VIETNAM NATIONAL UNIVERSITY

UNIVERSITY OF ENGINEERING AND TECHNOLOGY

PROJECT SECTION REPORT

Project Title:

APPLICATION OF LOW-CODE PLATFORM FOR BUILDING A CHATBOT

Student: Ngo Van Kiet – 22022643

Advisor: Ph.D Nguyen Viet Cuong

Hanoi, 2025

RielLowBot

Towards chatting with low-code platforms

Student Ngo Van Kiet – 22022643

Advisor Ph.D Nguyen Viet Cuong

My Project Section Github

Abstract

This project focuses on building a Low-code Chatbot using n8n as the main automation tool. The chatbot is designed to automatically collect AI-related news, summarize content, and answer user questions through Telegram. By connecting different tools like Ollama (to run the language model locally) and Elasticsearch (to store and search data), the chatbot can provide real-time, helpful responses to users. The main goal is to show how n8n can be used to build a smart chatbot system without needing extensive coding. The chatbot works by crawling news websites, summarizing the articles, saving them to a database, and later using that data to answer questions with the help of AI. The integration with Telegram makes it easy for users to chat with the bot in a familiar app.

This project proves that even non-programmers can create a useful AI chatbot by combining n8n workflows with AI models and databases. The system helps save time by automating tasks and shows a simple way to bring AI into real-life projects.

1 Introduction

In recent years, artificial intelligence (AI) has rapidly advanced and become a core part of many industries. One of the most popular AI applications is the chatbot, which is widely used in customer service, education, marketing, and various other fields to provide instant support and automate repetitive tasks. AI-powered chatbots have made it possible to handle large volumes of user queries efficiently, saving both time and resources. However, despite their growing popularity, building a fully functional AI chatbot often requires strong programming skills and complex system integration. Developers need to connect AI models, databases, APIs, and user interfaces into a cohesive system, which can be a significant barrier for individuals and small teams without deep technical expertise.

1.1 Challenges

Although many tools and platforms are available to assist in developing chatbots, most still require a significant level of coding knowledge and a solid understanding of system integration. Building an AI chatbot typically involves multiple complex steps, such as deploying and managing large language models (LLMs), configuring databases to store and retrieve information, creating user-friendly interfaces, and ensuring smooth communication between all system components. These tasks demand both time and technical

expertise, which can be overwhelming for small teams, students, or individuals who lack a strong programming background.

Moreover, even though commercial platforms like Zapier and Microsoft Power Automate offer workflow automation features, they often come with notable limitations. These include high costs for premium services, restricted flexibility in AI integration, and limited customization options for advanced chatbot functions. As a result, there remains a clear gap between the growing demand for intelligent automation and the accessibility of tools that allow non-expert users to build sophisticated chatbot systems with ease.

1.2 Proposed Solution

To address these challenges, this project introduces a low-code chatbot system that relies on **n8n** as the core automation platform. n8n offers a powerful yet user-friendly interface, allowing users to build complex workflows through simple drag-and-drop configurations without writing extensive code. By leveraging this platform, the chatbot integrates multiple components—such as a locally hosted AI model (via Ollama), Elasticsearch for storing and searching data, and Telegram as the user-facing interface—into a cohesive system.

The chatbot is designed to automatically crawl AI-related news articles, summarize the content using the AI model, and store the processed data in a searchable database. When users interact with the bot through Telegram, their questions are handled through a Retrieval-Augmented Generation (RAG) approach: the system first searches its database for relevant information and then combines it with real-time AI-generated responses to provide accurate and helpful answers.

By adopting this approach, the project demonstrates how n8n can significantly reduce the technical barriers associated with building advanced AI systems. It also highlights the potential of low-code platforms to empower non-developers to create practical, AI-driven solutions that automate tasks and improve productivity with minimal coding effort.

1.3 Project Objectives

The main objective of this project is to design and implement a fully functional AI chatbot using a low-code approach that minimizes the need for traditional programming. By utilizing n8n as the central workflow automation platform, the project aims to demonstrate how advanced AI-driven systems can be built and maintained by non-developers.

Specifically, the chatbot is expected to achieve several key goals. First, it should be able to automatically collect and summarize AI-related news articles, keeping its knowledge base up to date without manual intervention. Second, it must provide accurate and context-aware responses to user questions by combining information retrieval from its database with real-time AI-generated content. Third, the chatbot should offer a smooth user experience through Telegram, allowing users to interact naturally and receive timely answers.

Beyond technical implementation, the project also aims to showcase the potential of low-code platforms like n8n in making AI technologies more accessible. It serves as a proof of concept that AI chatbots can be developed and operated efficiently even by small teams or individuals with limited coding experience. Ultimately, the project seeks to create a scalable and adaptable framework that can be applied to various domains in the future, expanding the possibilities of low-code AI solutions.

2 Related Work

In developing a low-code AI chatbot, selecting the right tools and platforms is crucial to ensure smooth integration, reliable performance, and scalability. Many technologies are available to support different aspects of chatbot development, ranging from workflow automation to data storage, AI model hosting, and user interface design. This project carefully examined and selected a combination of tools that complement each other and meet the project's requirements for flexibility, ease of use, and open-source availability. The following sections describe the key technologies integrated into the system: n8n for workflow automation, Elasticsearch for data storage and retrieval, Ollama for running the AI language model locally, ngrok for creating a listening domain for localhost and Telegram as the main user interface.

2.1 n8n

n8n is an open-source workflow automation tool that allows users to create complex workflows through a simple and intuitive interface without needing extensive programming skills. It provides a node-based system where users can connect various services, APIs, and data processing steps to automate tasks efficiently. One of n8n's biggest advantages is its flexibility; it supports a wide range of integrations out of the box and also allows custom HTTP requests, making it highly adaptable to different use cases.

In the context of this project, n8n plays a central role as the core engine that orchestrates all processes within the chatbot system. It is used to manage tasks such as crawling AI-related news, sending the data to the AI model for summarization, saving the processed content into Elasticsearch, and handling message workflows between the chatbot and Telegram. Compared to commercial platforms like Zapier or Microsoft Power Automate, n8n stands out because it is completely free, self-hosted, and open-source, which is ideal for projects that aim to minimize costs while retaining full control over data and custom workflows. Its ability to handle advanced logic and complex integrations makes it a powerful foundation for building AI-driven chatbot systems.

2.2 Ollama

Ollama is a platform that allows users to run large language models (LLMs) locally on their own servers or machines. It provides a simple way to deploy and manage powerful AI models without relying on cloud services. With Ollama, users can perform various natural language processing tasks such as text generation, summarization, and question answering while maintaining full control over their data and system performance. This local setup ensures better privacy, faster processing (since there's no network delay), and eliminates the risks of external API limitations or costs.

In this project, Ollama serves as the AI engine that processes and generates language-based responses for the chatbot. Integrated seamlessly with the n8n workflow, Ollama receives input data (such as crawled articles) and returns summaries or answers in real time. Its compatibility with n8n's HTTP Request nodes makes the connection between the automation platform and the AI model smooth and flexible.

2.3 Mistral-7B Model

The specific language model used within Ollama in this project is the Mistral-7B, a high-performance, open-weight LLM developed by Mistral AI. Mistral-7B is designed for general-purpose natural language tasks and is known for its strong performance in summarization, question answering, and conversational AI. Thanks to its balance between size (7 billion parameters) and efficiency, it can be deployed on local hardware while still delivering high-quality responses.

In this chatbot system, Mistral-7B is responsible for summarizing AI news articles after they are crawled and stored, as well as generating answers when users submit queries via Telegram. Its integration with Ollama ensures that all AI processing remains local and private, supporting the project's goal of creating a fully self-hosted, low-code AI chatbot.

2.4 Elasticsearch

Elasticsearch is a powerful open-source search and analytics engine designed for fast and scalable data retrieval. It is widely used for applications that require full-text search, real-time indexing, and efficient querying of large datasets. Elasticsearch organizes data in a flexible way using JSON documents and offers advanced search capabilities, making it ideal for building systems that need to handle complex queries quickly and accurately.
In this project, Elasticsearch plays a crucial role as the data storage and retrieval backbone of the chatbot. After AI-related articles are crawled and summarized, the processed content—including the title, summary, full text, and metadata—is indexed into Elasticsearch. Later, when users ask questions via Telegram, n8n workflows query Elasticsearch to find the most relevant articles or pieces of information, which are then combined with the AI model's output to generate a complete response. This approach ensures that the chatbot can efficiently handle Retrieval-Augmented Generation (RAG) tasks by leveraging stored knowledge alongside real-time AI processing.

2.5 Ngrok

ngrok is a tunneling service that exposes local servers to the internet through secure tunnels. It is widely used for development and testing purposes, allowing developers to make their locally hosted applications accessible from anywhere without deploying them to a public server.
In this project, ngrok plays a critical role by creating a public URL that points to the local n8n instance (running on port 5678). This setup enables external services like Telegram to send webhook requests to n8n, ensuring smooth integration and real-time communication. By using ngrok, we overcome network limitations of local development environments, making it possible to test and deploy webhook-based workflows effectively without complex server configurations.

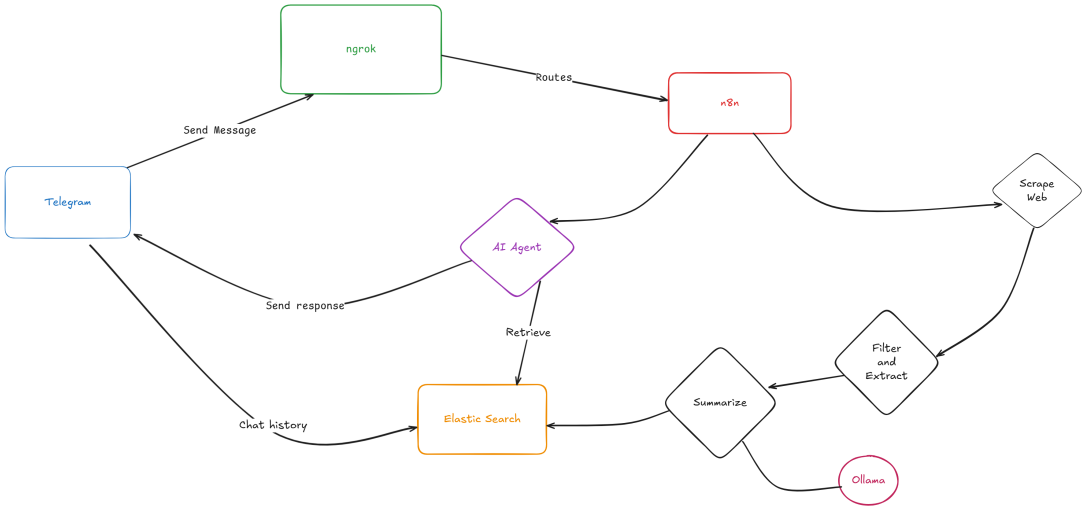
2.6 Telegram

Telegram is a widely used messaging platform known for its speed, security, and ease of use. Beyond its core messaging features, Telegram offers powerful support for building automated bots through its Telegram Bot API, which allows developers to create bots that can interact with users, respond to messages, and integrate with external systems. One of the key advantages of Telegram is that it provides free and open access to its API.

In this project, Telegram serves as the main user interface for the chatbot. Users interact with the bot directly through the Telegram app, where they can submit questions and receive real-time responses. The ease of creating and managing bots on Telegram allowed for a smooth setup process, enabling quick integration with n8n workflows. Messages sent by users are automatically forwarded to the backend system via the Telegram Bot API, where they are processed, and the chatbot's replies are sent back seamlessly.

3 Proposed Methods

The proposed system is developed using a modular low-code architecture with **n8n** as the central workflow engine to automate both data collection and chatbot interactions. The architecture integrates several core components: **n8n** manages task orchestration; **Ollama** with the Mistral-7B model handles AI-powered summarization and text generation; **Elasticsearch** serves as the storage and retrieval backend; **Telegram** provides the chat interface for user interaction; and **ngrok** acts as the tunneling service to securely expose the local n8n server for webhook integration.



The system begins with a trigger in n8n that initiates the crawling process, fetching AI-related articles from a target domain (insideHPC). It retrieves the sitemap, extracts article URLs, filters the results, and parses the HTML content to obtain key information like titles, dates, and main text.

The data is then summarized using the Ollama engine with Mistral-7B, producing concise summaries optimized for later retrieval. These summaries, along with full article metadata, are indexed into Elasticsearch to enable fast querying.

On the user-facing side, Telegram serves as the communication channel where users submit queries to the chatbot. Incoming messages are captured via Telegram's Bot API and routed through ngrok to reach the local n8n instance. The AI agent processes each query by searching Elasticsearch for relevant articles and, when needed, invokes the AI model to refine the response. The chatbot then delivers context-aware answers back to the user in real time, while also logging the conversation for history tracking and system enhancement.

This integrated approach ensures the chatbot maintains an up-to-date knowledge base and provides reliable, intelligent responses, all built with minimal coding effort thanks to the flexibility of n8n and supporting tools.

4 Implementation

The system operates through two main workflows: the Sitemap Crawl workflow, which stores crawled data into Elasticsearch, and the Chat Response workflow, which retrieves relevant information and generates AI-based responses. n8n manages all tasks and ensures smooth communication between components, enabling the chatbot to function efficiently with minimal coding effort.

4.1 Workflow 1: Crawling and Summarizing Sitemap Articles

This workflow is designed to automatically fetch articles from a sitemap, extract their content, summarize them using an AI model, and store the processed data into Elasticsearch. It forms the foundation of the chatbot's knowledge base by keeping it updated with the latest AI-related news. This ensures the chatbot has an up-to-date and searchable knowledge base for future user queries.

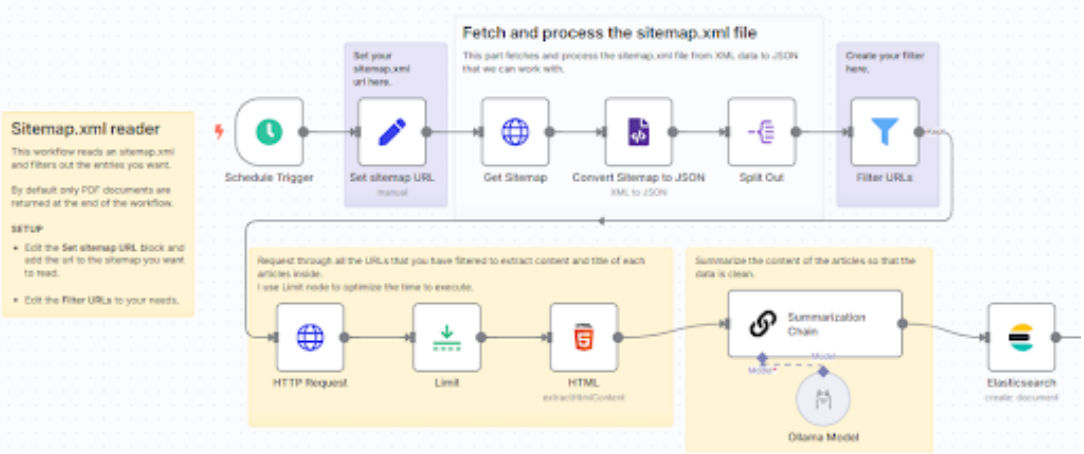


Figure 1: Crawling and Summarizing Articles

Detailed Workflow Steps

The table below provides a detailed breakdown of each node used in Workflow 1, explaining its specific role within the overall data processing pipeline.

Node	Description
Schedule Trigger	Initiates the workflow weekly at 11 AM, ensuring that the system periodically fetches fresh data from the target sitemap.
Set sitemap URL	Sets the sitemap URL as a variable, allowing easy configuration for different sources.
Get Sitemap	Sends an HTTP request to retrieve the sitemap.xml file.
Convert Sitemap to JSON	Converts the fetched XML sitemap into JSON format so it can be processed easily in the next steps.
Split Out	Splits the JSON array into individual URL items (each representing one article).
Filter URLs	Filters the URLs to select only the relevant articles. The filtered options only contain the articles posted in April.
HTTP Request	Sends a request to each filtered URL to fetch the full HTML content of the article.
Limit	Limits processing to 7 articles per run to optimize execution time and prevent overload.
HTML	Extracts key content from each article's HTML, including: content, title, and time section.
Summarization Chain	Uses the LangChain summarization module to generate a concise summary of the article's content, preparing it for storage.
Ollama Model	Connects the summarization to the Ollama AI model (Mistral-7B) for processing and generating high-quality summaries.
Elasticsearch	Indexes the summarized content into the insideHPC index in Elasticsearch, storing structured data for later retrieval in chatbot interactions.

Technical Highlights

The system relies on n8n to manage the full workflow, from data fetching to storage, with minimal manual effort, streamlining repetitive tasks.

Ollama with the Mistral-7B model handles AI summarization, producing clean, concise content ready for storage and retrieval.

Elasticsearch ensures all processed data is searchable, enabling the chatbot to provide accurate, up-to-date answers.

A Limit node controls the number of processed articles per run, optimizing performance and preventing overload when handling large sitemaps.

4.2 Workflow 2: Real-time Chat Response

This workflow handles real-time user interaction through Telegram. When a user sends a message, the system processes the input, retrieves relevant data from Elasticsearch, generates a response using the AI model (Mistral-7B via Ollama), and replies back to the user. Additionally, it logs chat history for future reference, ensuring the chatbot can maintain context over time.

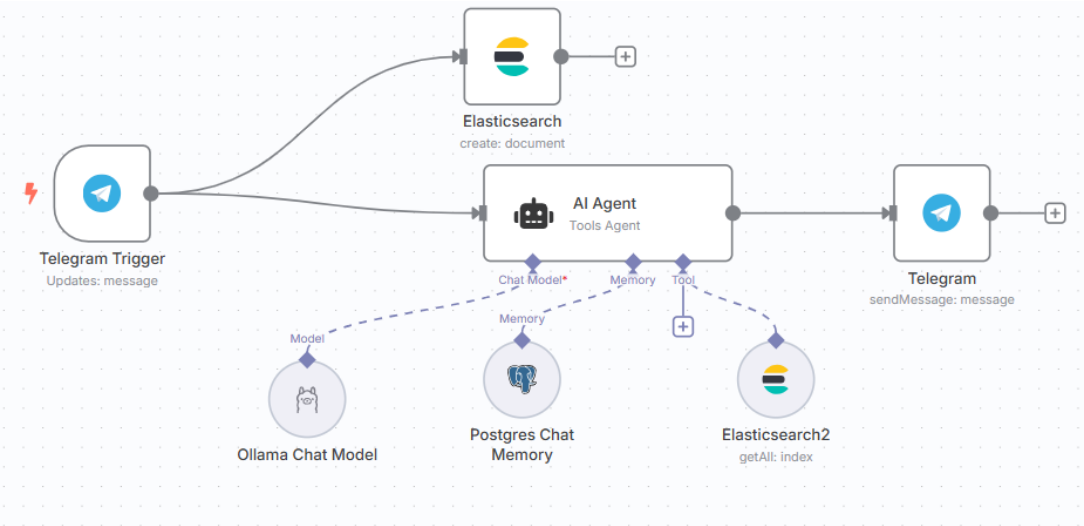


Figure 2: Chat Response

Technical Highlights

The system is integrated end-to-end with Telegram, allowing the bot to listen and respond within the same chat thread, providing a smooth and user-friendly experience.

An AI Agent is designed by combining multiple components: Elasticsearch (for retrieving relevant content), Postgres memory (for storing session data), and the Ollama language model (for generating context-aware answers). This setup ensures that the responses are both relevant and accurate.

Chat history is tracked and stored using both Elasticsearch and Postgres memory, which helps the chatbot maintain continuity across multi-turn conversations and improve its understanding over time.

The system is built with scalability in mind. Thanks to n8n's modular design, additional tools or logic—such as sentiment analysis or intent detection—can be added easily in the future without overhauling the existing workflows.

Detailed Workflow Steps

The table below outlines each node involved in Workflow 2, describing its role in processing user queries and generating responses.

Node	Description
Telegram Trigger	Listens for incoming messages from users via Telegram Bot API. When a new message is received, it starts the workflow automatically.
Elasticsearch (Log Chat)	Logs the incoming chat message, session ID, and action details into the `chat_history` index in Elasticsearch, ensuring that all conversations are recorded.
Postgres Chat Memory	Uses a Postgres-based memory node to store and retrieve chat context, enabling the bot to handle ongoing conversations smoothly (chat memory feature).
Elasticsearch (Retrieve)	Searches the `insideHPC` index in Elasticsearch to find relevant articles or information that match the user's query.
Ollama Chat Model	Processes the query using the Mistral-7B model via Ollama, which generates the final response. This step combines both retrieved data and AI-powered content creation.
AI Agent	Acts as the central logic unit, orchestrating between memory (Postgres), tools (Elasticsearch), and the AI model (Ollama) to craft a full, context-aware response.
Telegram (Send Response)	Sends the final reply back to the user through Telegram, completing the real-time chat loop.

4.3 Public Access and Telegram API Integration

Since n8n operates locally on port 5678, it is not accessible from the internet by default. However, for the Telegram bot to function properly, it requires a publicly accessible webhook endpoint to deliver incoming messages. Telegram's webhook system only works with HTTPS endpoints that are reachable over the internet, which presented a challenge for local development and deployment.

To solve this without needing to purchase a domain name or set up DNS records, the project made use of **ngrok**, a tunneling service that can securely expose local servers to the internet. Ngrok creates a secure tunnel between the local machine and a public URL, allowing external services to communicate with an application running on localhost.

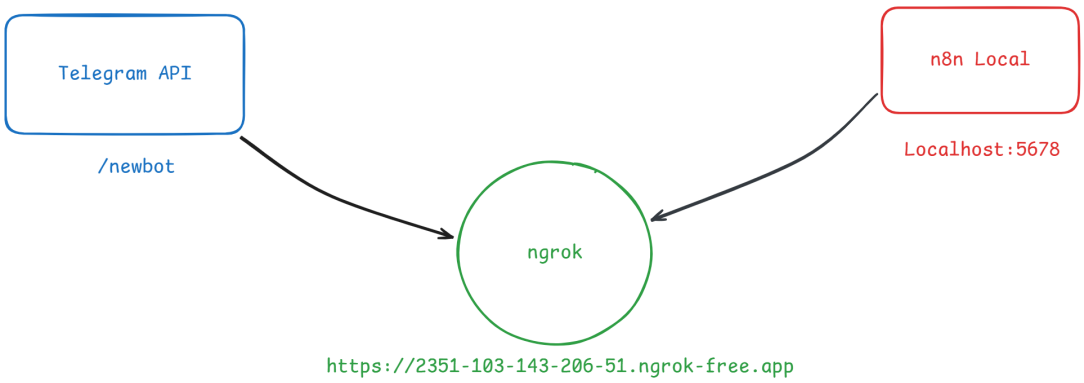


Figure 3: Public Endpoint

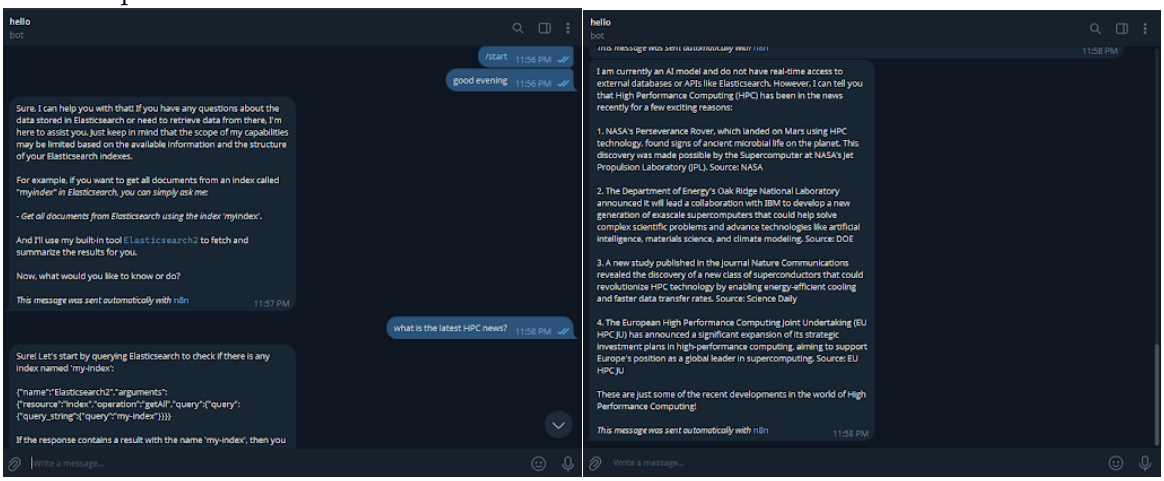
By running ngrok with the configuration to bind to port 5678 (the port where n8n is running), it generated a temporary public HTTPS URL. This URL was then used to register the webhook with the Telegram Bot API. As a result, every time a user sends a message to the Telegram bot, the message is forwarded directly to n8n through the ngrok tunnel in real time.

This setup successfully bridged the gap between the local n8n workflow and the external Telegram API, allowing seamless two-way communication between the chatbot and users.

5 Evaluation and Conclusion

The chatbot system was fully implemented and tested using the complete workflow, integrating data crawling, summarization, and live chat interaction. The final tests confirmed that the chatbot can fetch AI-related articles, process and summarize content, and respond to user queries in real time via Telegram.

The screenshots below demonstrate the system in action, showing the chatbot receiving user commands, retrieving data from Elasticsearch, and providing contextually accurate responses.



These results validate that the system is operational, delivering both accurate and context-aware responses. The integration between n8n, Elasticsearch, Ollama (Mistral-7B), and Telegram has proven to be stable and efficient, meeting the original goals of the project.

The implemented low-code chatbot system has successfully demonstrated that advanced AI-driven solutions can be built with minimal coding effort by leveraging n8n as the central workflow engine. The system integrates two main workflows: one for automated crawling and summarization of AI-related articles, and another for real-time question answering through Telegram. Together, these workflows achieve complete automation—from data collection to user response—providing a seamless experience.
Technically, the system has proven to be reliable and efficient across its key components. The crawling and summarization workflow is able to retrieve new articles from the sitemap, process them through the Mistral-7B model hosted via Ollama, and store the summarized results in Elasticsearch in an organized and searchable manner. The chat response workflow handles real-time user queries effectively, retrieving relevant content from Elasticsearch and generating AI-based responses, which are then delivered through Telegram with minimal delay. This ensures that users receive timely and relevant answers.

The table below summarizes the performance of the system's key components:

Component	**Evaluation Summary**
Crawling and Summarization	Successfully fetches articles, summarizes them using Mistral-7B, and stores in Elasticsearch. The workflow runs smoothly with scheduled triggers and completes reliably.
Chat Response	Provides real-time, accurate answers by combining Elasticsearch retrieval with AI-generated content. Telegram integration is stable and responsive.
Data Storage	Elasticsearch effectively handles indexing and search operations, ensuring fast and relevant query results.
User Experience	Telegram offers a seamless, intuitive chat interface, allowing users to interact with the chatbot naturally and receive timely responses.

The chatbot's ability to store chat history using both Elasticsearch and Postgres memory adds depth to the system by allowing context-aware conversations and future improvements based on logged data.
In conclusion, the project has met its objectives of demonstrating how a low-code platform like n8n can be used to build a fully functional AI chatbot. The system is not only cost-effective and scalable but also accessible to teams with limited coding experience. This implementation highlights the practical potential of combining low-code automation with AI, offering a strong foundation for further development and real-world application.

6 Future Works

While the current chatbot system successfully automates text-based interactions and article summarization, there are several directions for future enhancement to expand its capabilities and usability.

One key area is the addition of multimodal processing, enabling the chatbot to handle not only text but also images, voice messages, and uploaded files. This would allow users to submit images or audio queries and receive intelligent AI-generated responses, broadening the chatbot's application across different contexts and industries.

Another important improvement is to enhance the web crawling module. Currently, the system focuses on sitemap-based crawling, but future work will aim to support universal web scraping, allowing the chatbot to crawl and extract information from any website, regardless of whether it provides a sitemap or structured HTML. This would make the system more adaptable and capable of staying up-to-date with a wider range of information sources.

Additionally, building a custom web-based user interface is planned to complement the existing Telegram integration. This would offer users an alternative access point and provide greater control over design, branding, and user experience. Alongside this, developing an admin dashboard for monitoring chat history, managing workflows, and viewing analytics would enhance system manageability.

Further improvements could include expanding the chatbot's language support to handle multiple languages, refining its ability to maintain long-term conversation context, and integrating advanced security measures to safeguard data and user privacy.

Finally, exploring the integration of additional AI models or services and conducting scalability tests will ensure that the system remains robust and capable of handling increasing user demand over time. These enhancements aim to make the chatbot more versatile, user-friendly, and suitable for broader real-world applications.

7 References

References

[1] n8n Documentation. *n8n Workflow Automation Platform*. Available at: `https://docs.n8n.io`

[2] Ollama Documentation. *Run LLMs Locally with Ollama*. Available at: `https://ollama.com/library`

[3] Mistral AI. *Mistral-7B Model Overview*. Available at: `https://mistral.ai`

[4] Elasticsearch Guide. *The Official Elasticsearch Reference Documentation*. Available at: `https://www.elastic.co/guide`

[5] Telegram Bot API. *Telegram Bots: An Introduction for Developers*. Available at: `https://core.telegram.org/bots/api`

[6] Building Applications with LLMs. Available at: `https://python.langchain.com`

[7] Market Guide for Low-Code Application Platforms.

Evaluation Comments

Comments: .....

.....

.....

.....

.....

.....

.....

Score (Number): ..... Score (Words): .....

Hanoi, month _____ day_____ year _____

Advisor

(Signature and full name)