Journal of Artificial Intelligence and Digital Economy
Homepage : https://journal.antispublisher.id/index.php/JAIDE
Email : admin@antispublisher.com

e-ISSN : 3032-1077
JAIDE, Vol. 1, No. 12, December 2024
Page 75-83

AI Assistant-Based Chatbot Application in Solving Coding Script Issues Using the Ollama Large Language Models

Achmad Maulana Rochman [1], Sumarno [2], Suprianto[3], Irwan Alnarus Kautsar[4]
[1,2,3,4] Muhammadiyah University of Sidoarjo, Indonesia

DOI : https://doi.org/10.61796/jaide.v1i12.1544

Sections Info

Article history:
Submitted: September 05, 2024
Final Revised: October 23, 2024
Accepted: November 12, 2024
Published: December 31, 2024

Keywords:
Chatbots
Artificial intelligence
Large language models
Ollama
Streamlit
Programming

ABSTRACT

Objective: *This study aims to design and implement an artificial intelligence (AI)-based chatbot to assist software developers in debugging and generating programming solutions using Large Language Models (LLM) from Ollama.* ***Method:*** *The system was developed using the Waterfall methodology, encompassing stages of requirements analysis, design, implementation, testing, and maintenance. The AI model operates locally through Ollama, while Streamlit serves as the user interface for interactive communication between users and the chatbot.* ***Results:*** *Testing results show that the chatbot provides accurate and contextually relevant responses to programming queries, effectively supporting developers in debugging and code recommendation tasks. However, optimization is needed to improve system response time and processing efficiency.* ***Novelty:*** *The integration of a locally deployed LLM with a Streamlit-based interface presents a practical and secure solution for AI-assisted programming support, highlighting its potential to enhance developer productivity and reduce reliance on external cloud-based AI services.*

INTRODUCTION

The development of artificial intelligence (AI) technology has been growing rapidly in recent decades, having a significant impact on various sectors. One branch of AI that has experienced significant progress is natural language processing (NLP). Large language models (LLM), such as the Generative Pre-trained Transformer (GPT), are capable of processing and generating text with a level of accuracy approaching human intelligence [1]. This opens up significant opportunities for the development of applications that utilize AI to understand and generate human language in various contexts, including AI-based chat applications.

AI-based chatbot applications have been widely implemented in various sectors, such as customer service, education, and virtual assistants on mobile and desktop devices [2]. Most of these applications are web-based and utilize internet connectivity to access AI models running on central servers. By using cloud technology, these applications can offer flexibility in faster system development and updates, enabling the use of large language models such as those offered by the Ollama platform, which can optimize the user experience by providing relevant and accurate answers related to programming.

Ollama is one platform that provides access to a variety of large language models that can be integrated with local applications. This platform allows developers to build applications with AI models that can operate without relying on a central server, which

is very important in the context of software development applications [3]. The use of Ollama in web-based applications offers the advantage of leveraging large language models for specific tasks such as programming, debugging, or self-training [4] .

To facilitate the creation of user interfaces, this study utilizes Streamlit, an open-source framework that allows developers to build interactive web-based applications quickly and easily. Streamlit enables the creation of applications that are accessible to users without requiring in-depth front-end design skills, thereby accelerating the development process and focusing on application logic [5]. With Streamlit, web-based applications that integrate AI models can be easily accessed and used by developers to find quick solutions in debugging or answer questions about programming languages.

Based on these needs, this research focuses on the design and implementation of an AI-based chatbot application using Streamlit and leveraging Ollama's large language model. This application is designed to assist developers in solving programming language or code-related issues and provide quick solutions during the debugging process.

RESEARCH METHOD

Research methodology plays a crucial role in ensuring the application development and evaluation process meets its stated objectives. This study uses the Waterfall method for developing an AI Assistant-based chatbot application that focuses on software developer support. This method was chosen due to its structured approach, as shown in Figure 1 [6].

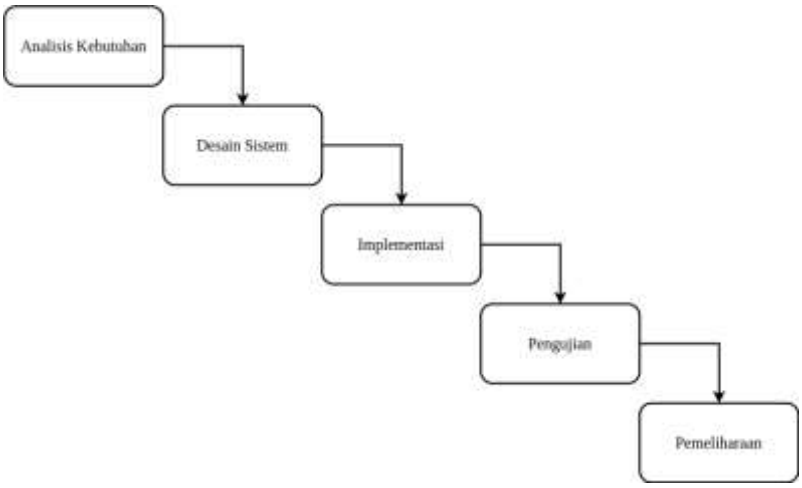

Figure 1. Waterfall Method Process Diagram

RESULTS AND DISCUSSION

A. Needs Analysis

At this stage, system requirements are collected based on literature studies and interviews with potential users, namely software developers. The needs analysis aims to understand the required features, such as the chatbot's ability to answer technical

questions related to debugging, code recommendations, and programming documentation [7].

B. System Design

This phase involves a comprehensive design for building a web-based chatbot system that uses Ollama's Large Language Models (LLM) as its natural language processing core and the Streamlit framework as its user interface. The system design encompasses the following aspects:

System Architecture

The system architecture is designed to integrate the main technologies used, such as virtual machines, Ollama for AI models, and Streamlit as a frontend framework. The system is designed to run efficiently with interconnected components [8] .

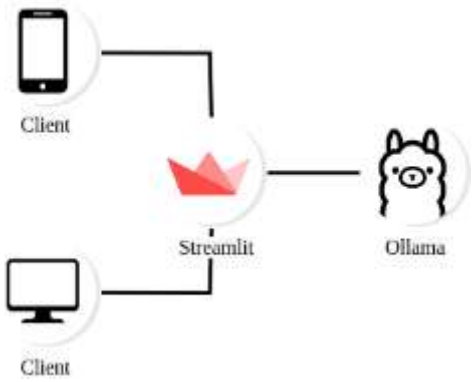

Figure 2. System Architecture

Wireframe Design

Wireframes depict the basic framework of an application's user interface, including the layout of the input prompt page, execution buttons, and result display area. Wireframes help in designing an intuitive user interaction flow [9] .

Figure 3. Wireframe Design

Usecase Diagram

Use case diagrams are used to model the relationship between actors (users) and the system, and describe key functionality such as entering prompts, validating model connections, processing input, and displaying output [6] .

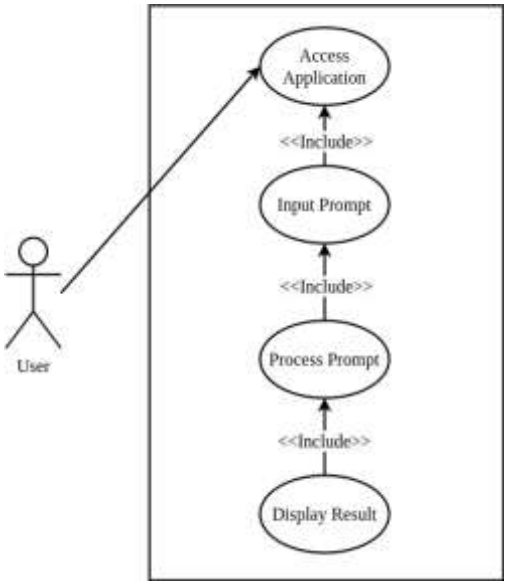

Figure 4. Usecase Diagram

Data Flow Diagram

A data flow diagram describes how data flows within the system, from the user to the backend, to the Ollama model, and back to the user in the form of results. This diagram includes both high-level and detailed information to describe the relationships between components [10].

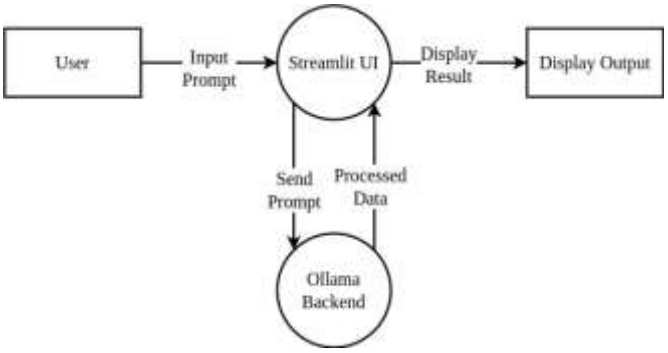

Figure 5. Data Flow Diagram

Flow chart

A flowchart depicts the logical flow of processes in a system, from user input to the output generated by the application. This flowchart provides a step-by-step understanding of how the system works [11].

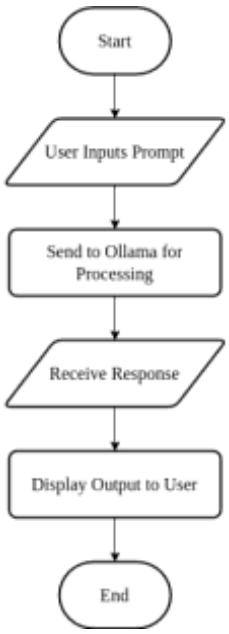

Figure 6. Flowchart

Sequence Diagram

The sequence diagram shows the sequence of communication between the user, the Streamlit interface, the application backend, and the Ollama model. This diagram describes the real-time interaction between components in processing a prompt [12].

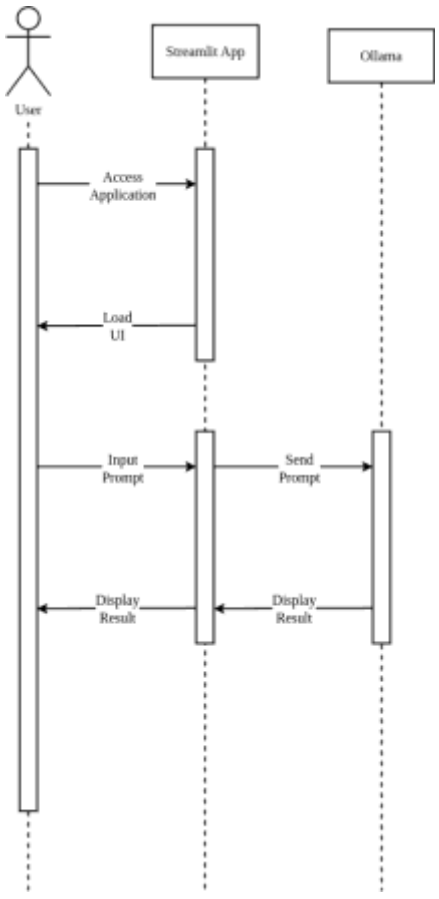

Figure 7. Sequence Diagram

C. Implementation

In the implementation phase, the system is developed according to the design that has been made [13]. The Ollama LLM model is integrated into the application, while the user interface is created using Streamlit. This process includes coding, model integration, and implementation of features that suit user needs.

Home Page

On the main page shown in Figure 5, users can select an available LLM model via the dropdown, enter a prompt or command in the input box, and press the Generate button to get a response from the model, shown in Figure 6. The Ollama Online status indicator ensures that the system is active and ready to use. With a minimalist design and a focus on functionality, this page is designed to provide a simple and efficient user experience in automatically generating code.

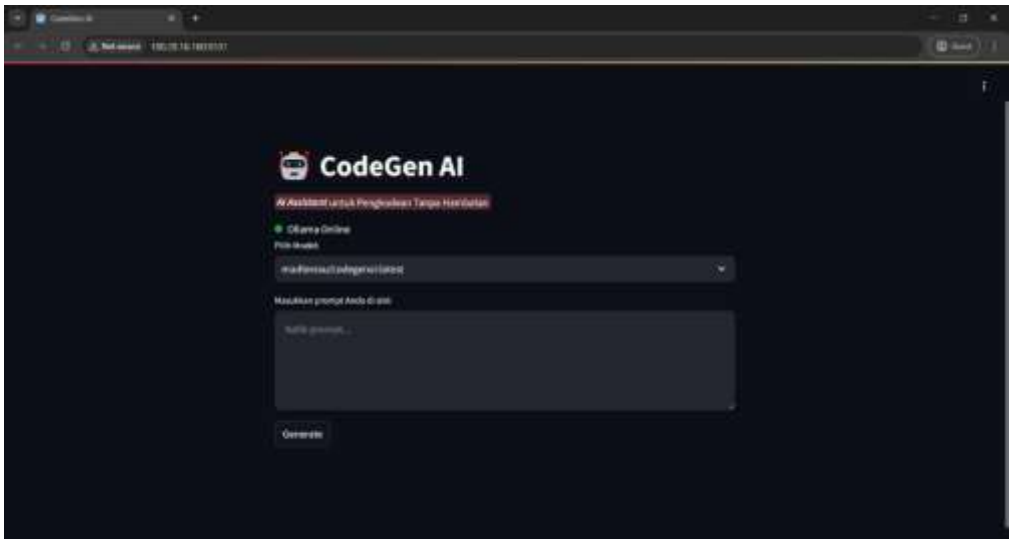

Figure 8. Main Page

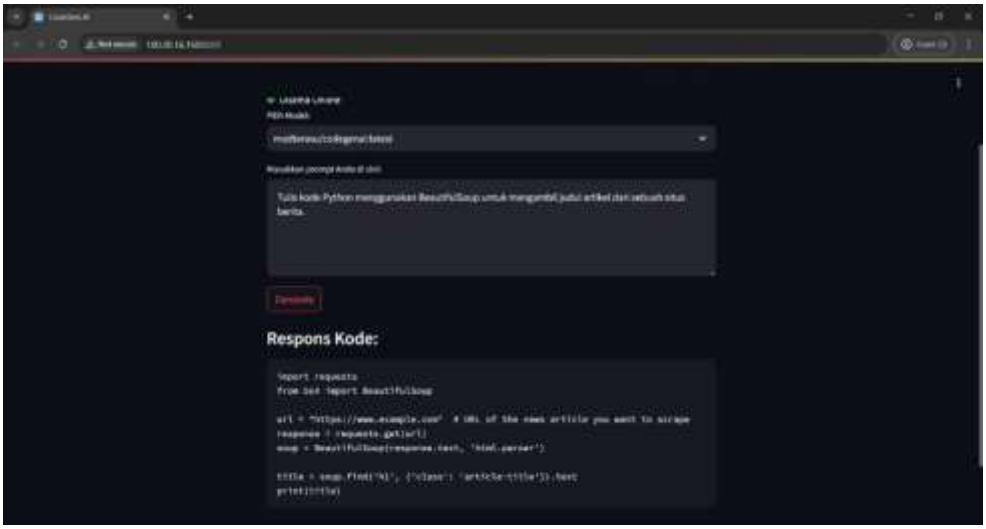

Figure 9. Main Page with Response

D. Testing

The testing phase is carried out to ensure that the application runs according to specifications. The testing method used is Black box testing. This method is a system test that is carried out from a functional specification perspective without testing the design and program code [14]. This black box testing focuses on the functional specifications of the software only, the tester can define a set of input conditions and perform testing on the functional specifications of the program [15]. The results of the tests carried out are shown in Table 1.

Table 1. Blackbox testing results

No	Types of Testing	Testing Objectives	Description	Success Criteria	Test Results
1.	Input and Output Testing	Ensures the system receives input correctly and produces appropriate output.	Test whether the system can accept text input from the user and provide relevant responses from the Ollama model.	Responses are appropriate to the input given, output is relevant to the context.	Succeed
2.	System Flow Testing	System Flow Testing	Testing the flow from the user entering a prompt, to the application displaying a response.	All system flows run smoothly without errors.	Succeed
3.	User Interface (UI) Testing	Ensure the interface looks as expected and is easy to use.	Test whether the Streamlit application interface is easy to understand and user-friendly.	The interface is easy to use and complies with specified design standards.	Succeed
4.	Response Performance Testing	Ensures the application responds within a reasonable time.	Tests the response time of an application from the moment input is provided.	Response time does not exceed 5 seconds in 90% of cases.	Fail
5.	Security Testing	Tests whether the application is secure against potential threats or input errors.	Test for incorrect or dangerous input (e.g. very long input or special characters) to ensure the application does not crash.	The application remains secure and does not crash when given incorrect or	Succeed

No	Types of Testing	Testing Objectives	Description	Success Criteria	Test Results
				malicious input.	

E. Maintenance

After the application is tested and implemented, maintenance is performed to fix bugs, enhance features, or adapt to new user needs. Maintenance also includes updating the LLM model to improve response accuracy.

CONCLUSION

Fundamental Finding : This study concludes that the development of an AI Assistant-based chatbot using Ollama's Large Language Models (LLM) and a Streamlit interface has successfully produced an intelligent system capable of assisting software developers in debugging and providing coding recommendations. The system demonstrated high accuracy and contextual relevance in responses, validating the effectiveness of the applied model and development approach. **Implication :** The findings suggest that integrating locally deployed AI systems can enhance software development efficiency by reducing dependency on external servers and ensuring greater accessibility. This innovation highlights the growing potential of AI assistants to become integral tools in programming and problem-solving workflows. **Limitation :** However, the study identified response time delays as a primary limitation, which may impact the fluidity of user interactions and overall performance in real-time scenarios. **Future Research :** Future work should focus on optimizing response speed, incorporating adaptive learning mechanisms, and expanding cross-platform integration to improve scalability, usability, and user engagement within various software development environments.

REFERENCES

[1] Sindy Nova, Nurul Khotimah, and Maria Y Aryati Wahyuningrum, "Utilization of Chatbot Using Natural Language Processing for Learning the Basics of Tkinter GUI in Python Programming Language," *J. Ilm. Tek.* , vol. 3, no. 1, pp. 58–65, 2024, doi: 10.56127/juit.v3i1.1162.

[2] S. Nurhayati and MA H, "Development of the Midwify Chatbot Application as a Supporting Media for Android-Based Midwifery Learning at Stikes Bhakti Kencana Bandung," *Komputika J. Sist. Komput.* , vol. 8, no. 1, pp. 45–52, 2019, doi: 10.34010/komputika.v8i1.1630.

[3] P. Dan, P. Skripsi, and D. Universitas, "Development of a Generative AI-Based Chatbot to Make it Easier for Students to Understand Guidelines," vol. 1, no. 2, pp. 19–32, 2024.

[4] F. Rizki, A. Sutiyo, NS Harahap, S. Agustian, and RM Candra, "KLIK: Scientific Study of Informatics and Computer Implementation of Question Answering Based on Telegram Chatbot on Al-Jalalain Interpretation Using Langchain and LLM," *Media Online* , vol. 4, no.

5, pp. 2464–2472, 2024, doi: 10.30865/klik.v4i5.1784.

[5] K. Bandung *et al.* , "Implementation of Folium and Streamlit on the Citarum River Water Quality Classification Website (CITASI)," vol. 11, no. 4, pp. 3148–3155, 2024.

[6] R. Farta Wijaya and R. Budi Utomo, "KLIK: Scientific Study of Informatics and Computers Using the Waterfall Method in Designing a Web-Based Mosque Activity Management Information System," *Media Online* , vol. 3, no. 5, pp. 563–571, 2023, [Online]. Available: https://djournals.com/klik

[7] A. Fergina, S. Alpariji, and A. Sujjada, "Designing a Natural Language Processing Whatsapp Chatbot for Digital Services at Nusa Putra University," *Jurasik (Jurnal Ris. Sist. Inf. dan Tek. Inform.* , vol. 9, no. 2, pp. 697–709, 2024.

[8] A. Irmayana, K. Aryasa, and Herlinda, "Student Attendance and Monitoring System Using Location Based Services (LBS) Method," *SISITI Semin. Ilm. Sist. Inf. and Technol. Inf.* , vol. 10, no. 2, pp. 124–133, 2021.

[9] MS Hartawan, "Implementation of User Centered Design (UCD) in Wireframe Design of User Interface and User Experience of Film Synopsis Application," *Jeis J. Elektro Dan Inform. Swadharma* , vol. 2, no. 1, pp. 43–47, 2022, doi: 10.56486/jeis.vol2no1.161.

[10] W. Harjono and Kristianus Jago Tute, "Designing a Web-Based Library Information System Using the Waterfall Method," *SATESI J. Sains Teknol. dan Sist. Inf.* , vol. 2, no. 1, pp. 47–51, 2022, doi: 10.54259/satesi.v2i1.773.

[11] MF Londjo, "Implementation of White Box Testing Using Path-Based Techniques in Login Form Testing," *J. Siliwaangi* , vol. 7, no. 2, pp. 35–40, 2021.

[12] N. Nurdam, "Sequence Diagram as a User Interface Design Tool," *J. Ultim.* , vol. 6, no. 1, pp. 21–25, 2014, doi: 10.31937/ti.v6i1.328.

[13] ENA Romadhoni, T. Widiyaningtyas, and U. Pujianto, "Implementation of the Waterfall Model in the Development of the Alumni Information System of SMKN 1 Jenangan Ponorogo," *Seminar. Nas. Sist. Inf. Indones.* , no. November, pp. 445–452, 2015.

[14] Nawassyarif, M. Julkarnain, and K. Rizki Ananda, "Web-Based Livestock Data Processing Information System for Technical Implementation Units of Production and Animal Health," *J. Inform. Technol. and Science* , vol. 2, no. 1, pp. 32–39, 2020, doi: 10.51401/jinteks.v2i1.556.

[15] J. Suwarno and G. Saputri, "Web-Based New Student Admissions Information System (PPDB) Using the Waterfall Model (Case Study: SDN Paku Jaya 02)," *Spectr. Multidiscip. Journal* , vol. 1, no. 21, pp. 123–141, 2024.

Achmad Maulana Rochman
Muhammadiyah University of Sidoarjo, Indonesia

* **Sumarno (Corresponding Author)**
Muhammadiyah University of Sidoarjo, Indonesia
Email: sumarno@umsida.ac.id

Suprianto
Muhammadiyah University of Sidoarjo, Indonesia

Irwan Alnarus Kautsar
Muhammadiyah University of Sidoarjo, Indonesia