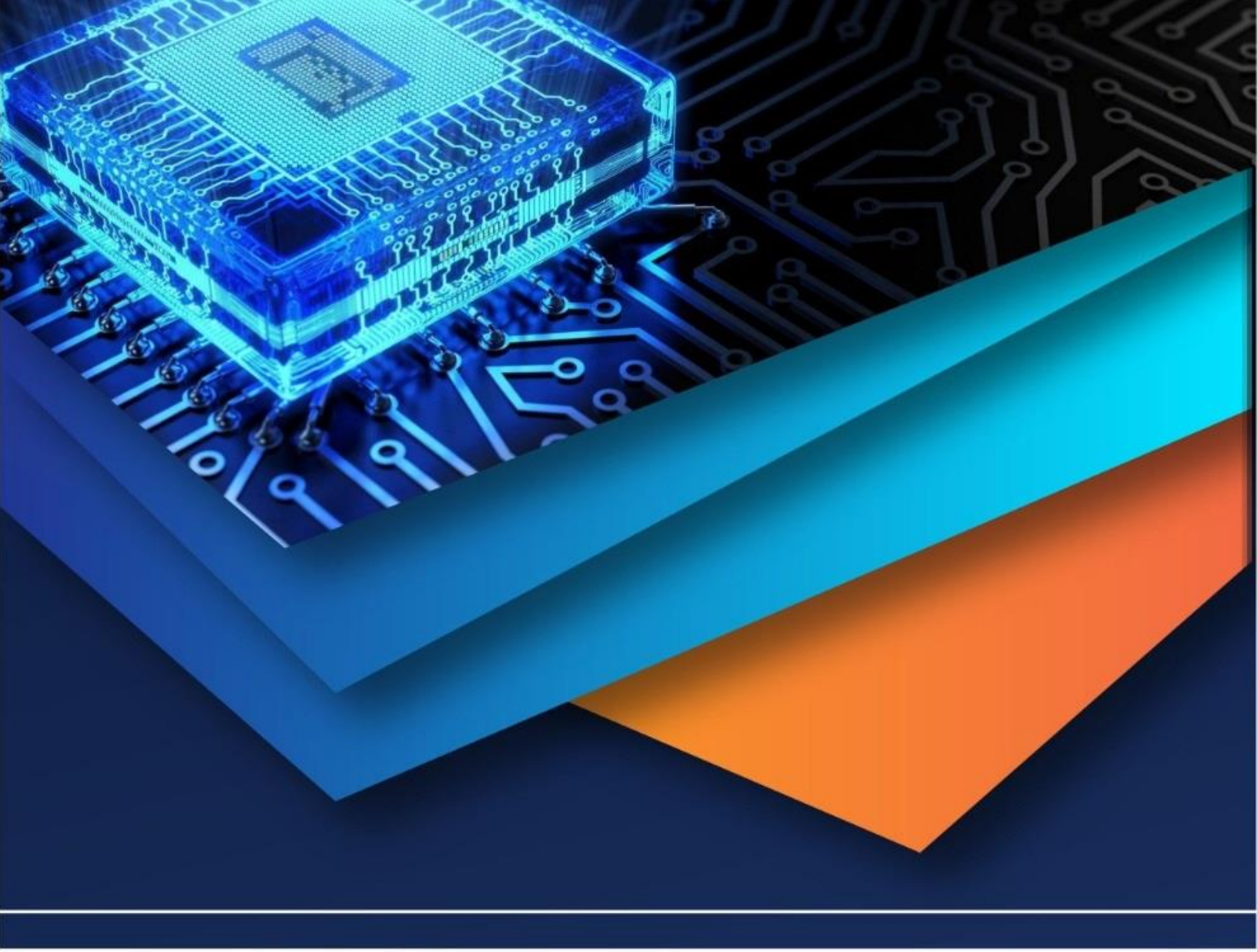
[20] Jaiswal R.C. and Saloni Takawale "Multi-Client Server Communication Enhancementthrough Intranet", International Journal for Research in Applied Science & Engineering Technology (IJRASET), ISSN: 2321-9653; UGC approved Journal, IC Value: 45.98; SJ Impact Factor :6.887, Volume 6 Issue 1, January 2018.

[21] Jaiswal R.C. and Swapnil Shah, "Customer Decision Support System", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056; p-ISSN: 2395-0072; UGC approved Journal,SJ Impact Factor:5.181, Volume: 04 Issue: 10 | Oct -2017.

[22] S. Shukla, "Debugging Microservices With Python," IIOAB Journal, vol. 10, pp. 32–37, Feb. 2019.

[23] Sameer Shukla, "Debugging Microservices with Pandas, PySpark using Actuators and Logs at Runtime," International Journal of Computer Sciences and Engineering, Vol.10, Issue.7, pp.27-30, 2022.

[24] Shreyas Purkar, Jaiswal R. C., "Business Information Security", Journal of Emerging Technologies and Innovative Research (JETIR), Open Access, Peer Reviewed and refereed Journal, Indexed in Google Scholar, Microsoft Academic, CiteSeerX, Thomson Reuters, Mendeley : reference manager, ISSN-2349-5162, Impact Factor:7.95, Volume 10, Issue 10 pp. d206-d213, October 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY