http://doi.org/10.35784/iapgos.8526 *received: 07.10.2025 | revised: 24.02.2026 | accepted: 08.04.2026 | available online: 30.06.2026*

DESIGN OF DIGITAL COOKING ASSISTANT SYSTEM WITH MODERN VOICE GENERATIVE AI MODEL

Robert Banasiak, Zdzisława Rowińska, Wojciech Szczucki, Dawid Jantosz, Łukasz Rembowski
Lodz University of Technology, Faculty of Electrical, Electronic, Computer and Control Engineering, Lodz, Poland

Abstract. *This paper presents the design and implementation of digital cooking assistant that leverages modern generative AI models to enhance accessibility and inclusiveness in everyday cooking, named as "Cooking Master". The proposed system integrates a voice-based assistant capable of real-time dialogue, recipe personalization, and hands-free control of the application interface. By combining LiveKit for low-latency communication, Groq hardware acceleration, and the LLaMA language model, the assistant enables natural interaction through speech recognition and synthesis. Particular emphasis was placed on creating solutions that support disabled users, especially the visually impaired, for whom traditional recipe formats often pose significant barriers. The assistant provides step-by-step instructions, intuitive voice commands, and accessible navigation, allowing independent cooking without the need for touch-based interaction. The multilingual architecture further extends inclusivity, enabling users from diverse linguistic backgrounds to benefit from AI-driven assistance. Experimental results confirmed that the system responds with minimal latency and natural-sounding feedback, making it suitable for real cooking environments. The developed solution demonstrates the potential of generative AI in supporting not only convenience and personalization but also in promoting digital accessibility for users with special needs.*

Keywords: artificial intelligence, generative AI, accessibility, voice assistant

PROJEKT CYFROWEGO ASYSTENTA KULINARNEGO Z NOWOCZESNYM GENERATYWNYM MODELEM GŁOSOWYM AI

Streszczenie. *W artykule przedstawiono projekt i implementację cyfrowego asystenta kulinarnego wykorzystującego nowoczesne modele generatywnej sztucznej inteligencji w celu zwiększenia dostępności i inkluzywności gotowania, o nazwie "Cooking Master". Zaproponowane rozwiązanie integruje asystenta głosowego, który umożliwia prowadzenie dialogu w czasie rzeczywistym, personalizację przepisów oraz bezdotykowe sterowanie interfejsem aplikacji. Dzięki zastosowaniu technologii LiveKit zapewniającej niskie opóźnienia, akceleracji sprzętowej Groq oraz modelu językowego LLaMA, asystent umożliwia naturalną interakcję poprzez rozpoznawanie i syntezę mowy. Szczególną uwagę poświęcono wsparciu osób z niepełnosprawnościami, zwłaszcza niewidomych i niedowidzących, dla których tradycyjne formaty przepisów stanowią istotną barierę. Asystent dostarcza instrukcje krok po kroku, intuicyjne komendy głosowe oraz ułatwioną nawigację, umożliwiając samodzielne gotowanie bez potrzeby obsługi dotykowej. Architektura wielojęzyczna dodatkowo zwiększa inkluzywność, pozwalając na korzystanie z systemu użytkownikom o różnych potrzebach językowych. Przeprowadzone testy potwierdziły, że system zapewnia bardzo niskie opóźnienia i naturalne brzmienie odpowiedzi, co czyni go praktycznym rozwiązaniem w realnych warunkach kuchennych. Opracowany system pokazuje potencjał sztucznej inteligencji generatywnej nie tylko w zakresie wygody i personalizacji, ale również w promowaniu dostępności cyfrowej dla osób ze szczególnymi potrzebami.*

Słowa kluczowe: sztuczna inteligencja, AI generatywne, dostępność, asystent głosowy

Introduction

In recent years, the integration of artificial intelligence (AI) into everyday life has led to the development of increasingly sophisticated assistive technologies. The food industry is no exception, as AI has also been rapidly adopted to enhance various aspects of cooking and meal preparation. A prevailing number of research papers consists of AI-based recipe personalization [11], augmented reality (AR) cooking assistants focused on ingredients recognition [15], and tools that help manage time and dynamically adapt recipes to changing conditions [1]. However, there remains a noticeable gap in literature when it comes to voice-based cooking assistants. In the absence of substantial academic research work in this area, we explored existing real-world implementations. On the local polish market there is currently only one web service – przepisy.pl [18] which, beyond functioning as a recipe database, offers only very limited gesture- and voice-based guidance, restricted to simply moving forward or backward within a recipe. There are also popular cooking appliances on the international market, such as *Thermomix* [19] or *Upliance AI* [20], both of which are multifunctional smart cookers with a wide range of food preparation capabilities. However, only the latter integrates artificial intelligence into its features, offering an AI-powered chatbot that enhances the cooking experience.

Definition of the problem

Although all the mentioned examples are great inventions, there remains an underdeveloped area. A key limitation lies in the necessity of physically interacting with a touchscreen while cooking, which conflicts with the practical requirement of maintaining a hygienic environment. While the concept of voice-based guidance addresses this issue to some extent, it does not fully resolve the problem. An extended approach is required – one that allows users to operate entirely hands-free, going beyond simple forward and backward navigation between recipe steps.

Universal need for adapting

We live in the days where digital overload and shortened attention spans are increasingly common among young people, which makes it more important to cultivate healthy culinary habits early in life. Two foundational studies highlight that higher cooking competence in adolescence not only predicts healthier dietary behaviours that persist into adulthood [14] but is also associated with better psychological well-being [13]. Building on these insights, our initial concept was designed to support beginners in cooking, who often benefit from clear and intuitive guidance, while being encouraged by an AI-enhanced youth-friendly interface that engages their interest and fosters long-lasting, health-promoting behaviours.

Visually – impaired users

Beyond engaging young users, the potential of an AI-enhanced interface naturally extends to accessibility. For visually impaired individuals, cooking often presents significant barriers, as traditional recipe formats rely heavily on visual cues. By integrating voice interaction, auditory feedback, and complementary accessibility features such as ARIA properties or high-contrast display modes, the assistant can help overcome these obstacles. Combined, these tools create a more inclusive environment that enables independent navigation, recipe control, and overall participation in cooking for users with diverse abilities.

Multi-language users

In the same spirit, accessibility can also be expanded beyond physical or sensory needs to encompass linguistic diversity. Extending the assistant's functionality to multiple languages – rather than limiting it to two languages – would allow a wider range of users to benefit from intuitive, AI-driven cooking support.

1. System design

1.1. User-centred functional design

From the outset of our Cooking Master project, we positioned user needs as the primary factor guiding the system design. To identify these needs, we conducted a survey aimed at understanding which functionalities would be considered most valuable in the context of an AI-based cooking assistant.

Participants were asked to evaluate a set of proposed features using a five-point scale: very useful (blue), rather useful (orange), neutral (yellow), rather not useful (green), and not useful at all (purple). The presented functionalities included:

- AI assistant,
- voice control over the application,
- personalized recipes considering allergens,
- portion-based ingredient scaling,
- automatic shopping list generation,
- saving favourite recipes.

The survey results presented in Figure 1 indicate that most participants rated these features as very useful or rather useful. Consequently, all of them were incorporated into the proposed solution.

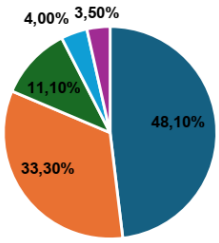

Fig. 1. User survey – perceived usefulness of a cooking voice assistant; very useful (blue), rather useful (orange), neutral (yellow), rather not useful (green), and not useful at all (purple)

1.2. Implementation

To ensure maximum accessibility, the system was designed as a web-based solution (HTML [2], CSS [16], TypeScript [7], React [6], Tailwind [12]). This approach was chosen for its convenience for both users and developers, as a single codebase can serve both desktop and mobile devices. While this design implies a minor performance trade-off compared to native implementations, it is negligible for the intended use case.

The application architecture relies on Docker [2] containers, separating the database, backend, and frontend components. All server requests are handled by a Flask-based Python server [9]. The entire solution was deployed to Google Cloud, making the application accessible to users through a public link.

1.3. User based recipes

Before accessing the full functionality of the application, each user is required to register. During the registration, users provide essential information, including:

- cooking experience,
- dietary preferences (e.g., vegan/vegetarian),
- allergy information,
- preferred cuisine region,
- recipe language,
- translation preferences,
- unit system preferences.

This information is stored in a structured MongoDB [8] database and is subsequently used to personalize the user experience. For instance, a vegan user will only be presented with vegan recipes, while a user with a walnut allergy will exclusively receive walnut-free recipes. These conditions are combined, ensuring that, for example, a vegan user with a walnut allergy only receives recipes satisfying both constraints.

Cooking experience also plays a role in tailoring content: beginner users are recommended simpler recipes, whereas experienced users are offered more advanced options.

Additionally, users can further refine their choices by filtering recipes by categories, calorie values, preparation time, and meal type. A favorite-marking feature allows users to store and easily access preferred recipes.

1.3.1. Additional functionality

The system also provides functionality for adjusting ingredient quantities based on the number of servings specified by the user. All ingredient amounts are automatically scaled up or down according to the chosen portion size.

Based on selected recipes presented in Figure 2, users can generate shopping lists, which can be exported in PDF, text, or JPG format.

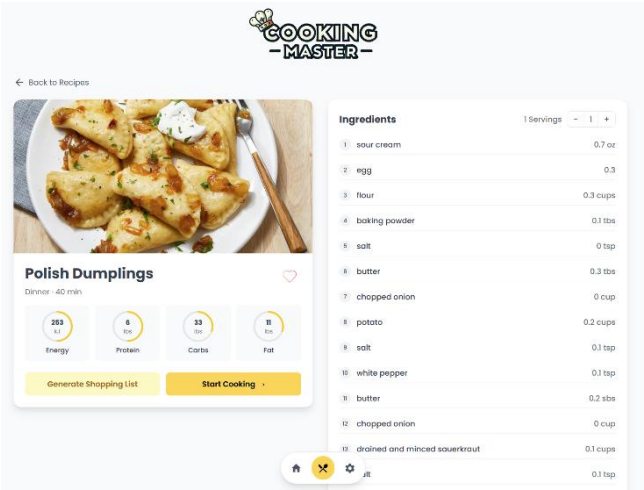

Fig. 2. Servings calculations and shopping list generation

The application was designed with multilingual support in mind. Currently, Polish and English are available. Thanks to the adoption of the i18n [4] standard, adding new languages is straightforward and requires only the addition of JSON files with translated content. Users can switch between languages via the application settings, instantly adapting the interface.

1.3.2. Voice assistant and voice control

For the voice assistant functionality, LiveKit [5], Groq [3], and the LLaMA [16] language model were integrated. This combination enabled seamless communication between LiveKit (handling real-time voice interaction) and LLaMA (providing natural language understanding and generation). The assistant presented in Figure 3 was equipped with context-specific prompts, ensuring that responses were relevant to cooking-related scenarios.

The assistant supported both content-related and interface-related queries, such as:

- providing detailed recipe descriptions,
- explaining step-by-step instructions,
- controlling the application (e.g., adjusting volume, starting/stopping video playback).

In all tested scenarios, the assistant responded with natural-sounding synthesized speech was able to control the application interface without noticeable delays.

Comprehensive technical details of the AI are discussed in Chapter 2 of this paper.

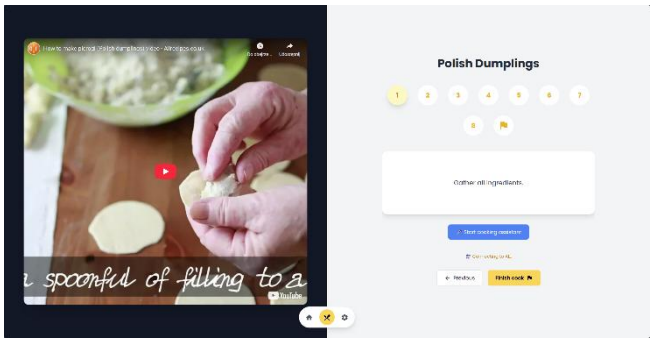

Fig. 3. Voice assistant and voice control view

1.4. Communication between containers – diagram

The containers interact through a microservice network that connects the frontend, backend, and databases. The React-based frontend communicates with the Flask server, which coordinates requests between services. PostgreSQL [10] stores structured data such as user accounts, while MongoDB manages unstructured data including recipes. The AI assistant container integrates with both databases to provide personalized responses. Management interfaces for PostgreSQL and MongoDB are isolated from user-facing services, ensuring controlled and secure data flow.

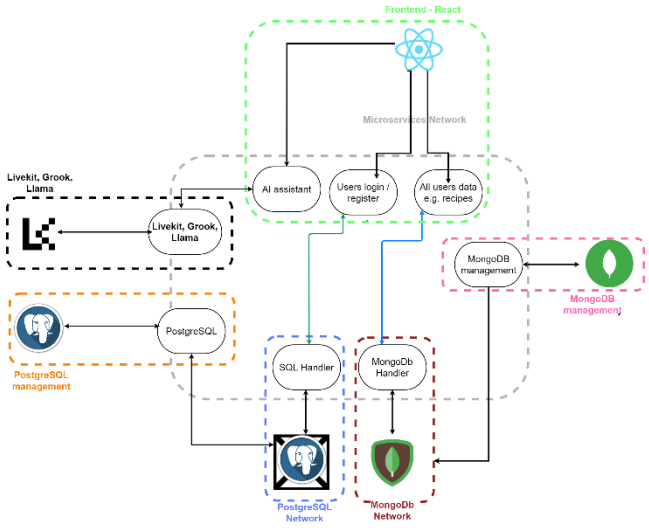

Fig. 4. Communication between containers

Figure 4 presents the modular system architecture of the platform. The design follows a microservice-based approach, which ensures scalability, flexibility, and maintainability. As shown in the diagram, containers interact through a service network that connects the frontend, backend, and databases. The React-based frontend communicates with the Flask server, which coordinates requests between services. PostgreSQL [17] stores structured data such as user accounts, while MongoDB manages unstructured data including recipes. The AI assistant container integrates with both databases to provide personalized responses. Management interfaces for PostgreSQL and MongoDB are isolated from user-facing services, ensuring controlled and secure data flow.

1.5. App deployment (google cloud)

The aim of the deployment was to ensure accessibility and safety. Our solution is accessible through a link as it shown in Figure 5, with no installation required. In addition, backups of all user data are created to prevent data loss. To maintain low latency, the system automatically connects users to the nearest server based on their location – for example, users in Poland are connected to a Polish server.

Fig. 5. Communication between google cloud and solution

2. AI solution

The primary functionality on which we focused our development efforts was the implementation of a voice assistant with appropriate contextual awareness to support cooking activities, while simultaneously providing efficient navigation capabilities within our application with minimal latency. To meet these requirements, we selected LiveKit – a real-time communication platform specifically designed for building voice and video applications with WebRTC technology, enabling low-latency, high-quality audio streaming between users and AI agents. Additionally, we integrated the Groq platform – a specialized AI inference infrastructure optimized for high-performance language model processing, offering significantly faster response times compared to traditional cloud-based AI services through their custom-built Language Processing Units (LPUs).

This technological stack allows us to achieve sub-second response times while maintaining natural conversation flow, essential for effective voice-controlled cooking assistance where timing and responsiveness are critical for user experience and safety.

2.1. Data flow diagram

The architectural core of this application lies in the proper utilization of the LiveKit Cloud platform for automatic room and session management, through which a unique AI agent orchestration system has been implemented, enabling automated lifecycle management of processes in a production environment. The system employs continuous monitoring mechanisms to detect new sessions initialized by users and allocate appropriate computational resources. The event-driven architecture ensures system reactivity, where key application functionalities related to the AI assistant are automatically triggered in response to system events.

The implementation of this approach eliminates architectural requirements related to session state management mechanisms, communication error handling, and manual resource scaling. As a result, separation of concerns is achieved between application logic and infrastructure responsible for AI assistant services, which translates to high scalability and system maintainability. As depicted in Figure 6, the system architecture comprises four distinct components that collaborate through specialized communication protocols to deliver seamless voice-controlled cooking assistance. This multi-layered design separates audio processing, data management, and user interface concerns while maintaining real-time synchronization across all system components.

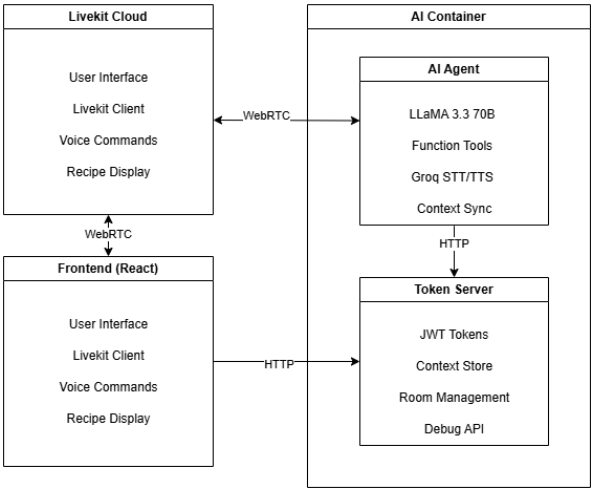

Fig. 6. AI solution diagram

Frontend component implemented as a React application, handles direct user interaction. It integrates the LiveKit SDK, enabling seamless use of LiveKit capabilities within custom UI components. This provides real-time communication, browser-based audio capture for voice commands, and rich presentation of recipe content. The interface also includes built-in accessibility features to support users with disabilities, ensuring an intuitive, inclusive experience.

The AI container hosts the system's intelligence layer. It includes the AI Agent module, written in Python, which orchestrates a LLaMA 3.3 70B language model for natural language understanding and generation, implements specialized function tools for recipe/state management, and leverages Groq STT/TTS services for high-performance speech processing with context synchronization.

The Token server, created as a Flask server, functions as the central authentication and state management hub. It generates secure JWT tokens with room-specific permissions for LiveKit access, maintains an in-memory context store for recipe progression tracking, provides comprehensive room management functionality, and exposes debugging APIs for development and monitoring purposes.

The LiveKit Cloud infrastructure serves as the communication backbone, handling sophisticated voice routing between participants, managing real-time communication channels, and processing WebRTC audio streams with automatic quality adaptation.

Data transmission occurs through two distinct pathways: HTTP-based request-response communications for particular data operations, and WebRTC streaming protocols for continuous audio exchange. The Frontend establishes unidirectional HTTP connections to the Token Server for authentication token retrieval, while the AI Container maintains bidirectional HTTP communication with the Token Server for recipe context synchronization and state updates. Simultaneously, both Frontend and AI Container maintain full-duplex WebRTC connections through LiveKit Cloud, enabling real-time voice command transmission from users and AI response delivery with minimal latency.

2.2. AI interaction

The system implements a dual-processing architecture that handles user voice commands through two distinct pathways, depending on the command type and required system response complexity.

Recipe-Related Commands presented in Figure 7 follow a comprehensive processing pipeline that begins with voice capture through the browser's microphone interface. The audio stream is transmitted via WebRTC to LiveKit Cloud, which routes it to the AI Agent module for processing. Upon receiving the audio, the system employs Silero Voice Activity Detection (VAD) to identify speech segments, followed by Groq's Speech-to-Text service for transcription. The resulting text is processed by the LLaMA 3.3 70B language model, which interprets the user's intent and triggers appropriate function tools. For recipe navigation commands such as "next step" or cooking questions like "what ingredients do I need," the system executes the corresponding function tools, updates the local context state when necessary, and synchronizes changes with the Token Server through HTTP requests. The updated recipe state is then used to generate contextually appropriate responses, which are converted to natural speech using Groq's Text-to-Speech service and delivered back to the user through the same WebRTC pathway.

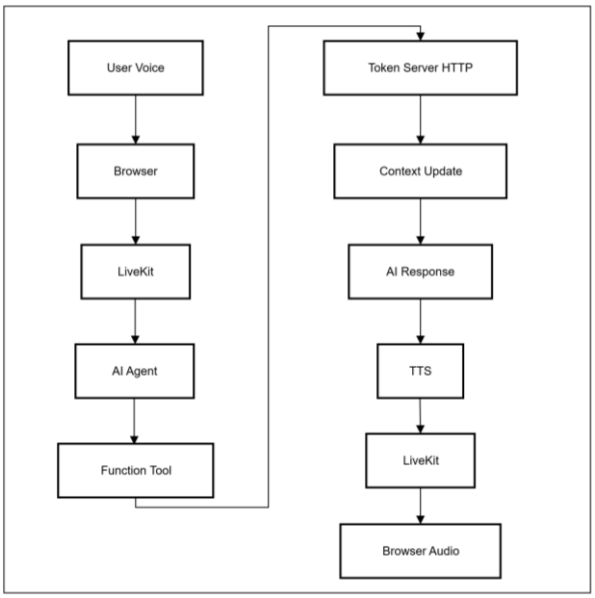

Fig. 7. Recipe related commands

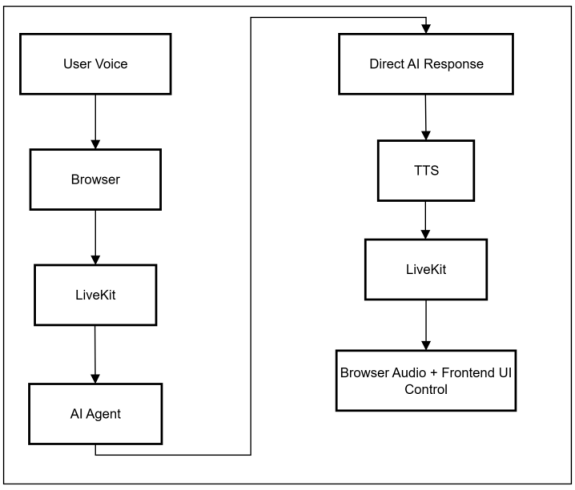

Fig. 8. Recipe related commands

Simple Interface Commands presented in Figure 8 utilize a streamlined processing approach that bypasses complex state management requirements. When users issue interface control commands such as "pause video," "increase volume," or request general cooking assistance that doesn't require recipe context updates, the audio follows the same initial pathway through LiveKit Cloud to assistant.py for VAD and STT processing. However, the LLaMA model recognizes these as direct response commands and generates immediate confirmations or answers without invoking function tools or server synchronization. For video controls, the AI provides brief acknowledgments like "Video paused," while for general cooking questions, it offers

direct advice from its knowledge base. These responses are synthesized through TTS and transmitted back to the user. For video controls specifically, the React frontend simultaneously monitors transcription streams and detects control keywords, triggering appropriate UI actions directly in the browser interface.

2.3. Metrics

Implementing voice control functionality requires a specific level of performance to ensure the application fulfils its role in a way that is satisfactory to the user. Currently, for real-time conversation, there are two dominant architectural paradigms: the traditional pipeline architecture (STT-LLM-TTS) and end-to-end real-time models.

We utilized a pipeline architecture (STT-LLM-TTS) that segments the conversational process into distinct stages: speech-to-text transcription, large language model processing, and text-to-speech synthesis. This modular approach provides several advantages including model specialization, independent optimization of each component, and robust error handling with component-level fall-back mechanisms.

In contrast, recent developments in real-time conversational AI, such as OpenAI's Realtime API and similar architectures, attempt to eliminate pipeline latency through integrated speech-to-speech processing. These models achieve significantly reduced response times compared to traditional pipeline approaches. However, this improvement comes with trade-offs in terms of computational resource requirements and reduced flexibility in component-specific optimization.

The decision to implement the traditional pipeline architecture (STT-LLM-TTS) for this cooking assistance system was based on domain-specific requirements that prioritize reliability over minimal latency. The modular approach enables independent optimization of each component for cooking environments, with STT configured for kitchen noise and the LLM specialized for recipe terminology and sequential instruction understanding. Recipe navigation demands robust state management across multiple interactions, which the pipeline architecture handles more reliably through persistent context synchronization between the AI agent and token server. The achieved response time is acceptable for culinary tasks where users typically pause between instruction requests, while simultaneously providing superior accuracy and contextual understanding. This architectural choice optimizes the trade-off between response time and system reliability, making it ideal for cooking assistance where consistent performance is more critical than immediate responses.

3. Conclusions

The aim of this work was to design and implement an AI-based cooking assistant – Cooking Master – that supports users in the preparation of meals through intelligent functionalities. The proposed system integrates features such as a voice-controlled assistant, recipe personalization based on dietary preferences and allergies, automatic ingredient scaling, shopping list generation, and multilingual support. The conducted user survey confirmed that these functionalities are considered highly valuable, which justifies their inclusion in the final solution.

The application was successfully implemented as a web-based system, ensuring accessibility across multiple platforms. The use of modern technologies, including React, Flask, Docker, MongoDB, and cloud deployment, allowed for building a scalable and resilient architecture. Integration of AI-based language models enabled natural and responsive voice interaction, which enhances usability in real cooking environments.

At the current stage, the system demonstrates state-of-the-art solutions for AI-assisted cooking and can be considered a robust proof of concept for intelligent culinary support. Nevertheless, several directions for further development have been identified. Future work will focus on:

- **Extending the range of supported commands** for the AI assistant, enabling dynamic modification of the selected recipe during cooking or suggesting alternative, similar dishes in real time.
- **Expanding the role of the AI model within the application**, by embedding the assistant across all pages, thereby allowing full website control and improving accessibility for disabled users.
- **Enhancing personalization** through recipe suggestions generated on the basis of previously viewed or selected dishes.
- **Introducing additional language options** to provide broader multilingual support.
- **Allowing users to contribute their own recipes**, thereby enriching the database with community-driven content.
- **Implementing nutrition-based portioning**, enabling users to adjust serving sizes according to caloric or nutritional requirements.
- **Improving speech recognition and synthesis**, to ensure more natural, accurate, and responsive interactions.

These planned enhancements will further strengthen the system's usability and inclusiveness, consolidating its position as a comprehensive AI-driven solution for modern cooking environments.

In summary, the Cooking Master assistant demonstrates that combining AI technologies with user-cantered design can significantly enhance everyday cooking experiences, providing an effective and innovative tool for modern households.

References

[1] Değerli, A., & Tatlisu, N. (2023). Cooking with ChatGPT and Bard: A study on competencies of AI tools on recipe correction, adaption, time management and presentation. *Journal of Tourismology and Gastronomy Studies 11*(4), 2658–2673, https://doi.org/10.21325/jotags.2024.1312

[2] Docker Inc. (2025). *Docker: Accelerated container application development*. https://www.docker.com/

[3] Groq Inc. (2025). *Groq: Accelerating large language models with AI hardware*. https://groq.com/

[4] Internationalization Working Group. (2025). *Internationalization (i18n) guidelines*. W3C. https://www.w3.org/International/

[5] LiveKit. (2025). *LiveKit: Build and scale real-time audio, video, and data apps*. https://livekit.io/

[6] Meta Open Source. (2025). *React – A JavaScript library for building user interfaces*. https://react.dev/

[7] Microsoft. (2025). *TypeScript: JavaScript with syntax for types*. https://www.typescriptlang.org/

[8] MongoDB Inc. (2025). *MongoDB: The developer data platform*. https://www.mongodb.com/

[9] Pallets Projects. (2025). *Flask: Web development, one drop at a time*. https://flask.palletsprojects.com/

[10] PostgreSQL Global Development Group. (2025). *PostgreSQL: The world's most advanced open source relational database*. https://www.postgresql.org/

[11] Shashank, M. (2025). Smart cooking companion: AI-personalized recipe finder. *International Journal of Scientific Research in Engineering & Management 9*(4), 1–9, https://doi.org/10.55041/IJSREM44681

[12] Tailwind Labs. (2025). *Tailwind CSS: Rapidly build modern websites without ever leaving your HTML*. https://tailwindcss.com/

[13] Utter, J., Denny, S., Lucassen, M., & Dyson, B. (2016). Adolescent cooking abilities and behaviors: Associations with nutrition and emotional well-being. *Journal of Nutrition Education and Behavior 48*(1), 35–41, https://doi.org/10.1016/j.jneb.2015.08.016

[14] Utter, J., Larson, N., Laska, M. N., Winkler, M., & Neumark-Sztainer, D. (2018). Self-perceived cooking skills in emerging adulthood predict better dietary behaviors and intake 10 years later: A longitudinal study. *Journal of Nutrition Education and Behavior 50*(5), 494–500, https://doi.org/10.1016/j.jneb.2018.01.021

[15] Vir, R., & Madinei, P. (2024). *ARChef: An iOS-based augmented reality cooking assistant powered by multimodal Gemini LLM*. arXiv. https://doi.org/10.48550/arXiv.2412.00627

[16] World Wide Web Consortium. (2023). *Cascading Style Sheets (CSS) snapshot 2023*. https://www.w3.org/TR/css-2023/

[17] World Wide Web Consortium. (2025). *HTML standard*. https://html.spec.whatwg.org/

[18] Przepisy.pl. (2025). (available 17.08.2025) https://www.przepisy.pl/

[19] Thermomix. (2025). (available 17.08.2025) https://www.thermomix.com/

[20] Upliance.ai. (2025). (available 17.08.2025) https://upliance.ai/

D.Sc. Eng. Robert Banasiak
e-mail: robert.banasiak@p.lodz.pl

Prof. R. Banasiak specializes in image reconstruction algorithms, computational and artificial intelligence applications in industry, and parallel data processing with GPU technologies. Since 2020, he has chaired the University Commission for Education Quality at TUL, he is active member of IEEE, the Polish Information Processing Society (PIPS), and the International Society of Industrial and Process Tomography (ISIPT).

https://orcid.org/0000-0002-1234-4949

Ph.D. Eng. Zdzisława Rowińska
e-mail: zdzislawa.rowinska@p.lodz.pl

Research and teaching assistant, Technical University Lodz. Member of Didactic Committee of HCI and Computer Science. Area of professional work and scientific research: C, C++, Python, discrete mathematics, design thinking, problem based learning projects, human computer interaction.

https://orcid.org/0000-0001-8600-3760

B.Eng. Wojciech Szczucki
e-mail: wojtekszczucki@gmail.com

Computer Science student specializing in software development, especially in C++/Python.
His professional and scientific interests focus on AI, NLP, connecting IT with different working areas like medicine.

https://orcid.org/0009-0001-3881-2536

B.Eng. Dawid Jantosz
e-mail: dawid.jantosz3@gmail.com

Computer Science student and Operations Engineer specializing in robust computer infrastructure within the banking sector.
His professional and scientific interests focus on high-availability systems, network security, and automating critical operational processes to ensure resilient financial services.

https://orcid.org/0009-0001-1448-6975

B.Eng. Łukasz Rembowski
e-mail: lukasz.rembowski26@gmail.com

Computer Science student specializing in Python programming.
His scientific and professional interests include software engineering and the integration of artificial intelligence into information systems.

https://orcid.org/0009-0001-1586-8641