Jurnal Teknologi Informasi dan Pendidikan
Volume 19, No. 1, March 2026
https://doi.org/10.24036/jtip.v19i1.1105

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

Development of an Interactive Chatbot Using Sahabat-AI Model with Retrieval-Augmented Generation Method

Fauzi Isyrin Apridal[1], Humaira[1*]✉, Fitri Nova[1]
[1]Information Technology Department, Politeknik Negeri Padang, Padang, Indonesia
**Corresponding Author: humaira@pnp.ac.id*

Article Information

Article history:

No. 1105
Rec. January 29, 2026
Rev. March 15, 2026
Acc. March 16, 2026
Pub. March 30, 2026
Page. 1380 – 1396

Keywords:

- *Chatbot*
- *Sahabat-AI*
- *RAG*
- *Llama*
- *FAISS*

ABSTRACT

The rapid advancement of language-based artificial intelligence, particularly Large Language Models, has enabled the development of adaptive and context-aware virtual assistants. This research aims to develop an interactive chatbot for Politeknik Negeri Padang utilizing the Sahabat-AI model (Gemma2 9B CPT), a large-scale model specialized in Bahasa Indonesia and local dialects (Javanese, Sundanese), combined with the Retrieval-Augmented Generation (RAG) method to enhance document-based answer accuracy. The system architecture integrates a Streamlit-based user interface supporting text/voice input and multilingual output, an automated web-scraping module using Scrapy to update institutional data, a structured knowledge base in Supabase, and a semantic vector search with FAISS. The development process followed a systematic design and implementation approach, with the RAG pipeline incorporating all-indo-e5-small-v4 embeddings to ensure semantic relevance. Performance evaluation using LangSmith demonstrated that Sahabat-AI outperformed Llama 3, achieving an average score of 0.84 (correctness: 0.89, relevance: 0.90, groundedness: 0.80, retrieval quality: 0.77) in Indonesian language testing. The chatbot exhibited strong local language understanding, scoring 0.74 for Javanese and 0.71 for Sundanese, while reducing hallucinations through RAG integration. Black-box testing confirmed the reliability of multimodal features such as speech-to-text and text-to-speech. The findings contribute to the development of the first Sahabat-AI–based multilingual chatbot for Politeknik Negeri Padang, integrating automated document retrieval and embedding pipelines for efficient information services.

How to Cite:
Apridal, F. I., & et al. (2026). Development of an Interactive Chatbot Using Sahabat-AI Model with Retrieval-Augmented Generation Method. Jurnal Teknologi Informasi Dan Pendidikan, 19(1), 1380-1396. https://doi.org/10.24036/jtip.v19i1.1105

Jurnal Teknologi Informasi dan Pendidikan
Volume 19, No. 1, March 2026
https://doi.org/10.24036/jtip.v19i1.1105

This open-access article is distributed under the Creative Commons Attribution-ShareAlike 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. ©2023 by Jurnal Teknologi Informasi dan Pendidikan.

1. INTRODUCTION

The rapid advancement of information technology in Indonesia, particularly artificial intelligence (AI), has significantly impacted various sectors, including higher education [1], [2], [3], [4], [5]. One of the applications of AI is the chatbot. Interactive chatbots are developed to support organizational needs in handling stakeholder question-and-answer interactions. A key focus in developing such chatbots is the continuous updating of their knowledge base. This ensures that stakeholder questions are answered with responses that remain relevant to the current conditions and situations within the organization. While public search engines like Google offer speed in global searches, they cannot access internal organization data and often fail to provide contextually relevant answers for it.

A previous project by Alvin Fadli Dwi Mulya developed an interactive chatbot using Llama 3 [6] with the Retrieval-Augmented Generation (RAG) method [7], effectively delivering academic and service-related information [8]. Despite its success, further improvements are needed, including broader coverage, faster response times, multimodal input (e.g., voice), and better adaptation to the Indonesian cultural context.

The emergence of Sahabat-AI, launched on 14 November 2024 through a collaboration between GoTo, Indosat Ooredoo Hutchison, Nvidia, AI Singapore, and the Indonesian Ministry of Communication and Digital Affairs offers new opportunities. Specially trained in Bahasa Indonesia, Javanese, Sundanese and English, Sahabat-AI excels in sentiment analysis, translation, and language inference with culturally relevant datasets [9].

This project aims to develop a Sahabat-AI–based chatbot for Politeknik Negeri Padang, leveraging RAG to deliver accurate, context-aware, and locally relevant answers. Key stages include needs analysis, system development, and data scraping [10], [11], [12], [13] for real-time updates, voice input support. The final stage will involve performance testing and comparative evaluation against the previous Llama 3-based implementation.

2. RESEARCH METHOD

This research followed the Waterfall development methodology, which proceeds sequentially through requirements analysis, system design, implementation, testing, deployment, and maintenance, as illustrated in Figure 1. This approach was selected because it enforces a clear structure, enabling thorough validation at each stage before

moving forward, which is essential for integrating multiple system components such as web scraping, a knowledge base, and a retrieval pipeline.

The target system is a campus information chatbot for Politeknik Negeri Padang (PNP) capable of processing both text and voice inputs, while offering two primary interfaces: an end-user chatbot application and an administrative web dashboard. The chatbot is designed to deliver factual, contextually relevant answers grounded in official institutional data. Source materials primarily come from official websites (e.g., pnp.ac.id), and department pages. These sources are processed into plain-text files through automated scraping and stored in a structured knowledge base to ensure they are accessible for retrieval.

Voice interaction was implemented using Google Speech-to-Text (STT) for capturing spoken queries and Google Text-to-Speech (TTS) for producing audio responses [14], [15].

The combination of these services ensures high transcription accuracy and natural-sounding output, which is important for accessibility, particularly for users with disabilities or limited typing skills.

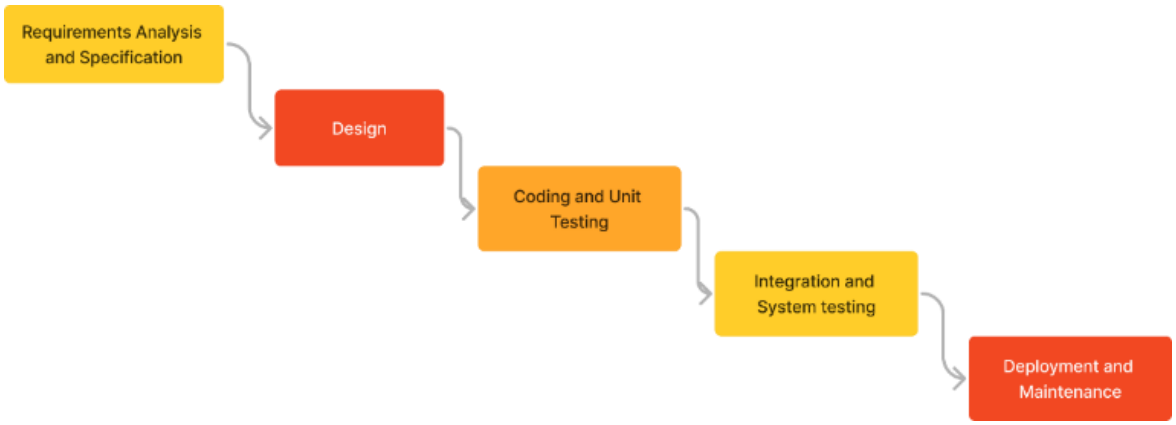

Figure 1. Waterfall development process

The data acquisition pipeline is built with the Scrapy framework, chosen for its asynchronous crawling capabilities and fine-grained control over crawl depth and request rates [11]. The crawler targets pre-defined URL patterns and is configured with a depth limit of 3, preventing unnecessary exploration of unrelated pages while still capturing relevant subpages such as program descriptions, announcements, and staff profiles. To reduce the risk of overloading source servers, a download delay and polite throttling are implemented.

The scraped HTML content is converted into normalized plain text (.txt) and stored in Supabase Storage [16]. Each file is linked with metadata, including its source URL, and last modification timestamp. This structure enables efficient updates: if a new checksum matches the stored one, the document is skipped, minimizing redundant processing. Scheduled scraping is triggered via cron-job.org on a weekly basis, while administrators may manually trigger on-demand updates through the dashboard interface.

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

The retrieval pipeline employs the RAG [7] pattern as outlined in Figure 2. Source documents are segmented into overlapping chunks. Each chunk is embedded using the all-indo-e5-small-v4 model, selected for its strong performance on Indonesian semantic similarity tasks. Embeddings [17], [18] are stored in a FAISS vector index, which enables millisecond-level similarity searches across thousands of documents.

At query time, the user's question is embedded and matched against the FAISS index to retrieve the top k (default k = 6) most relevant chunks [19]. These retrieved texts are injected into a guarded prompt template, which is then sent to the Sahabat-AI Large Language Model [20], [21] (Gemma 2 9B CPT variant) hosted on Replicate [22]. The prompt design enforces source attribution, ensuring that the model only generates answers supported by retrieved content. The generated output, along with references, is stored for logging and evaluation.

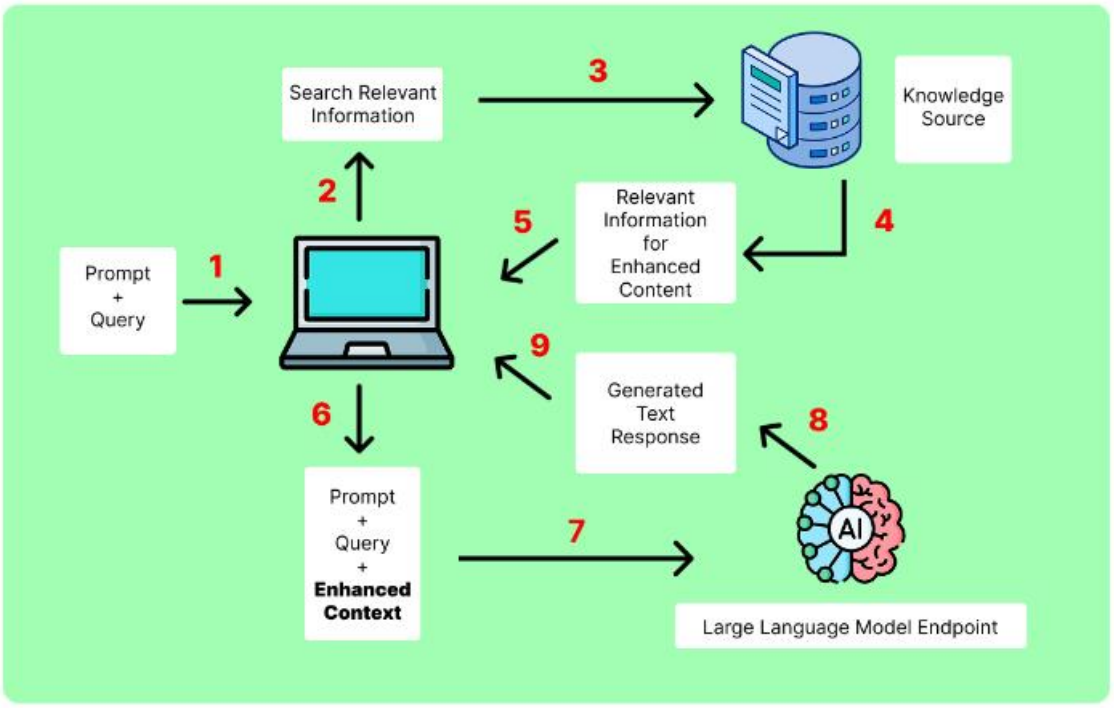

Figure 2. RAG pipeline overview

Jurnal Teknologi Informasi dan Pendidikan
Volume 19, No. 1, March 2026
https://doi.org/10.24036/jtip.v19i1.1105

2.1. Chatbot Design

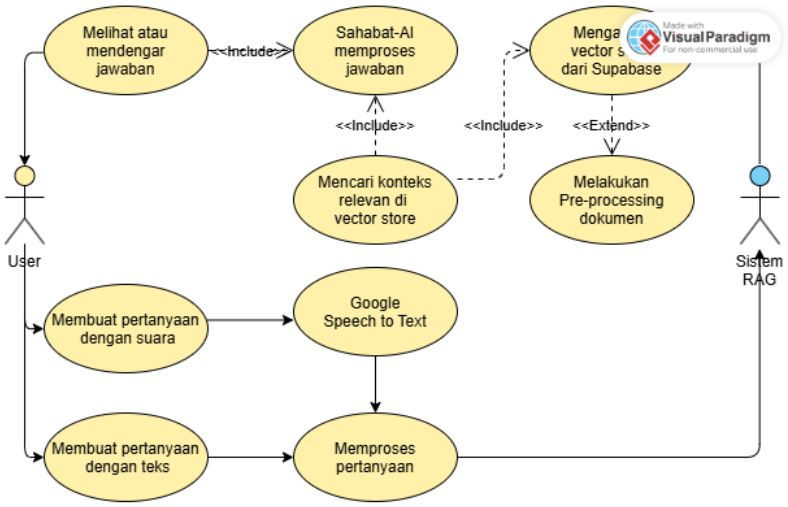

Figure 3. Chatbot use case diagram

The use case diagram Figure 3 illustrates the interaction between the user and the RAG-based chatbot system. Users can submit questions either via text or voice, with voice input processed through Google Speech-to-Text. The system retrieves relevant contexts from the vector store previously built from documents stored in Supabase before processing the query using the Sahabat-AI model. The user can then view or listen to the generated response. The system also includes a preprocessing stage for newly added documents.

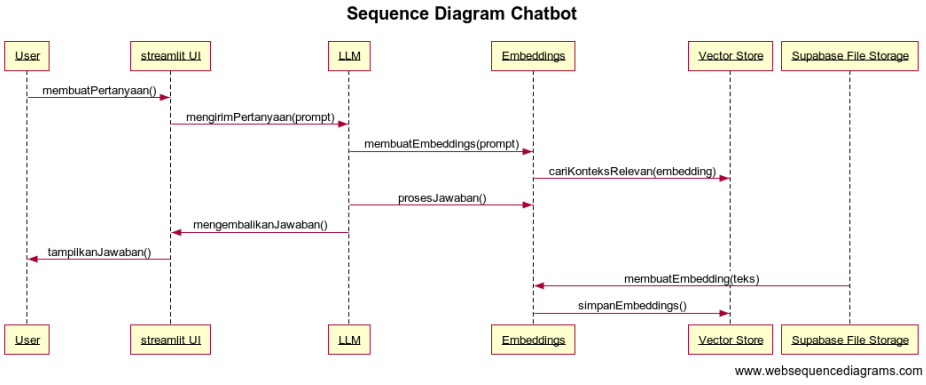

Figure 4. Chatbot sequence diagram

The sequence diagram as shown in Figure 4 describes the step-by-step interaction flow of the chatbot. The user creates a query, which is sent through the Streamlit [23] interface to the LLM. The LLM generates embeddings from the query and searches for relevant context within the vector store. Document embeddings are generated and stored in

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

Supabase File Storage as needed. The LLM then processes the retrieved context to produce a final answer, which is returned to the user interface for display.

2.2. Chatbot Admin Design

The admin dashboard was developed using Next.js, chosen for its scalability and efficient server-side rendering capabilities [24]. Supabase Auth manages secure email/password authentication to ensure that only authorized personnel can access the system, while Supabase Storage stores the uploaded text files along with their processed embeddings for retrieval and search purposes.

The use-case diagram, as shown in Figure 5, illustrates the main interactions between the administrator and the system. The administrator can securely log in, view and filter documents, upload new files, delete single or multiple documents, trigger automated scraping jobs, and monitor their progress in real time. This diagram provides a concise overview of the system's core functionalities from the administrator's perspective.

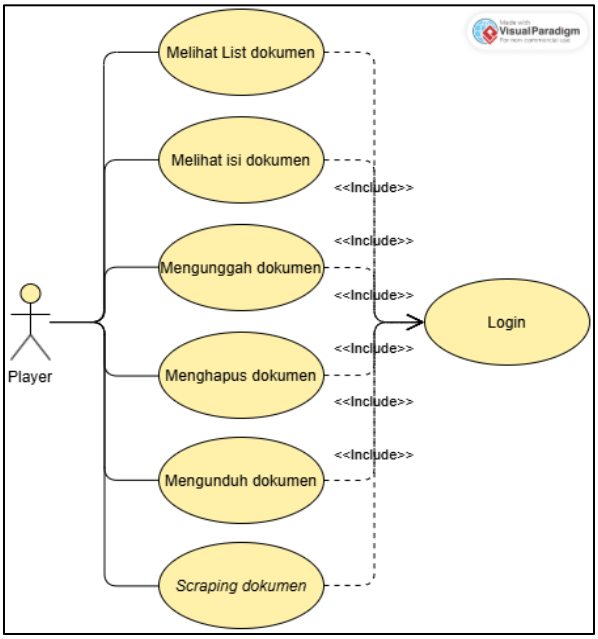

Figure 5. Admin use-case diagram

The system's key functionalities are demonstrated in a series of figures. The secure login process, as shown in Figure 6, allows administrators to enter their email and password, which are then verified through Supabase Auth before granting access to the dashboard.

Jurnal Teknologi Informasi dan Pendidikan
Volume 19, No. 1, March 2026
https://doi.org/10.24036/jtip.v19i1.1105

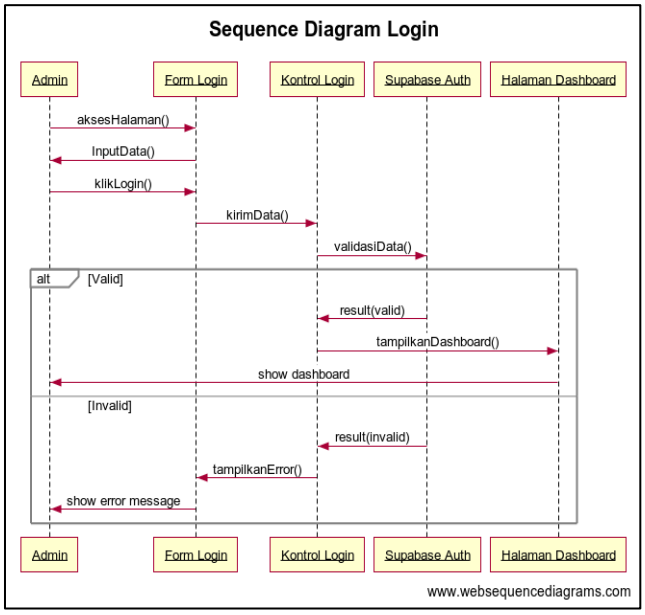

Figure 6. Sequence diagram: admin login

Document management features are also a core part of the system. Viewing, filtering, and previewing stored documents, as shown in Figure 7, enable administrators to quickly locate relevant files, check their contents, and apply filters to refine search results.

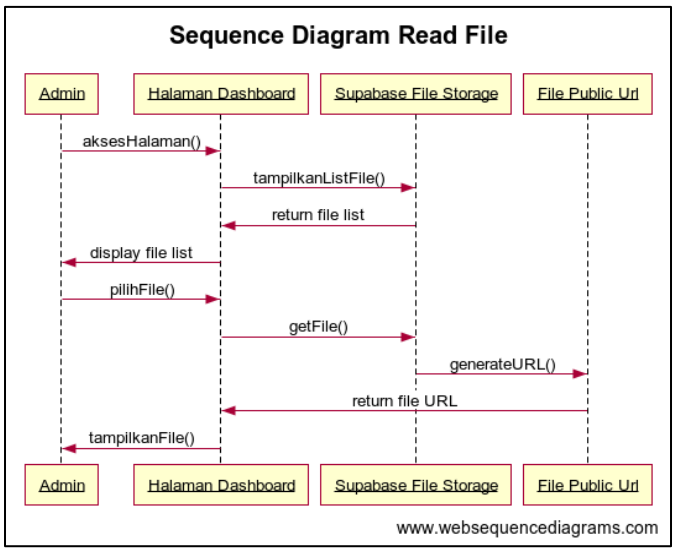

Figure 7. Sequence diagram: read/preview

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

The ability to upload new documents, as shown in Figure 8, provides a simple interface for adding text files to Supabase Storage, after which the system automatically processes them into embeddings and updates the vector store.

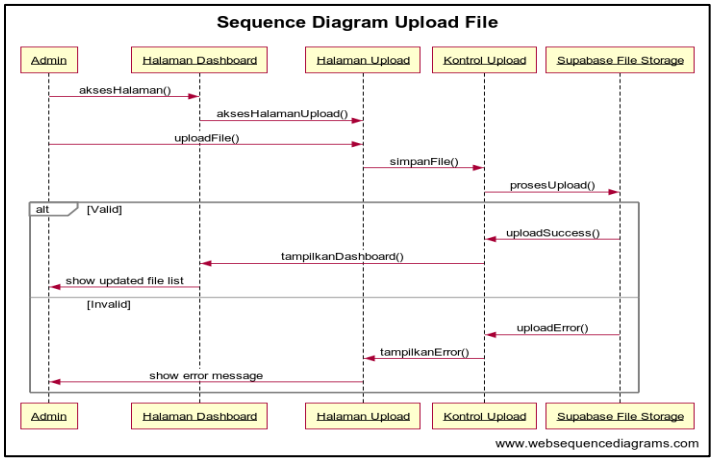

Figure 8. Sequence diagram: upload

For content maintenance, the dashboard supports deleting one or multiple documents simultaneously, as shown in Figure 9, ensuring that outdated or irrelevant data can be efficiently removed from both storage and the vector index.

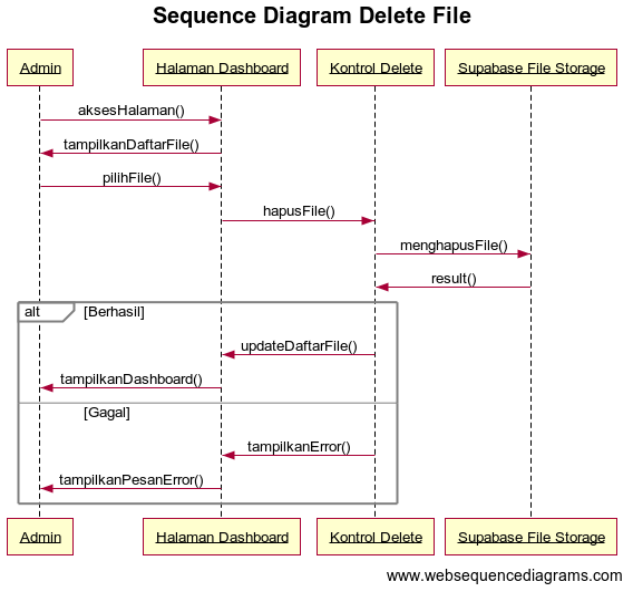

Figure 9. Sequence diagram: delete

Finally, scraping job management, as shown in Figure 10, allows administrators to initiate automated data extraction processes from predefined sources and track the progress

in real time through a built-in monitoring panel. This feature helps ensure that the database remains current without requiring constant manual updates.

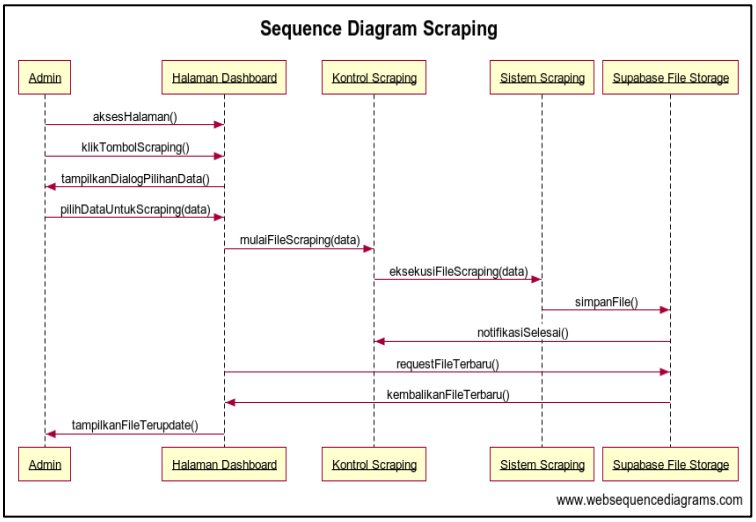

Figure 10. Sequence diagram: trigger scraping

The system architecture as shown in Figure 11 shows the interaction between the chatbot frontend, admin dashboard, RAG pipeline, and backend services. The deployment strategy as shown in Figure 12 uses Hugging Face Spaces [25] for hosting the frontends, Replicate for LLM inference, and Supabase for authentication, storage, and database.

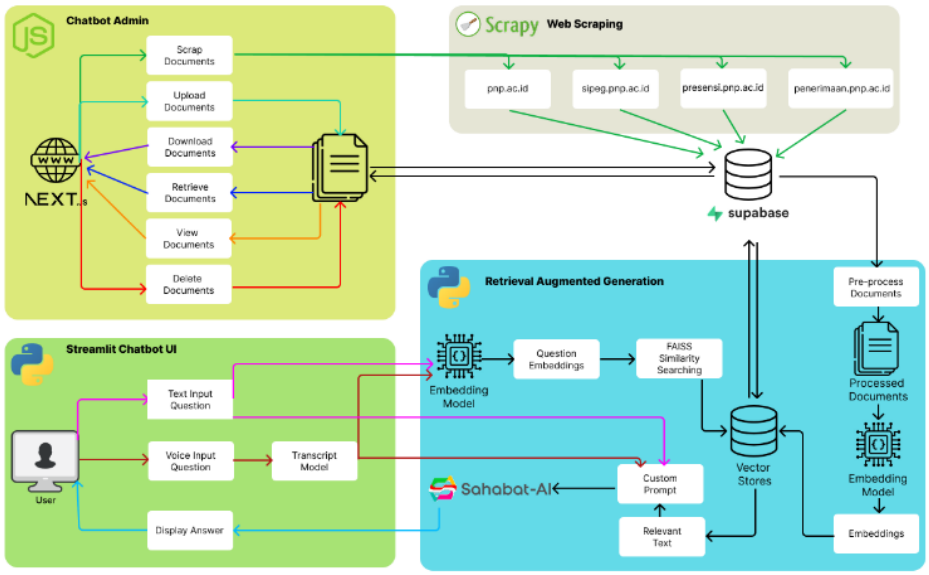

Figure 11. System architecture

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

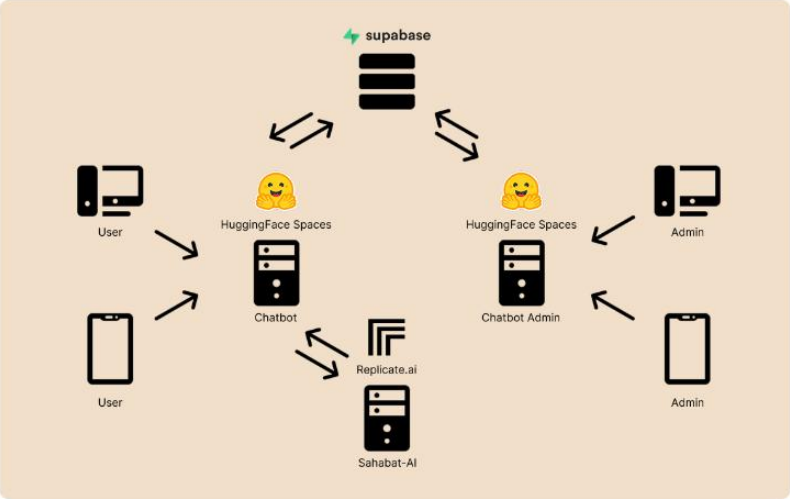

Figure 12. System infrastructure

Testing combined functional black-box checks and LLM evaluation. Black-box testing covered both the user-facing chatbot and the admin dashboard. On the user side, scenarios included text queries with TTS enabled/disabled, voice queries with TTS enabled/disabled, and the verification of source citations. On the admin side, tests included login/logout, valid and invalid document uploads, previewing and filtering documents, deleting records, downloading backups, and triggering scraping jobs. For each scenario, expected outputs and UI messages were pre-defined.

For answer quality evaluation, LangSmith [26] from Langchain [27] was used to run batch evaluations with LLaMA 3 acting as LLM-as-a-Judge [28]. Four metrics Correctness, Relevance, Groundedness, and Retrieval Quality were scored on a 0–1 (the higher the better) scale according to a predefined rubric. The evaluation process logs each prompt, retrieved context, generated output, model-judge rationale, and score, producing aggregated statistics for analysis. Any weaknesses identified, such as low groundedness, inform adjustments to retrieval parameters (k, chunk size/overlap) and prompt engineering, followed by re-evaluation.

Operations and maintenance procedures ensure the system remains up-to-date. A weekly scheduled cron job [29] scraping and re-indexing job runs automatically, with the option for administrators to trigger immediate updates.

3. RESULTS AND DISCUSSION

The implementation stage involved executing the project according to the designs and specifications established in earlier phases, followed by functional and quality evaluations to ensure the resulting system met expectations. The primary development environment used a Lenovo LOQ 15 laptop running Windows 11, with 20 GB of RAM, an

Intel Core i7-13650HX processor, and an NVIDIA GeForce RTX 4060 8 GB GPU. For inference, the chatbot employed the open-source Sahabat-AI model (Gemma 2 9B CPT), hosted on Replicate using an NVIDIA L40S GPU with 48 GB VRAM.

3.1. Implementation Results

The end-user chatbot interface as shown in Figure 13 enables communication between users and the system through either text or voice input, with responses generated via the RAG pipeline. The interface was built using Streamlit in Python.

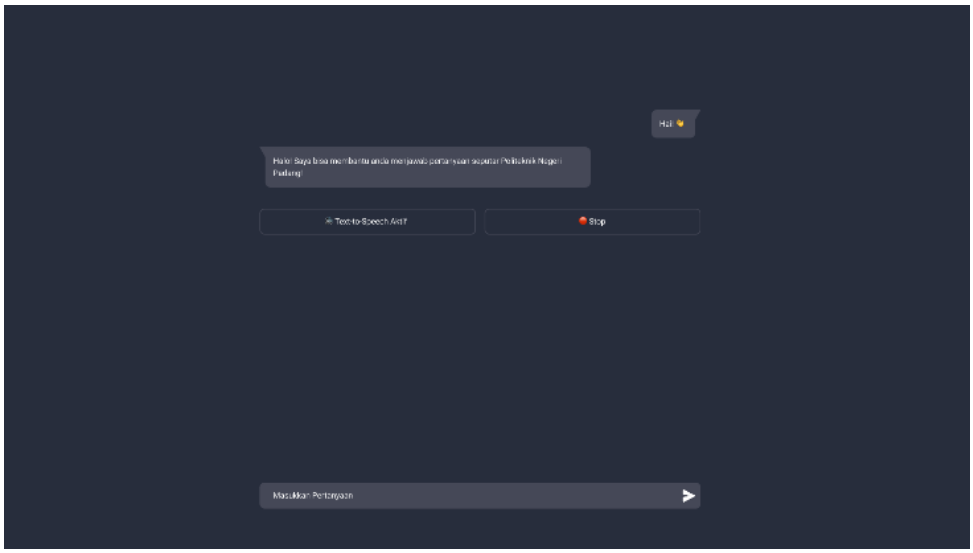

Figure 13. Chatbot interface

Figure 14. Admin login page

The admin interface begins with a secure login page as shown in Figure 14 developed using Next.js, React.js [30], and Supabase Auth to restrict access to verified administrators only.

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

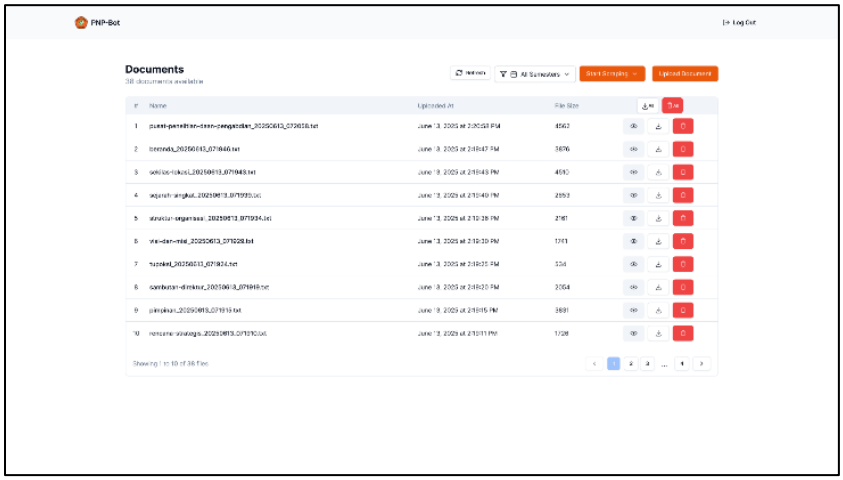

Figure 15. Admin dashboard

Upon successful login, administrators are presented with the dashboard as shown in Figure 15, where they can monitor, manage, filter documents by semester, add new documents by.

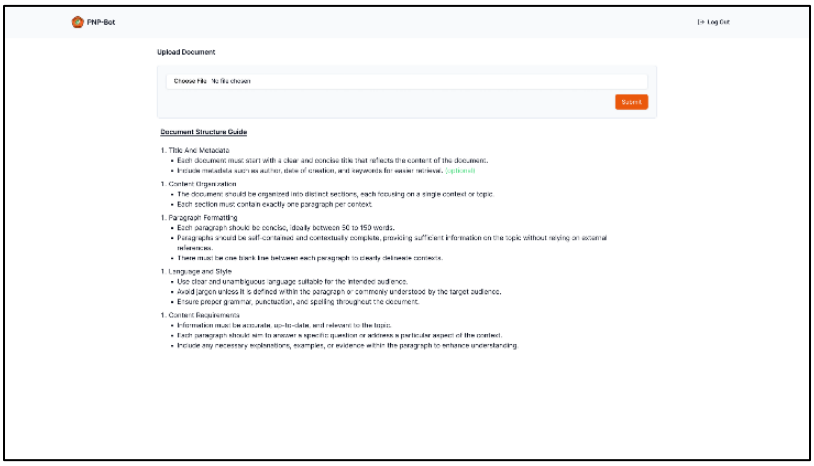

Figure 16. Document upload page

Additional functionalities include a document upload page as shown in Figure 16, enabling administrators to add new documents manually.

System TestingTwo testing approaches were employed: functional testing via the Black-Box method and quality evaluation using LangSmith.Black-box testing verified both user and admin functionalities without inspecting source code [31]. The chatbot was tested for various interaction modes (text with TTS on/off, voice with TTS on/off), ensuring expected outputs such as message rendering and the presence/absence of audio tags. The admin interface was evaluated for login authentication, document viewing, uploading,

deleting, scraping initiation manual and via cron-job.org, filtering by semester, and refreshing the dashboard. All test cases met their expected outcomes as shown in Table 1 and Table 2.

Table 6. Testing input and tts features

Test Case	Action	Expected	Status
Text Input + TTS Enabled	Enable TTS, enter text, press Enter	Response appears, audio tag present	Pass
Text Input + TTS Disabled	Disable TTS, enter text, press Enter	Response appears, audio tag not present	Pass
Voice Input + TTS Enabled	Enable TTS, speak the question	Response appears, audio tag present	Pass
Voice Input + TTS Disabled	Disable TTS, speak the question	Response appears, audio tag not present	Pass

Table 2. Functional testing of admin dashboard

No.	Test	Action	Expected	Status
1	Access Login Page	Open the web	Login page with email & password form loads	Pass
2	Admin Login	Fill login form, click "Submit"	Redirected to admin dashboard	Pass
3	Verify Document Count	Check document count on dashboard	Document count text visible	Pass
4	Upload Document	Click "Upload Document", select file, click "Submit"	Document uploaded, success notification shown	Pass
5	Delete Document	Click "Trash" on selected document	Document deleted, success notification shown	Pass
6	View Document	Click "Eye" on selected document	Document content opens in new tab	Pass
7	Start Scraping	Click "Start Scraping", choose "Data Dosen"	Scraping starts, dashboard displayed	Pass
8	Start Cron Job Scraping	Test run API URL on cron-job.org	Scraping starts, results shown on dashboard	Pass
9	Filter by Semester	Select "All Semesters" → "Ganjil 2025/2026"	Table shows filtered data	Pass
10	Show All Documents	Click "Show all documents"	Table shows all documents	Pass
11	Refresh Page	Click "Refresh"	Page updates with latest data	Pass
12	Log Out	Click "Log Out"	Returned to login page	Pass

To assess RAG output quality, LangSmith was used with the LLM-as-a-Judge approach, employing LLaMA 3 as the evaluation model. Four metrics were scored from 0 to 1 the higher the better: Correctness (accuracy against ground truth), Relevance (appropriateness to the query), Groundedness (factual basis from retrieved documents), and Retrieval Quality (semantic relevance of retrieved chunks). The evaluation covered four languages Indonesian, Javanese, Sundanese, and English across multiple queries as shown in Figure 17.

Results indicated that Sahabat-AI outperformed LLaMA 3 in most cases, particularly in Correctness and Relevance for Indonesian (0.89 and 0.90 vs. 0.77 and 0.79) and Javanese (0.79 and 0.82 vs. 0.59 and 0.70). In Sundanese, both models performed similarly overall (0.71 each), with Sahabat-AI stronger in Correctness (0.78 vs. 0.60) and

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

Relevance (0.80 vs. 0.78), while LLaMA 3 led in Groundedness and Retrieval Quality. For English, Sahabat-AI held a slight advantage in Correctness (0.82 vs. 0.56) and Retrieval Quality (0.85 vs. 0.80), but LLaMA 3 was more consistent in language adherence.

Overall, Sahabat-AI demonstrated superior multilingual performance, especially in maintaining linguistic consistency with the input language an essential feature for chatbots in Indonesia.

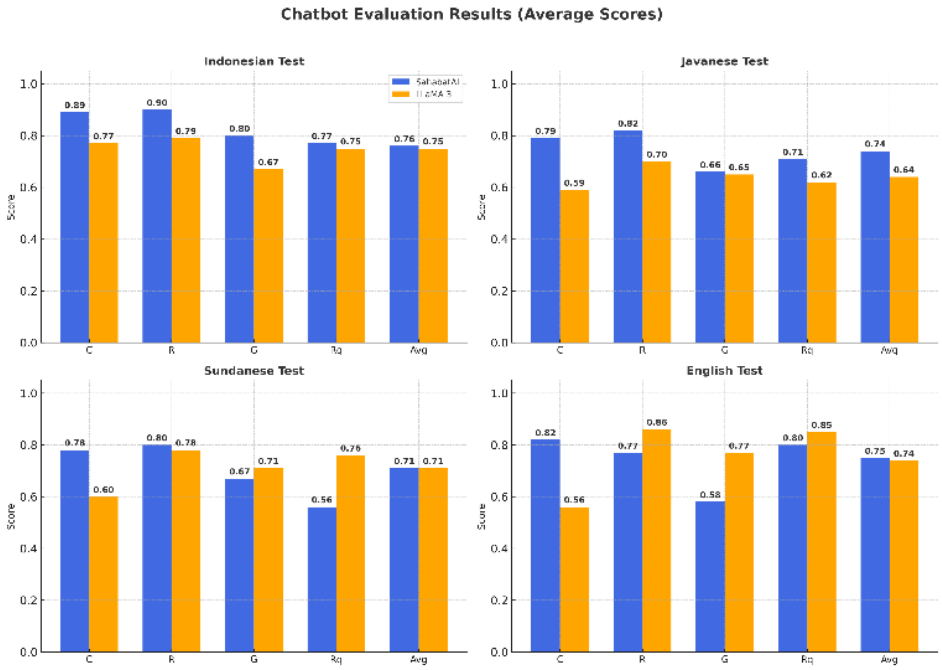

Figure 17. Chatbot evaluation graphs

3.2. Discussion

Compared with the work of Alvin Fadli Dwi Mulya (2024), this research introduces significant feature enhancements, notably the integration of voice input/output and automated web scraping. Both projects implemented RAG, but this study used Sahabat-AI (Gemma 2 9B CPT) fine-tuned for Indonesian, Javanese, and Sundanese, enabling richer local language support. The system architecture is more complex. Evaluation methods also differ, with this study employing LangSmith instead of Ragas, enabling more granular analysis through multiple quality metrics as shown in Table 3.

This chatbot system is equipped with web scraping technology to update its knowledge base. However, one of the challenges is that changes in the organization's web technology may hinder the data collection process. This condition can affect the chatbot system, causing it to generate responses that are no longer relevant to the most recent knowledge from its organization.

Table 3. Comparison with previous studies detail

Aspect	Alvin (2024)	Fauzi (2025)
AI Model	Llama 3 (Meta)	Sahabat-AI (Gemma2 9B CPT)
Language	Global (Llama-based)	Local (Indonesian, Javanese, Sundanese)
Method	RAG	RAG
Objective	Provide info on TI PNP department	Provide info on PNP from official site scraping
Features	Text input	Text & voice input, auto scraping
Evaluation	Black Box & Ragas	Black Box & LangSmith
Strengths	Early RAG–LLM integration focus	More complex system, local context
Limitations	No voice input or scraping	Dependent on source website

4. CONCLUSION

Based on the implementation and testing of the interactive chatbot system developed using the Sahabat-AI model with the RAG method, several key conclusions can be drawn.

First, the chatbot system was successfully built with an integrated architecture in which core components such as the Streamlit-based user interface, automated web scraping module using Scrapy, Supabase and FAISS-based knowledge storage, and the Sahabat-AI model integration operate synergistically to deliver campus information services.

Second, the RAG approach proved effective in enhancing response quality by combining document retrieval and natural language generation, producing relevant, accurate, and contextual answers. Evaluation using LangSmith demonstrated high scores across correctness, relevance, groundedness, and retrieval quality metrics by relying on valid and up-to-date document sources.

Third, Sahabat-AI demonstrated strong multilingual capabilities, effectively understanding and responding in Bahasa Indonesia, Javanese, and Sundanese an advantage over more generic models such as LLaMA 3 used in previous studies although it has not yet achieved consistent language alignment for English queries.

Fourth, the chatbot's speech input and output features functioned effectively, as confirmed through black-box testing, offering an inclusive alternative for users who prefer or require verbal interaction over text input. Finally, the developed chatbot is capable of delivering a wide range of information, from institutional details to department profiles, study programs, and class schedules at Politeknik Negeri Padang. This eliminates the need for users to manually search through official websites or documents, thereby improving the efficiency of information delivery and enhancing the overall user experience.

REFERENCES

[1] D. R. Arikkat et al., "IntellBot: Retrieval Augmented LLM Chatbot for Cyber Threat Knowledge Delivery," Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.05442

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

[2] A. Fitriansyah Universitas Borobudur, "Edusight International Journal of Multidisciplinary Studies Implementation Of Technology And Information Developments In Improving Indonesian Education," 2024.
[3] N. Mayasari et al., "Effectiveness of Using Artificial Intelligence Learning Tools and Customized Curriculum on Improving Students' Critical Thinking Skills in Indonesia," The Eastasouth Journal of Learning and Educations, vol. 2, no. 02, pp. 111–118, 2024, doi: 10.58812/esle.v2i02.
[4] S. Saprudin, "Artificial Intelligence Function Management in Supporting the Process of Government Implementation and Public Services in Indonesia," Journal of Management and Administration Provision, vol. 4, no. 1, pp. 88–96, Jul. 2024, doi: 10.55885/jmap.v4i1.352.
[5] I. Pujiono, I. M. Agtyaputra, and Y. Ruldeviyani, "Implementing Retrieval-Augmented Generation And Vector Databases For Chatbots In Public Services Agencies Context," JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer), vol. 10, no. 1, pp. 216–223, Aug. 2024, doi: 10.33480/jitk.v10i1.5572.
[6] A. Grattafiori et al., "The Llama 3 Herd of Models," Nov. 2024, [Online]. Available: http://arxiv.org/abs/2407.21783
[7] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," May 2020, [Online]. Available: http://arxiv.org/abs/2005.11401
[8] A. Fadli Dwi Mulya, "Pengembangan Chatbot Interaktif Menggunakan Large Language Model Dengan Metode Retrieval-Augmented Generation," Padang, Jul. 2024.
[9] I. O. H. GoTo, "GoToCompany/gemma2-9b-cpt-sahabatai-v1-instruct · Hugging Face." Accessed: Dec. 17, 2024. [Online]. Available: https://huggingface.co/GoToCompany/gemma2-9b-cpt-sahabatai-v1-instruct
[10] Y. Dikilitaş et al., "Performance Analysis for Web Scraping Tools: Case Studies on Beautifulsoup, Scrapy, Htmlunit and Jsoup," in Lecture Notes in Networks and Systems, Springer Science and Business Media Deutschland GmbH, 2024, pp. 471–480. doi: 10.1007/978-3-031-56728-5_39.
[11] H. Chaib, M. El Asikri1, S. Krit, and H. Chaib, "Using Web Scraping In A Knowledge Environment To Build Ontologies Using Python And Scrapy Article in," Eur J Transl Clin Med, 2020, [Online]. Available: https://www.researchgate.net/publication/346215371
[12] N. K. Kahlon and W. Singh, "Comparative Analysis of Web Scraping Tools for Low-Resource Language Text," International Journal of Engineering Trends and Technology, vol. 72, no. 1, pp. 284–299, Jan. 2024, doi: 10.14445/22315381/IJETT-V72I1P128.
[13] A. Ulfah and I. Najiah, "Implementasi Web Scraping Pada Situs Jurnal Sinta Menggunakan Framework Selenium Webdriver Python," JIKA (Jurnal Informatika), vol. 7, no. 1, p. 29, Feb. 2023, doi: 10.31000/jika.v7i1.7037.
[14] H. J. Yang, E. B. Oh, and J. M. Kim, "Comparison of Automatic Speech Recognition System for School-aged Children's Narratives: Naver Clova Speech and Google Speech-to-Text," Communication Sciences and Disorders, vol. 28, no. 1, pp. 30–38, 2023, doi: 10.12963/csd.23952.
[15] N. ' Arrizqi, I. Santoso, D. Yosua, and A. A. Soetrisno, "Implementasi Google Text To Speech Pada Aplikasi Pendeteksi Uang Berbasis Android," Jurnal Ilmiah Teknik Elektro Undip, vol. 10, no. 3, pp. 2685–0206, 2021, doi: https://doi.org/10.14710/transient.v10i3.510-516.
[16] A. Zewdie Ayezabu, "Ayezabu Amanuel Supabase vs Firebase: Evaluation of performance and development of Progressive Web Apps," 2022. Accessed: May 22, 2025. [Online]. Available: https://www.theseus.fi/handle/10024/771009

[17] M. A. H. Wadud, M. F. Mridha, and M. M. Rahman, "Word Embedding Methods for Word Representation in Deep Learning for Natural Language Processing," Iraqi Journal of Science, vol. 63, no. 3, pp. 1349–1361, 2022, doi: 10.24996/ijs.2022.63.3.37.
[18] L. Wang et al., "Text Embeddings by Weakly-Supervised Contrastive Pre-training," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2212.03533
[19] M. Douze et al., "The Faiss library," Jan. 2024, doi: https://doi.org/10.48550/arXiv.2401.08281.
[20] H. Naveed et al., "A Comprehensive Overview of Large Language Models," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.06435
[21] F. Koto, N. Aisyah, H. Li, and T. Baldwin, "Large Language Models Only Pass Primary School Exams in Indonesia: A Comprehensive Test on IndoMMLU Department Natural Language Processing, MBZUAI," 1237. [Online]. Available: https://github.com/
[22] R. Replicate, "Replicate - Run AI with an API.," https://replicate.com/.
[23] S. Samanta, A. Saha, P. Pramanick, S. Zaman, and A. Mitra, "Application of Python-Based Streamlit App for Evaluating Growth and Feed Efficiency in Pterophyllum scalare," Parana Journal of Science and Education, vol. 11, no. 11, pp. 5–12, 2025, [Online]. Available: https://sites.google.com/site/pjsciencea
[24] M. Fariz, S. Lazuardy, and D. Anggraini, "Modern Front End Web Architectures with React.Js and Next.Js," International Research Journal of Advanced Engineering and Science, vol. 7, no. 1, pp. 132–141, 2022.
[25] J. Jones, W. Jiang, N. Synovic, G. K. Thiruvathukal, and J. C. Davis, "What do we know about Hugging Face? A systematic literature review and quantitative validation of qualitative claims," Jun. 2024, doi: 10.1145/3674805.3686665.
[26] L. Langsmith, "Evaluate a RAG application | LangSmith," https://docs.smith.langchain.com/evaluation/tutorials/rag.
[27] V. Mavroudis, "LangChain," 2024. Accessed: May 22, 2025. [Online]. Available: https://hal.science/hal-04817573/
[28] L. Zheng et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2306.05685
[29] cron-job.org, "Cron Job," https://cron-job.org/en/.
[30] J. Kryk and M. Plechawska-Wójcik, "Multi-aspect comparative analysis of JavaScript programming frameworks - React.js and Solid.js," 2025.
[31] A. Maspupah, "Literature Review: Advantages And Disadvantages Of Black Box And White Box Testing Methods," Jurnal Techno Nusa Mandiri, vol. 21, no. 2, pp. 151–162, Sep. 2024, doi: 10.33480/techno.v21i2.5776.

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id