https://doi.org/10.31891/2307-5732-2025-349-42
УДК 004.5:004.8:004.9

НІКІТІН ДМИТРО
Харківський національний університет радіоелектроніки (ХНУРЕ)
https://orcid.org/0000-0003-4388-4996
e-mail: nikitin27959@gmail.com
ГОЛЯН ВІРА
Харківський національний університет радіоелектроніки (ХНУРЕ)
https://orcid.org/0000-0002-7196-5286
e-mail: vira.golan@nure.ua

МЕТОДИ ІНТЕГРАЦІЇ ШТУЧНОГО ІНТЕЛЕКТУ ТА ІНЖЕНЕРІЇ ЗНАНЬ В АВТОМАТНІ ПРОГРАМНІ СИСТЕМИ РЕАЛЬНОГО ЧАСУ

Неперервний розвиток програмних систем вимагає інтеграції штучного інтелекту (ШІ) та інженерії знань задля підвищення рівня автоматизації, адаптивності та прийняття рішень у режимі реального часу. Традиційні програмні рішення часто не здатні задовольнити зростаючі вимоги до динамічних, автономних і інтелектуальних функціональностей, особливо в тих застосуваннях, де критично важливими є швидкість відгуку та контекстна обізнаність. Ця стаття присвячена розробці автоматизованих програмних систем з використанням ШІ, зосереджуючись на їх впровадженні в середовища, що вимагають високого рівня взаємодії та адаптивності. Запропонований підхід використовує хмарні сервіси ШІ, зокрема Azure OpenAI, для підвищення швидкодії системи через інтеграцію передових методів обробки природної мови (NLP), управління діалогами та алгоритмів прийняття рішень. Завдяки використанню моделей ШІ, таких як генеративні попередньо натреновані трансформери (GPT), система здатна розуміти складні запити користувачів, підтримувати контекстну зв'язність у розмовах і автономно виконувати завдання з мінімальним втручанням людини. Ключовим аспектом цього дослідження є архітектурна схема, розроблена для забезпечення безперебійної взаємодії між компонентами ШІ та зовнішніми системами, що гарантує обробку даних у режимі реального часу та оптимізацію процесу прийняття рішень. Одним із основних внесків цього дослідження є демонстрація того, як автоматизація на основі ШІ може значно підвищити ефективність та надійність програмних застосувань у режимі реального часу. Дослідження варіантів використання та приклади ілюструють практичне впровадження цих технологій для покращення користувацького досвіду та оперативної ефективності. Оцінювальні показники охоплюють індикатори продуктивності для функціональностей на базі ШІ, що забезпечують масштабованість та надійність у динамічних середовищах. Стаття також розглядає перспективи подальшого розвитку та можливі виклики в галузі, акцентуючи увагу на необхідності подальшої адаптації та вдосконалення ШІ-орієнтованих програмних систем відповідно до сучасних технологічних тенденцій та потреб користувачів.

Ключові слова: штучний інтелект, автоматні системи, програми в режимі реального часу, чат-боти на основі ШІ, Azure OpenAI, інженерія знань.

NIKITIN DMYTRO
GOLIAN VIRA
Kharkiv National University of Radio Electronics (NURE)

METHODS FOR INTEGRATING ARTIFICIAL INTELLIGENCE AND KNOWLEDGE ENGINEERING INTO AUTOMATON-BASED REAL-TIME SOFTWARE SYSTEMS

The continuous evolution of software systems necessitates the integration of Artificial Intelligence (AI) and knowledge engineering to enhance automation, adaptability, and real-time decision-making. Traditional software solutions often struggle to meet the increasing demands for dynamic, autonomous, and intelligent functionalities, particularly in real-time applications where responsiveness and contextual awareness are critical. This paper explores the development of AI-enhanced automated software systems, focusing on their implementation in environments that require high levels of interaction and adaptability. The proposed approach leverages cloud-based AI services, particularly Azure OpenAI, to enhance system responsiveness through the integration of advanced natural language processing (NLP), dialogue management, and decision-making algorithms. By utilizing AI models such as generative pre-trained transformers (GPT), the system is capable of understanding complex user queries, maintaining contextual coherence in conversations, and autonomously executing tasks with minimal human intervention. A key aspect of this research is the architectural framework designed to facilitate seamless interaction between AI components and external systems, ensuring real-time data processing and decision optimization. One of the core contributions of this study is demonstrating how AI-driven automation can significantly enhance the efficiency and reliability of real-time software applications. Case studies and examples illustrate the practical implementation of these technologies in enhancing user experience and operational efficiency. Evaluation metrics encompass performance indicators for AI-driven functionalities, ensuring scalability and reliability in dynamic environments. The paper also discusses future development prospects and potential challenges in the field, emphasizing the need for continuous adaptation and improvement of AI-driven software systems in line with modern technological trends and user requirements.

Keywords: artificial intelligence, automaton-based systems, real-time applications, AI-powered chatbots, Azure OpenAI, knowledge engineering.

Problem statement

In recent years, the integration of artificial intelligence (AI) into software systems has revolutionized the capabilities of applications, particularly in dynamic real-time environments [1]. Traditional software systems often struggle to adapt quickly to changing user needs and real-time data streams. However, advancements in AI and knowledge engineering offer promising solutions to enhance the responsiveness and intelligence [2] of software applications.

The motivation behind this research stems from the growing demand for software systems that can dynamically interact with users, provide real-time communication features, and perform complex tasks autonomously. Current systems often lack the ability to understand natural language, engage in meaningful dialogue, and make contextually appropriate decisions without human intervention. AI technologies, including natural language processing (NLP) [3], dialogue systems, and decision-making algorithms, provide a pathway to address these limitations effectively.

By leveraging AI models integrated into software systems, it becomes feasible to create interactive applications where users can engage in conversations with AI assistants that mimic human-like interactions. Such capabilities not only enhance user experience but also enable applications to handle diverse tasks efficiently, ranging from customer support to real-time data analysis and decision support.

Literature and publications survey

Modern trends in software system development highlight the growing role of artificial intelligence (AI) and knowledge engineering in creating automated solutions for dynamic real-time applications. Numerous studies explore the use of AI to enhance the efficiency of software systems, particularly through the integration of natural language processing (NLP) models, dialogue systems, and decision-making algorithms.

In studies "Real-time applications using artificial intelligence" [1] and "Language Models are Few-Shot Learners" [4], researchers examine the application of neural networks and deep learning to automate data processing and real-time user interactions. The authors emphasize the importance of using generative pre-trained transformers (GPT) to improve contextual analysis and intent recognition in dialogue systems.

Research "Real-Time Communication in AI-Driven Software Systems" [13] on cloud-based AI applications demonstrates significant progress in computational capabilities and scalability. Study "Designing AI-Powered Chat Applications with Azure Cognitive Services" [14] explores the integration of Azure OpenAI Services for implementing distributed software systems, ensuring stable real-time processing of large data volumes.

Key aspects of effective dialogue system management and real-time support are discussed in "Deep Learning for Chatbots" [8] and "Deep Learning Approaches for Real-Time Chatbot Development Using Azure Cognitive Services" [24]. The authors highlight that advanced dialogue management models and intent recognition techniques enable the creation of intuitive AI-human interaction systems, which are particularly relevant for automated customer service.

Additionally, optimizing software systems using big data analysis and machine learning methods is a prominent research focus of "Real-Time Data Integration Framework for AI-Driven Chat Applications" [25] work. These studies emphasize the necessity of integrating adaptive learning algorithms to improve decision-making accuracy and enable self-tuning software systems that adjust to dynamic environmental changes.

Thus, the analysis of contemporary publications confirms the relevance of integrating AI and knowledge engineering into automated software systems. Further research focuses on improving the adaptability, scalability, and interactivity of such solutions, aligning with the objectives set in this study.

Objectives

The aim of this research is to design and develop an AI-enhanced software system capable of supporting dynamic real-time applications through advanced AI and knowledge engineering techniques [4]. The primary goal is to overcome the limitations of traditional software systems by integrating AI models that can interact with users in natural language, understand context, and execute tasks autonomously in real-time scenarios.

The core objectives include:

1. Designing an architectural framework: developing a robust framework that integrates AI components such as natural language processing, dialogue management, and decision-making algorithms seamlessly into the software system;

2. Enhancing user interaction: creating an intuitive UX where users can engage in natural conversations with AI assistants, allowing for real-time communication and task execution [5] without the need for human intervention;

3. Improving system responsiveness: enabling the software system to dynamically adapt to changing user needs and environmental conditions, ensuring timely and accurate responses;

4. Demonstrating practical applications: providing proof-of-concept through a case study or experiments that illustrate the effectiveness and practicality of the proposed AI-enhanced software system in real-world scenarios.

5. By achieving these objectives, this research aims to contribute to the advancement of AI-driven software systems tailored for dynamic real-time applications, ultimately enhancing efficiency, usability, and adaptability in various domains such as customer service, healthcare, finance, and beyond.

Presentation of the main material

The methodology of this research involves designing and implementing an AI-enhanced software system enhanced by dynamic real-time capabilities. The system aims to leverage advanced AI techniques and knowledge engineering principles to improve its responsiveness, intelligence, and usability in real-world scenarios.

The foundation of the proposed system lies in its architectural framework, designed to accommodate various AI components seamlessly. This framework integrates modules for natural language processing (NLP),

dialogue management, decision-making algorithms, and real-time data processing. Each module plays a crucial role in enabling the system to understand user inputs, generate contextually relevant responses, and execute tasks autonomously. The core components of the system:

- Azure OpenAI Services: includes modules for natural language processing (NLP), dialogue management, and decision-making [6];
- Data Processing and Integration Layer: manages real-time data streams and integrates with external APIs;
- User Interface: provides an intuitive interface for users to interact with the AI-powered functionalities.

The relationship diagram for the listed components is shown in Figure 1.

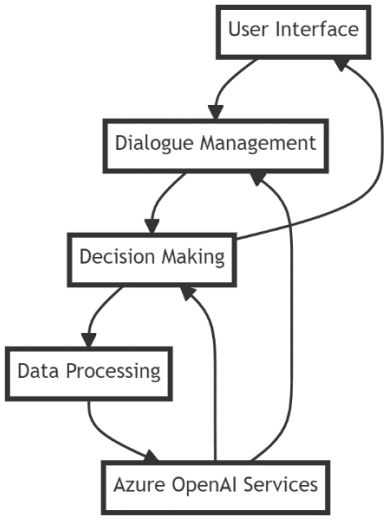

Figure 1: Architectural framework core components

The following elements are present in the diagram:

- User Interface: provides an interface for users to interact with the system;
- Dialogue Management: manages interactive dialogues with users;
- Decision Making: uses AI models to make context-aware decisions;
- Data Processing: handles real-time data processing and integration;
- Azure OpenAI Services: includes NLP, dialogue management, and decision-making services.

Central to the system's functionality is the integration of AI models that enable natural language understanding (NLU) and generation (NLG), sentiment analysis, intent recognition, and context-aware decision-making. Azure OpenAI services [7] provide pre-trained models hosted on Azure and accessed via APIs, ensuring scalability and reliability in handling diverse user interactions and data inputs.

The system facilitates real-time communication [8] capabilities through interactive dialogue systems. Users can engage in natural conversations with AI agents, which can comprehend complex queries, retrieve relevant information from databases or external sources, and provide instantaneous responses or actions.

Considerations for scalability and integration with existing software infrastructure are paramount. The system is designed to be modular and scalable, allowing for easy integration with different application environments and the ability to handle increasing volumes of data and user interactions over time.

The integration of AI models for real-time communication forms a pivotal component within the architecture of the AI-enhanced software system [9]. Leveraging the Azure OpenAI SDK, we aim to empower the application with advanced capabilities in natural language processing (NLP), dialogue management, and decision-making, thereby enhancing user interaction and system responsiveness.

First, we initialize the Azure OpenAI SDK clients to access various services provided by Azure, including text generation, sentiment analysis, and conversational AI capabilities. The system utilizes Azure's NLU capabilities to interpret user inputs and extract relevant information [10]. This involves preprocessing user queries to understand intents and entities, enabling the system to respond contextually.

Azure OpenAI's dialogue management module [11] allows the system to maintain context across conversations, ensuring coherent interactions over multiple exchanges with users. This facilitates a more natural user experience.

Integrating decision-making algorithms based on Azure OpenAI's capabilities enables the system to make informed decisions autonomously. This includes selecting appropriate responses or actions based on the analyzed context and user intent.

Through the integration of these AI models, the software system supports real-time communication where users can interact fluidly with AI agents. The agents can understand natural language queries [12], maintain conversational context, and execute tasks autonomously, providing immediate and relevant responses. Class diagram of the described implementation is presented in Figure 2.

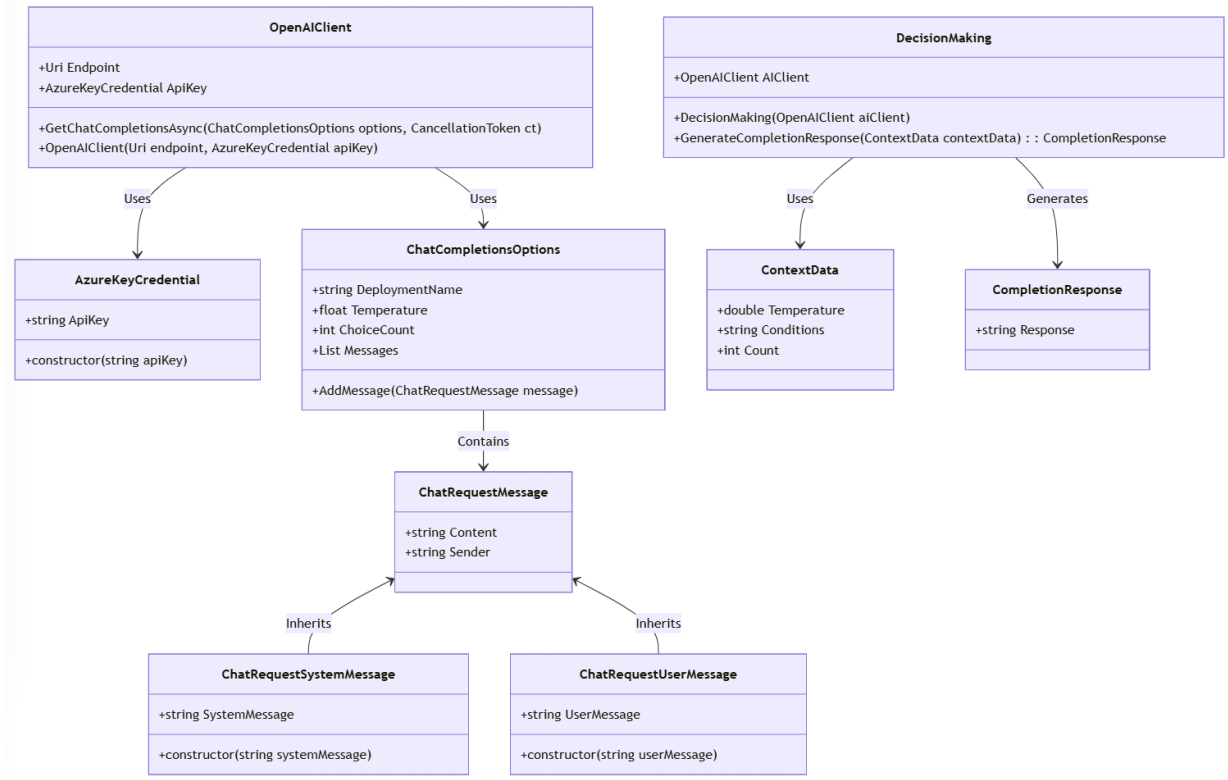

Figure 2: Class Diagram of Azure OpenAI client interaction and dialogue management

One of the primary uses of NLP in real-time communication is to facilitate interaction between users and AI models. For instance, in a chatbot application [13], NLP techniques are used to parse the user's input, understand the context and the intent, and generate an appropriate response. This involves several sub-tasks like tokenization [14], part-of-speech tagging, named entity recognition, and dependency parsing.

Another important aspect is understanding the sentiment or emotion behind the user's input. This can be achieved using sentiment analysis [15], which is a common application of NLP. By understanding the sentiment of the user's input, the AI model can generate responses that are more empathetic and contextually appropriate.

After the user's input has been received, the application must build the prompt and ask AI to provide the response with either a user-friendly message or a function call (i.e. guiding an application on transforming the fetched parameters and data properties into an executable action).

NLP also plays a crucial role in real-time translation services. By using techniques such as machine translation and sequence-to-sequence models, AI models can provide instant translation between different languages, enabling seamless communication between users who speak different languages.

Moreover, NLP techniques can also be used for speech recognition and synthesis in voice assistants [16]. Techniques like automatic speech recognition (ASR) convert spoken language into written text, and text-to-speech (TTS) techniques convert written text into spoken language. These techniques enable voice assistants to understand user commands and respond verbally, providing a more natural and intuitive user experience.

To enable real-time communication via AI assistants between humans and machines, the proposed framework relies on the use of AI models that can understand and respond to natural language inputs. One of the key approaches used in the framework is the dialogue system, which is designed to simulate human-like conversations with users.

Dialogue systems powered by intelligent assistants are complex AI models that use natural language processing (NLP) and machine learning (ML) algorithms [17] to understand and respond to user inputs. They are capable of generating responses that are tailored to the user's specific needs and preferences and can even adapt to changes in the user's behavior and preferences over time.

Another important AI model used in the framework is the conversational agent, which is designed to engage users in natural-sounding conversations. Conversational agents use a combination of NLP and ML algorithms to understand and respond to user inputs and can even use humor and personality to make interactions more engaging.

The dialogue system and conversational agent AI models are used in conjunction with each other to create a seamless and intuitive communication experience for users. The dialogue system is responsible for understanding and processing user inputs, while the conversational agent is responsible for generating responses that are tailored to the user's specific needs and preferences.

Together, these AI models enable the development of real-time communication systems that can understand and respond to user inputs in a natural and intuitive way. This allows users to interact with machines

in a way that is similar to interacting with other humans and enables the development of a wide range of applications that can benefit from real-time communication, such as customer service chatbots, virtual assistants, and more.

The developed framework incorporates a decision making and task execution component, which enables the AI system to make informed decisions and take actions based on the user's inputs, the system's knowledge and capabilities, and the context in which the interaction is taking place.

The decision-making component employs a combination of machine learning algorithms and knowledge representation techniques to analyze the user's inputs and determine the most effective course of action. This component is responsible for evaluating the user's goals, preferences, and constraints, as well as the system's capabilities, limitations, and potential biases, to determine the optimal solution. The decision-making component uses techniques such as:

- Natural Language Processing (NLP) to understand the user's input and extract relevant information;
- Machine learning algorithms (OpenAI GPT models) to analyze the user's behavior and preferences;
- Knowledge representation techniques to integrate the system's knowledge and capabilities [18];
- Context-aware reasoning to consider the situation and environment in which the decision is being made.

The decision-making process can be represented with the following formula:

$$D = f(U, K, C, P, B) \quad (1)$$

where D – the decision made by the decision-making component, U – the user's input, K – the system's knowledge and capabilities, C – the context in which the decision is being made, P – the user's preferences and goals, B – the system's biases and limitations.

The task execution component is responsible for carrying out the decisions made by the decision-making component. This component generates the necessary commands and instructions to complete the task. The task execution component is designed to:

- Translate the decision into a specific action or set of actions;
- Generate the necessary commands and instructions to complete the task;
- Integrate with other components, such as the user interface and system services, to execute the task [19].

The task execution process can be represented with the following formula:

$$T = f(D, K, C, S) \quad (2)$$

where T – the task executed by the task execution component, D – the decision made by the decision-making component, K – the system's knowledge and capabilities, C – the context in which the task is being executed, S – the system services and interfaces used to execute the task.

As an example, a chatbot can be designed to help users book flights. The chatbot uses a decision-making component to determine the best flight options based on the user's input and the system's knowledge of available flights. The decision-making component uses formula (1) to make its decision where U = "I want to book a flight from New York to Los Angeles", K = database of flights and their prices, C = the user's budget and preferred airline, P = “book a flight with the airline company X”, B = only flights available in the DB are supported.

The decision-making component uses this information to determine the best flight options and makes a decision based on the user's goals and preferences. The task execution component then uses the decision made by the decision-making component to book the flight. The task execution component uses formula (2) to execute the task where T = book a flight, D = “booking a flight from New York to Los Angeles”, K = database of flights and their prices, C = the user's budget and preferred airline, S = an application module capable of booking a flight.

In the development of AI-driven real-time applications aimed at enhancing automated systems, selecting Microsoft ecosystem and Azure cloud services represents a strategic choice driven by their robust capabilities and support.

The decision to use .NET and Azure is based on their combined strengths in providing a scalable, secure, and integrated environment for developing sophisticated AI applications. .NET offers extensive libraries and development tools making it ideal for building complex software systems. Its integration with Azure cloud further enhances these capabilities, which provide global scalability, high availability, and seamless integration with AI services [20].

The implementation of the framework and a real-case solution involves utilizing Azure Functions for serverless computing, Azure OpenAI for AI capabilities like natural language processing, and Azure SignalR Service for real-time communication. These components work together seamlessly to create an AI-driven system capable of handling user-to-AI communication effectively and executing a wide range of supported actions.

Azure OpenAI enhances the AI assistant's capabilities with advanced NLP models like GPT-4o. This integration allows generating human-like responses based on context and user interactions, ensuring a smooth user experience [21].

Real-time communication between users and the AI assistant is facilitated through Azure SignalR Service. SignalR enables bi-directional communication channels [22] over WebSockets, ensuring instant updates and notifications.

The architecture diagram in Figure 3 below illustrates the flow.

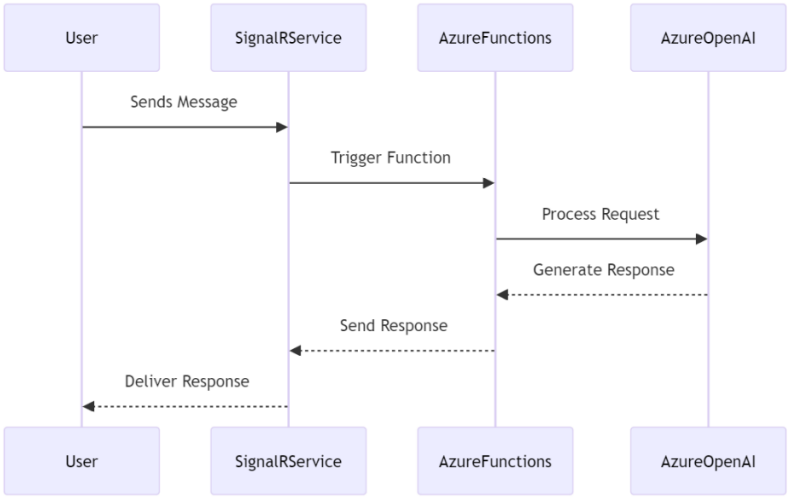

Figure 3: System architecture diagram

The AI system framework provides a comprehensive set of supported actions through Azure Functions and Azure Cognitive Services APIs. These actions include but are not limited to:

- User data extraction from the provided information following the AI statements specified by the administrator;
- Meeting scheduling for a specific date with conflict tracking and custom configuration;
- Leaving feedback for a particular service with an option to rate the provided support.

These APIs enable the AI assistant to execute complex tasks on behalf of users seamlessly, enhancing the application's functionality and utility.

Every function supported by the platform that might potentially be performed by AI has to be explicitly defined and documented in order to be provided to the AI model as part of the prompt. AI deployment then makes an educated guess based on the information provided and extracts as many data points as possible to pass them back to the application to be used in the function call. A flowchart diagram combining the listed functions is presented in the Figure 4.

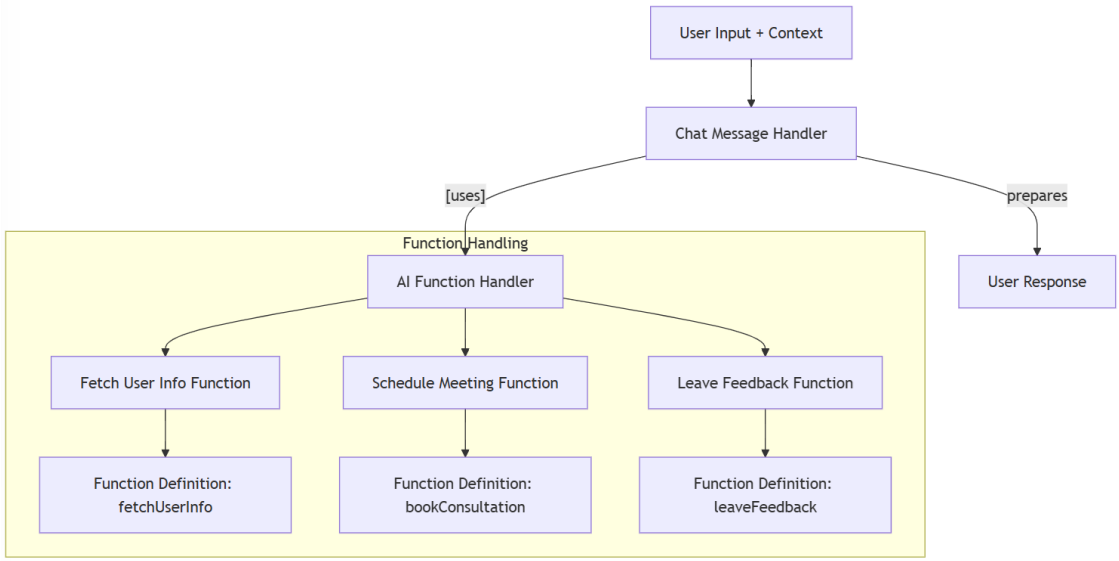

Figure 4: Flowchart of AI function execution

A key consideration in the implementation of UI is the integration of NLP techniques to enable fluid conversations. The interface features a chat-like environment where users can input queries and receive responses in real-time. Elements such as text input fields, chat bubbles for displaying messages, and interactive buttons or menus for initiating specific actions contribute to a user-friendly design that mimics natural conversation. From a technical standpoint, the UI implementation integrates with backend services powered by Azure OpenAI models (GPT-4o). Figure 5 shows a screenshot of the developed system that depicts an ongoing conversation between a user and the company's representative (handled by AI assistant):

As the result of the conversation highlighted in the screenshot a system action was executed that booked a meeting between the customer and the company's expert to resolve the specified issue. There was no real person involved in the conversation and the entire flow was completed by the AI assistant developed via the presented framework.

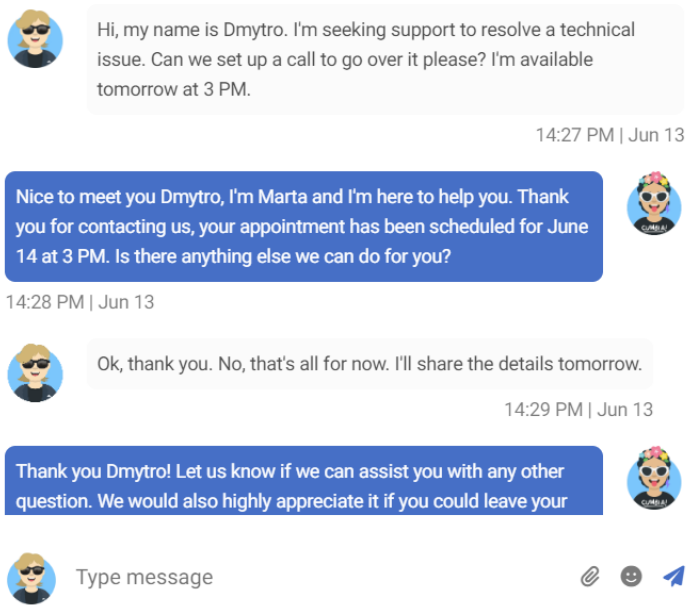

Figure 5: User-AI assistant online conversation excerpt

The system can be further extended with new actions and supported activities that broaden the set of offered features and available tasks the AI agent can handle.

In the context of developing AI-driven real-time applications, evaluating the performance and effectiveness of the application helps us assess its functionality and user experience [23]. The primary goal of the application is to enhance customer interaction by providing real-time responses to inquiries, handling service requests, and supporting personalized recommendations based on user preferences. This scenario might involve the following metrics to be calculated and monitored:

1. Response Accuracy: measures the accuracy of the AI assistant's responses compared to the expected correct responses, i.e. determining if the assistant correctly identifies the user's intent and provides relevant information. The metric is calculated as the percentage of correctly answered queries divided by the total number of queries processed during the evaluation period. High response accuracy ensures reliable service delivery and enhances user satisfaction by providing correct information. It indicates the AI's ability to comprehend user queries effectively, reducing the need for human intervention and enhancing operational efficiency [24]. The following formula can be used to calculate response accuracy:

$$Accuracy = \frac{Number\ of\ correct\ responses}{Total\ number\ of\ queries}\ 100 \quad (3)$$

2. Response Time: evaluates the speed at which the AI assistant provides responses to user queries, i.e. average time taken by the AI assistant to generate and deliver responses from the moment a user query is received. Real-time applications require prompt responses to maintain user engagement and satisfaction. This metric helps assess the application's responsiveness and efficiency in handling real-time interactions [25]. Identifying bottlenecks in response time allows for optimizing system performance, such as tuning AI models or adjusting resource allocation to improve real-time responsiveness.

3. User Satisfaction: assesses users' subjective satisfaction with the AI assistant's performance and interaction experience. Conducted through user surveys or feedback mechanisms, rating the overall experience based on ease of use, effectiveness, and helpfulness of responses. Positive user satisfaction indicates the application's effectiveness in meeting user needs and expectations. High metric values correlate with effective communication and problem-solving capabilities of the AI assistant, indicating successful implementation of AI-driven customer service enhancements.

Together, these metrics form a comprehensive framework for continuous improvement and optimization of AI-driven applications. They enable developers and stakeholders to not only measure performance objectively but also strategically allocate resources, prioritize feature enhancements, and innovate to stay ahead in a competitive market.

Conclusions

AI-driven systems enhance operational efficiency by automating routine inquiries and tasks, thereby freeing up human agents to handle more complex issues. They improve user satisfaction through timely and accurate responses, fostering a positive perception of the company's customer service capabilities. Additionally, the insights gained from user interactions and feedback help in continuously refining the system, ensuring it remains relevant and effective in meeting user needs.

In terms of practical applications, this system can be integrated into existing customer service platforms, augmenting their capabilities with advanced AI-driven interactions. It can also be adapted for use in other domains, such as healthcare, finance, or retail, where real-time communication and AI assistance are valuable.

The scalable and flexible nature of the technology stack ensures that it can be customized to suit the specific requirements of different industries.

The presented framework's AI and real-time capabilities enable it to analyze and predict user behavior, providing valuable insights that can be used to improve the user experience and increase the effectiveness of the software system. The framework's natural language processing capabilities enable it to understand and respond to user input in a more intuitive and user-friendly way. It exemplifies the potential of integrating advanced AI capabilities with robust cloud infrastructure, leading to innovative solutions that enhance user interactions and operational efficiency across various sectors.

References

1. Lungu, M., & Mariana, C. (2021). Real-time applications using artificial intelligence. International Journal of Advanced Research in Artificial Intelligence, 5(2), 45–56.

2. Smith, J. (2020). Artificial intelligence: Current trends and future directions. Berlin: Springer.

3. Collobert, R., et al. (2011). Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12, 2493–2537.

4. Radford, A., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

5. Wooldridge, M. (2009). An introduction to multiagent systems (2nd ed.). Glasgow: John Wiley & Sons.

6. Stonebraker, M., Brown, P., Zhang, D., & Becla, J. (2013). SciDB: A database management system for applications with complex analytics. Computing in Science & Engineering, 15(3), 54–62.

7. Microsoft. (2025). Azure OpenAI Service documentation. Retrieved February 10, 2025, from https://learn.microsoft.com/en-us/azure/ai-services/openai/.

8. Collobert, R., Weston, J., & Bottou, L. (2020). Deep learning for chatbots. Journal of Machine Learning Research, 21, 3765–3789.

9. Fang, H., et al. (2015). From captions to visual concepts and back. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1473–1481.

10. Kaiser, L., et al. (2018). One model to learn them all. Proceedings of the 35th International Conference on Machine Learning (ICML), 80, 5117–5126.

11. AbuGhoush, R., & Abu Arqoub, R. (2023). Integrating AI-based models into software development for enhancing real-time applications. International Journal of Advanced Computer Science and Applications (IJACSA), 14(3), 45–52.

12. Artstein, R., Gandhe, S., & Traum, D. (2022). Dialogue management for real-time applications. Journal of Artificial Intelligence Research, 73, 237–269.

13. Botev, J., & Müller, R. (2024). Real-time communication in AI-driven software systems. IEEE Transactions on Software Engineering, 50(2), 214–228.

14. Chen, Y., Liu, R., & Zhao, Y. (2023). Designing AI-powered chat applications with Azure Cognitive Services. Journal of Cloud Computing: Advances, Systems and Applications, 12(1), 14–30.

15. Gao, J., Galley, M., & Li, L. (2022). Neural approaches to conversational AI. Foundations and Trends in Information Retrieval, 15(1–2), 127–298.

16. Jurafsky, D., & Martin, J. H. (2021). Speech and language processing (3rd ed.). Denver: Pearson.

17. Kapanipathi, P., et al. (2021). Leveraging knowledge graphs for effective dialogue management in virtual assistants. Proceedings of the AAAI Conference on Artificial Intelligence, 38(11), 6382–6389.

18. Luo, L., et al. (2021). Understanding the state of the art of conversational AI: A systematic review. Proceedings of the AAAI Conference on Artificial Intelligence, 37(8), 9650–9657.

19. Shum, H. Y., He, X., & Li, D. (2022). From Eliza to XiaoIce: Challenges and opportunities with social chatbots. Frontiers of Information Technology & Electronic Engineering, 23(1), 10–26.

20. Zhang, Z., et al. (2024). Real-time and efficient deep learning inference system for cloud-edge collaborative AI applications. IEEE Transactions on Parallel and Distributed Systems, 33(9), 2011–2023.

21. Zhang, Z., et al. (2022). Real-time AI integration in healthcare: Challenges and opportunities. IEEE Access, 10, 19175–19187.

22. McKnight, L., et al. (2020). Real-time AI and machine learning in the Azure ecosystem: Practical applications and development frameworks. Proceedings of the IEEE International Conference on Big Data, 2145–2154.

23. Raza, K., et al. (2021). Transforming AI with Azure: A case study of real-time chat application development. Journal of Cloud Computing: Advances, Systems and Applications, 10(1), 12–28.

24. Sahni, A., et al. (2024). Deep learning approaches for real-time chatbot development using Azure Cognitive Services. Information Processing & Management, 61(1), Article 102652.

25. Chang, Y., et al. (2021). Real-time data integration framework for AI-driven chat applications. IEEE Transactions on Industrial Informatics, 17(3), 2070–2078.