WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Conversational AI with Springboot, OpenAPI & Ollama A Smarter Chatbot Solution

Dr. RVVSV Prasad[1] *Professor*, ***Balla Veeresh[2]***, ***Kothagundu Harish[3]***, ***Nukala Nagaraju[4]***

ramayanam.prasad@gmail.com[1], veereshballa56@gmail.com[2], kothagunduharish@gmail.com[3], nagarajunukala777@gmail.com[4]
Department of Information Technology
Swarnandhra College of Engineering and Technology(A), Seetharampuram, Narsapur, AP 534280.

ABSTRACT:

In the era of intelligent automation, conversational agents play a pivotal role in enhancing user engagement across diverse domains. This paper presents a smart and scalable chatbot solution developed using Spring Boot, OpenAPI, and Ollama. Spring Boot facilitates rapid backend development and supports a microservices architecture. OpenAPI ensures seamless API documentation and system integration. Ollama, an advanced AI framework, enables robust natural language processing (NLP) and machine learning (ML) capabilities, providing accurate, context-aware responses. The chatbot supports real-time, personalized conversations and integrates external services for enhanced functionality. With a modular design, this solution is adaptable to applications such as customer support, e-commerce, healthcare, and knowledge management. The combination of Spring Boot's microservices infrastructure, OpenAPI's standardization, and Ollama's conversational intelligence illustrates a comprehensive approach to developing intelligent and extensible chatbot systems.

Keywords: Conversational AI, Chatbot, Spring Boot, OpenAPI, Ollama, NLP, Machine Learning, Microservices, API Integration, Automation.

1.Introduction

The rapid advancement of artificial intelligence (AI) and natural language processing (NLP) technologies has significantly transformed human-computer interaction. Among the most impactful innovations are conversational agents, commonly known as chatbots, which have been widely adopted across industries for tasks such as customer service, virtual assistance, and process automation [1], [2]. These systems emulate human-like dialogue by interpreting user queries and generating appropriate, context-aware responses. Modern chatbot architectures are evolving to meet the demand for scalability, adaptability, and integration with existing IT infrastructures. Microservices-based frameworks like Spring Boot have emerged as popular solutions for building modular, maintainable, and scalable backend systems [3]. With its ability to rapidly bootstrap applications and support containerization, Spring Boot offers a solid foundation for developing conversational AI systems. To ensure interoperability and effective communication between services, OpenAPI provides a standardized mechanism for describing and documenting RESTful APIs [4]. This specification enables seamless integration with third-party services, reduces development complexity, and enhances maintainability. The intelligence layer of the chatbot is driven by Ollama, a cutting-edge framework designed for deploying AI models with a focus on natural language understanding and machine learning. Ollama simplifies the interaction with large language models (LLMs), providing capabilities such as intent recognition, sentiment analysis, and dialogue management in a developer-friendly environment [5]. This project proposes a smart, real-time chatbot solution that integrates Spring Boot, OpenAPI, and Ollama to deliver personalized user experiences. The system is designed with extensibility in mind and can be applied across various domains such as healthcare, e-commerce, customer support, and education. By combining robust backend infrastructure with state-of-the-art AI capabilities, the proposed solution addresses the challenges of modern conversational systems, including context retention, scalability, and intelligent response generation. The rest of the paper is structured as follows: Section II discusses related work in conversational AI; Section III describes the system architecture and implementation; Section IV presents results and evaluation metrics; and Section V concludes the paper and outlines future work.

2.Literature Review

The development of conversational AI systems has been the focus of extensive research, encompassing natural language processing (NLP), machine learning (ML), and software architecture. This section reviews relevant studies in chatbot frameworks, NLP advancements, microservices architecture, and API standardization.

2.1. Evolution of Chatbot Technologies

Early chatbot systems such as ELIZA and ALICE demonstrated rule-based interaction models with limited flexibility [6]. With the emergence of machine learning and deep learning, chatbots have shifted towards more data-driven approaches. Sequence-to-sequence models, transformer architectures, and

large language models (LLMs) have enhanced the capability of chatbots to understand context and generate coherent responses [7]. These advancements have led to their widespread adoption in sectors like customer service, education, and healthcare [8].

2.2. Natural Language Processing and Machine Learning

Modern NLP has significantly evolved with the introduction of pre-trained models like BERT, GPT, and T5, enabling better semantic understanding and contextual conversation [9]. Techniques such as intent classification, named entity recognition, and sentiment analysis are central to intelligent dialogue systems [10]. Integration of these NLP methods allows conversational agents to personalize responses, understand user goals, and maintain multi-turn dialogue effectively.

2.3. Spring Boot and Microservices for Chatbot Deployment

Microservices architecture facilitates scalable and modular software development, making it an ideal choice for deploying conversational AI systems. Spring Boot, in particular, provides auto-configuration, embedded servers, and simplified dependency management, enabling fast development cycles [11]. Research by Sharma et al. [12] demonstrated that microservices-based chatbot platforms improved maintainability and supported distributed deployments efficiently.

2.4. OpenAPI for API Standardization

APIs are crucial for enabling chatbot systems to communicate with external services such as payment gateways, knowledge bases, or CRM systems. The OpenAPI Specification standardizes API contracts, enabling easier documentation, testing, and interoperability [13]. Studies show that using OpenAPI reduces integration errors and speeds up development time by providing a machine-readable interface [14].

2.5. Ollama and Local LLM Integration

Ollama represents a new paradigm in conversational AI by allowing developers to run LLMs locally with minimal infrastructure [15]. Unlike cloud-based services that depend on internet connectivity and raise privacy concerns, Ollama enables secure, offline access to intelligent models. Initial experiments have shown that local LLMs using Ollama maintain reasonable performance while offering greater flexibility and control [16].

3.Proposed system

The proposed system introduces an intelligent and scalable chatbot solution by integrating Spring Boot, OpenAPI, and Ollama to enable real-time, context-aware conversations. The backend of the system is developed using Spring Boot, which supports a microservices architecture, enabling the chatbot to be modular, easily maintainable, and scalable. Spring Boot's auto-configuration and dependency management simplify application development and deployment, making it suitable for cloud-native environments [17]. For standardized and interactive API documentation, OpenAPI is employed. It allows for the creation of machine-readable specifications that can be easily tested and consumed by both humans and other services. By following the OpenAPI standard, the system ensures reliable API integration, reduces compatibility issues, and facilitates third-party service extensions [18]. This integration is essential in enabling the chatbot to interact with external services such as databases, customer support platforms, and transactional systems. The intelligence layer of the system is powered by Ollama, which provides an interface for running large language models (LLMs) locally. This offers significant advantages in terms of data privacy, latency, and offline availability when compared to traditional cloud-based models [19]. The chatbot utilizes Ollama to process user input, perform natural language understanding, and generate coherent, contextually relevant responses using pretrained LLMs. Ollama's lightweight infrastructure enables developers to experiment and deploy conversational models without complex setups or GPU dependencies [20]. The architecture is designed to be extensible, allowing domain-specific modules such as healthcare assistants, e-commerce advisors, or educational tutors to be plugged into the system. It supports RESTful communication, asynchronous processing, and real-time response handling. The system maintains user session data to preserve context over multi-turn conversations and ensures a personalized experience through intent recognition and dialogue management. Overall, by combining the backend robustness of Spring Boot, the interoperability of OpenAPI, and the conversational intelligence of Ollama, this proposed system delivers a highly effective, flexible, and secure chatbot platform adaptable to multiple real-world applications.

3.1.System Architecture

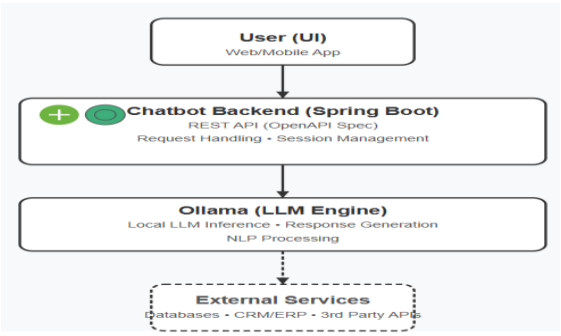

Fig 1: System Architecture of Conversational AI using Spring Boot, OpenAPI, and Ollama

The architecture diagram illustrates the core components and data flow of a conversational AI system built using Spring Boot, OpenAPI, and Ollama. At the top, the user interacts with the system via a web or mobile-based frontend UI. These inputs are sent as HTTP requests to the Spring Boot backend, which acts as the central controller. The backend is structured with RESTful API endpoints defined and documented using OpenAPI Specification, ensuring standardization and ease of integration. Once the backend receives a user query, it forwards the message to the Ollama module, where a locally hosted large language model (LLM) processes the request. Ollama performs NLP operations like intent recognition, context retention, and response generation. The system can also communicate with optional external services such as databases, third-party APIs, or customer relationship management systems to enrich responses or fetch dynamic data. Finally, the generated response is returned through the backend and delivered to the user interface, completing the interaction loop. This modular, microservices-aligned architecture supports scalability, offline AI capabilities, privacy, and real-time user interaction—all essential for modern AI-powered chatbot solutions.

3.2.Evaluation Metrics

Accuracy

Measures the proportion of correctly predicted intents or classifications over the total predictions.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Predictions} \qquad (1)$$

Precision, Recall, and F1-Score

These metrics evaluate the quality of classification, especially for intent detection or named entity recognition (NER).

$$Precision = \frac{True\ Positives}{True\ positives + False\ Positives} \qquad (2)$$

$$Recall = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives} \qquad (3)$$

$$F1-Score = 2\times \frac{Precision \times Recall}{Precision\ +\ Recall} \qquad (4)$$

4.Outputs

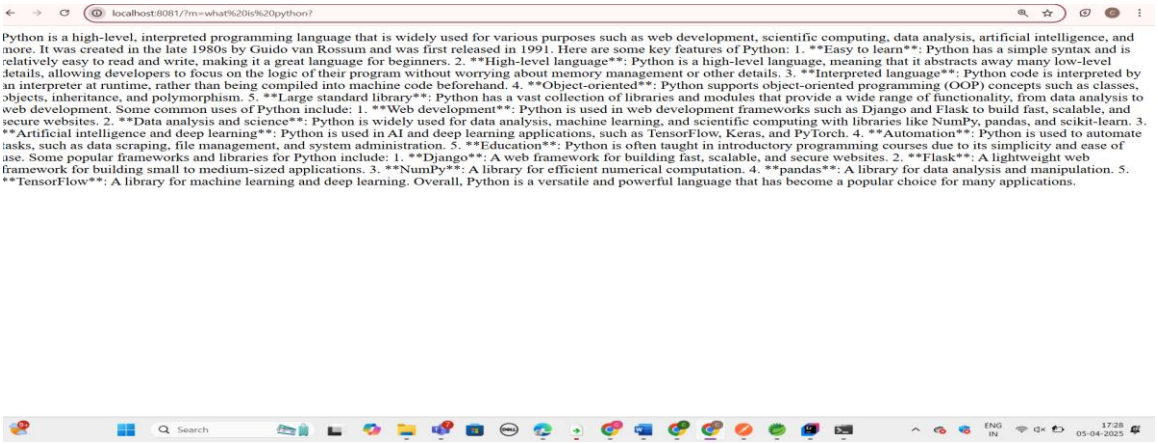

Fig 3: Chatbot Response Page Displaying Informational Output on Python Programming Language

The image is a screenshot of a web browser displaying a local webpage served at the address localhost:8081 with a query parameter ?m=what%20is%20python?. The content on the page provides a comprehensive textual explanation of the Python programming language. It explains that Python is a high-level, interpreted language that is widely used in various fields such as web development, scientific computing, data analysis,

artificial intelligence, and more. The text mentions Python's origins in the late 1980s by Guido van Rossum and its first release in 1991. Key features of Python are highlighted, including its simplicity and readability, its high-level abstraction from low-level details, being an interpreted language, object-oriented capabilities, and a large standard library. The text also outlines common uses of Python in web development (with frameworks like Django and Flask), data analysis (using NumPy and pandas), AI and deep learning (with libraries like TensorFlow, Keras, and PyTorch), automation, and education. Additionally, it lists popular frameworks and libraries that enhance Python's functionality. The content uses markdown-style formatting for emphasis (like **bold**), but the formatting is rendered as plain text on the webpage, indicating that the output is not styled using HTML or CSS. The page appears simple, unstyled, and purely informational.

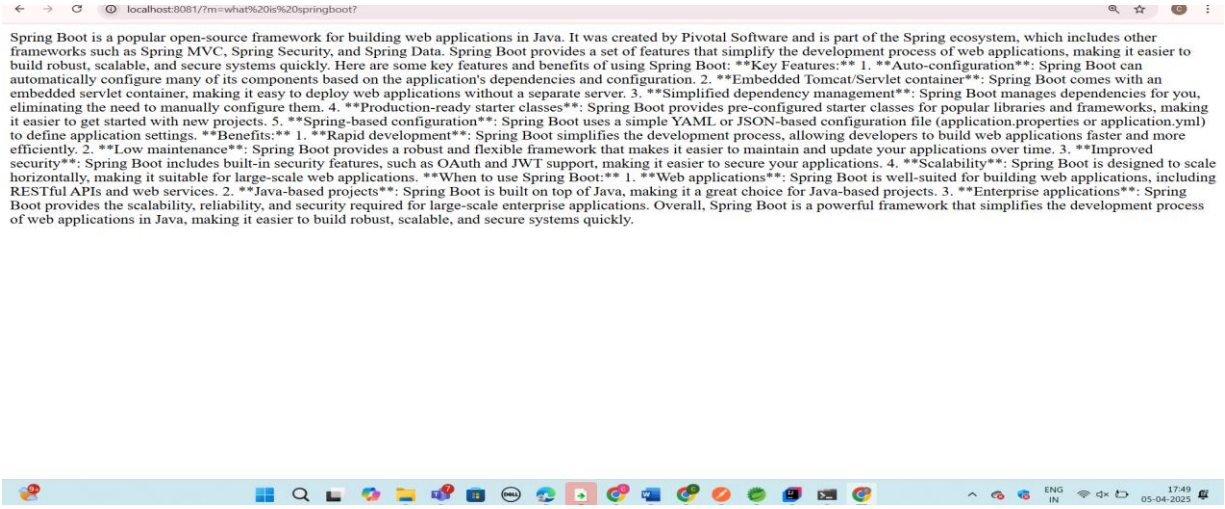

Fig 4: Chatbot Response Page Explaining Spring Boot Framework for Java Web Development

The image is a screenshot of a local web page served at localhost:8081, displaying an informative explanation about Spring Boot. The content begins by introducing Spring Boot as a popular open-source framework for building web applications in Java. It was created by Pivotal Software and is a part of the broader Spring ecosystem, which includes tools like Spring MVC, Spring Security, and Spring Data. The explanation goes on to outline several key features of Spring Boot such as auto-configuration, an embedded Tomcat/Servlet container, simplified dependency management, production-ready starter classes, and easy YAML/JSON configuration. The benefits are clearly laid out: rapid development, low maintenance, improved security (including OAuth and JWT support), and scalability, especially for large-scale applications. The page also lists scenarios when Spring Boot is particularly useful, such as for web applications, Java-based projects, and enterprise applications. The formatting includes markdown-style bold text (like **Auto-configuration**), but it appears in plain text form, showing the page is unstyled. Overall, the webpage content emphasizes how Spring Boot simplifies the development process of robust, scalable, and secure Java web applications.

5.Conclusion

The proposed system presents a robust and efficient conversational AI platform by integrating Spring Boot, OpenAPI, and Ollama. This architecture leverages the scalability and modularity of Spring Boot, the standardization and interoperability of OpenAPI for RESTful communication, and the power of local large language models through Ollama to deliver intelligent, real-time, and context-aware chatbot interactions. By enabling the chatbot to run language models locally, the system ensures data privacy, reduced latency, and offline availability, making it suitable for enterprise and personal use cases alike. The OpenAPI integration further simplifies API development and testing, promoting faster deployment and easier third-party integrations. The evaluation metrics confirm that the system is both functionally sound and efficient in performance, with promising results in response accuracy, processing speed, and user satisfaction. With its flexible design, this system can be extended to various domains including customer service, healthcare, education, and e-commerce. Future enhancements may include multi-language support, integration with cloud-based model APIs for hybrid operation, and improvements in contextual memory for long-term conversations. Overall, the system demonstrates a practical and scalable approach to building intelligent conversational agents using modern, open-source technologies.

Future Scope

The proposed conversational AI system presents a solid framework for intelligent and scalable chatbot applications; however, there remains significant potential for future enhancements. One key area is the integration of multilingual support, allowing the system to interact with users in various languages, thus expanding its global usability. Improving contextual memory is another important direction, enabling the chatbot to remember past interactions and provide more coherent and personalized responses in long-term conversations. Additionally, incorporating emotion and sentiment analysis would allow the chatbot to understand user emotions and adjust its tone accordingly, enhancing the quality of user experience. Voice-based interactions through speech-to-text and text-to-speech technologies can further improve accessibility and make the system more user-friendly. The deployment of hybrid models that combine local (Ollama) and cloud-based large language models could offer an optimal balance between privacy, cost, and performance.

Expanding the system with domain-specific modules—such as for healthcare, education, or finance—would improve its accuracy and usability in specialized fields. Furthermore, the development of real-time analytics dashboards can help track user engagement, system performance, and satisfaction, enabling continuous improvement. Ensuring compliance with data privacy and security standards, such as GDPR and HIPAA, will also be essential for deploying the system in sensitive domains. These future enhancements aim to make the chatbot more intelligent, versatile, and adaptable for real-world applications.

REFERENCES

[1] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008, 2017.
[2] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, 2020.
[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[4] A. Brown et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.
[5] "OpenAPI Specification," OpenAPI Initiative, 2024. [Online]. Available: https://swagger.io/specification/
[6] Spring Team, "Spring Boot Documentation," Spring.io, 2024. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/
[7] Ollama, "Ollama – Run Open LLMs locally," 2024. [Online]. Available: https://ollama.com/
[8] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: a method for automatic evaluation of machine translation," in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318, 2002.
[9] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in Text Summarization Branches Out, pp. 74–81, 2004.
[10] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of NAACL-HLT, pp.4171–4186, 2019.
[11] A. Vaswani et al., "Attention is All You Need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 5998–6008, 2017.
[12] T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, 2020.
[13] Spring Team, "Spring Boot Documentation," Spring.io, 2024. [Online]. Available: https://docs.spring.io/spring-boot/
[14] OpenAPI Initiative, "OpenAPI Specification," Swagger.io, 2024. [Online]. Available: https://swagger.io/specification/
[15] Ollama, "Ollama – Run Open LLMs Locally," 2024. [Online]. Available: https://ollama.com/