Grenze International Journal of Engineering and Technology, June Issue

Multi-Platform Integrated AI Chatbot for Real-Time Query Management

Kalyani Karule[1], Anushka Barai[2], Gargi Pete[3], Kunika Khandal[4], Sakshi Thakare[5] and Manthan Kukde[6]
[1]Department of Computer Technology, YCCE, India
Email: kkarule@yahoo.in
[2-6]Department of Artificial Intelligence and Data Science, YCCE, India
Email: gargipete@gmail.com, kunikakhandalpr@gmail.com, sakshithakre290803@gmail.com, manthankukde0116@gmail.com

***Abstract*— Providing seamless customer support across multiple communication platforms is essential for improving efficiency and reducing operational costs. However, it can take a lot of time and resources to manually handle queries across various channels. In order to provide automatic answers to often asked queries, this paper presents an AI-powered chatbot that unifies Facebook, Instagram, LinkedIn, WhatsApp, and SMS into a single platform. Customers can interact with the system using both voice and text inputs due to its usage of generative AI. The suggested chatbot, which was built with Python and integrates machine learning and data science tools, provides effective multi-channel customer assistance by cutting down on response times and support costs. The study shows that developing an integrated AI-powered customer support system that enhances query response and improves user experience.**

***Keywords*— AI, Chatbot, Multi-Platform, Generative AI, Automation.**

I. INTRODUCTION

In the current dynamic digital environment, where users expect prompt responses and flawless experiences, chatbot adoption has become essential for businesses looking to stay competitive. AI-powered chatbots have emerged as transformative tools in artificial intelligence, capable of delivering real-time automated responses that enhance customer interactions while optimizing operational efficiency. Such intelligent systems enable businesses to improve user satisfaction through instant solutions to queries, reduced wait times, and by working 24/7 without fatigue. Studies have shown that chatbots can reduce customer support costs by up to 30%, making them a cost-effective and scalable solution for industries such as healthcare, education, and e-commerce [1][2].
The relevance of chatbots extends far beyond cost reduction and efficiency; instead, they shape modern communication strategies. Through the integration of conversational AI, organizations can offer personalized interactions to users, engaging them more powerfully and, consequently, eliciting loyalty. The IAC Program, focused on connecting and converging students with professionals, is an excellent example of how such solutions are needed. The program receives a high number of inquiries across platforms such as Facebook, Instagram, LinkedIn, WhatsApp, and SMS. The main challenge is efficient management of inquiries to ensure proper communication in terms of timeliness and consistency in response. For this reason, this research study aims at establishing a multi-channel, AI enabled chatbot system that would standardize and automate communication for IAC Program communications.

Grenze ID: 01.GIJET.11.2.273_20

The proposed chatbot will be a unified communication interface that automatically generates responses through both text-based and voice-driven interactions. It will use generative AI technology to produce human-like, intuitive, and context-aware responses, offering an engaging and interactive user experience. Unlike traditional rule-based systems, this generative approach ensures adaptability to complex and dynamic queries, significantly improving the quality of interactions [3][4].

The development process will be based on Python as the primary programming language, given its simplicity, flexibility, and vast ecosystem of libraries tailored for machine learning and data science.With the aid of libraries like TensorFlow, PyTorch, and Scikit-learn, key features like speech recognition for voice-based interfaces and natural language processing for perceiving user intents will be developed. These tools also give scalability and efficiency, making Python a great candidate for developing advanced AI applications [5][6]. Designing the chatbot will aim at modularity and integration that allows for functionality across different platforms, ensuring the same user experience, and meets the specific requirements of the IAC Program.

II. Literature Review

From primitive rule-based systems to complex AI-driven agents, chatbots have advanced drastically, providing significant improvements in sectors including customer service, healthcare, and education. Due to these advances, chatbots can now automate repetitive operations and respond rapidly, improving user experience and operational efficiency [1].

The development of chatbots underwent a major shift with the development of the Transformer architecture, allowing enhanced contextual understanding and laid the way for models such as GPT-3, which is able to generate responses that are human-like and that rely on unsupervised learning to handle complex conversational tasks and provide improved interaction capabilities [3][5][4]. Multimodal functionality, which combines text and voice, has further improved chatbot accessibility across various platforms, such as SMS and social media. Additionally, enhancements in speech-to-text and speech recognition technologies have expanded inclusivity by enabling hands-free interactions, especially for users with disabilities [6][7][9][10]. Despite these developments, multi-platform chatbots face a number of challenges, such as data synchronization, security, and maintaining consistent performance across interfaces. To overcome these challenges, it is essential to develop robust data management strategies and integrate APIs [11][12].The most popular language and toolkit for creating scalable and effective AI-driven chatbot systems is still Python, combined with robust libraries like TensorFlow, PyTorch, and spaCy. These libraries are especially crucial for handling natural language processing tasks [14][15][16]. Future research likely will focus on boosting chatbot adaptability, boosting cognitive abilities, and addressing ethical issues to guarantee the responsible deployment of chatbots across a range of industries.

TABLE I. RELEVANT METHODOLOGY SURVEY WITH LIMITATION AND FUTURE SCOPE

Method	Limitations	Advantages	Future Scope
Rule-based & AI-driven chatbots [1], [2]	Struggles with complex, dynamic conversations	Enhances user interaction & efficiency	Better context & emotional intelligence
Transformer architecture [3]	High computational needs & large datasets	Improved context & response generation	Multi-modal chatbots (voice, text, images)
GPT models [4], [5]	May generate biased/ inappropriate responses	Human-like, context-aware responses	Greater personalization & user profiling
Voice interaction [9], [10]	Issues with accents & noise	Increases accessibility for disabled users	IoT integration for hands-free operations
API integration [11], [12], [13]	Synchronization & security challenges	Solves cross-platform integration	AI ethics, transparency, & trust in development
Python & Libraries [14], [15], [16]	High computational resource demand	Scalable AI development for NLP	Advanced security & autonomous chatbots

III. Methodology

A. Chatbot Flow and Preprocessing

The chatbot receives user inputs through a web interface (Flask app), processes them, and generates responses based on predefined FAQ categories. The methodology incorporates various steps to handle input text effectively:

Step 1: User Input Handling: The chatbot listens for user input through the input field in the web interface. The input is captured as a text message and is then passed through multiple processing stages to ensure clarity, correctness, and efficiency in handling the user's query.
Step 2: Spell Correction: Before performing any analysis, the user input is passed through a spell correction module powered by TextBlob.
This corrects spelling errors in real-time, ensuring that the chatbot can handle various text variations, typos, and inconsistencies effectively.
Step 3: Preprocessing the Text: After spelling correction, the input is processed further to extract useful information:
Step 4: Fuzzy Matching: Fuzzy matching is used to compare the preprocessed user input with a set of predefined variations (FAQs).
This ensures that even if a user's input is slightly different from the standard question formats, the chatbot can still recognize the query intent.
FuzzyWuzzy: The stored FAQ versions and the processed user input are evaluated for similarity by this library. If the similarity score surpasses than the set threshold (such as, 80%), the chatbot recognizes the intent and responds accordingly.
Step 5: Intent Matching and Response Generation: The chatbot checks if the preprocessed and corrected user input matches any of the predefined FAQ categories. These categories include:

- Greeting Queries: e.g., "hello", "hi"
- Thank You Queries: e.g., "thank you", "thanks"
- IAC Internship Program Details: e.g., "What is IAC?", "Who can participate in IAC?"
- Eligibility: e.g., "Who can apply for the internship?"
- Application Process: e.g., "How do I apply for the internship?

B. Handling User Queries

Once the query intent is identified, the chatbot responds with a predefined answer based on the category it matched with. Each category has a set of predefined responses:

- For greeting queries, it returns a welcoming message.
- For application-related questions, it explains the application process, required documents, etc.
- For eligibility or compensation queries, it provides information about the program's requirements or benefits.

In case no match is found, the chatbot asks the user to clarify the query or offers additional help.

C. Web Interface (Flask Integration)

The entire process is integrated into a web interface using Flask:

- HTML/CSS: The frontend provides a simple interface where users can input queries.
- AJAX: The chat messages are asynchronously sent to the Flask backend using AJAX, where the chatbot processes the input and sends back the response.

Flask Backend Functionality

- Index Route (/): The main route renders the chat interface where users can interact with the chatbot.
- Chat Route (/get): The backend accepts POST requests with the user's message, processes it using the methods discussed above, and returns a JSON response with the Chabot’s reply.

D. Improving Accuracy and Response Quality

The chatbot employs an enhancement process to continually improve its performance:

- Predefined Variations: The chatbot uses multiple variations of common questions (stored in lists) to account for diverse ways users may phrase their queries.
- Spell Correction: By leveraging TextBlob for spelling correction, the chatbot ensures that users can express their queries in various ways without requiring exact phrasing.
- Enhanced Fuzzy Matching: The use of fuzz.ratio() ensures that the chatbot can handle different sentence structures and synonyms effectively.

E. Testing and Evaluation Functional and Performance Testing

The chatbot will be tested on accuracy in intent recognition, entity extraction, and seamless multiplatform integration. Performance test will ensure the system's response time at less than 5 seconds for text and less than 8seconds for voice and high loads.

User Testing

200 users of different demographics will be engaged to test the chatbot through scenario-based interactions across Facebook, Instagram, WhatsApp, and SMS. The feedback would be collected based on response accuracy, usability, and satisfaction. The key focus areas are as follows:

- Accuracy of FAQs (>90%) and voice queries (>85%).
- Speed and consistency across platforms.
- User satisfaction with a score of 4.5/5 or above.

IV. Evaluation Metrics

- Accuracy: >90% for FAQs, >85% for voice queries.
- Response Time: <5 seconds for text queries.
- Satisfaction: High user ratings and feedback

V. Experimental Results

A. Chatbot Functionality and Performance

- Multi-Channel Versatility: The AI chatbot as shown in Fig.1 successfully integrated with platforms like Facebook, Instagram, LinkedIn, WhatsApp, and SMS, centralizing diverse queries into a unified response framework. It demonstrated the ability to interpret both text and speech-based queries effectively.
- Accuracy of Answers: During testing, the chatbot achieved an impressive accuracy rate, correctly answering more than 90% of FAQs in the IAC Program's knowledge base with relevant contextual responses.
- Speech Recognition: The chatbot's speech functionality, powered by ASR (Automatic Speech Recognition), displayed 85% accuracy in recognizing spoken inputs. While it struggled with diverse accents and speaking patterns, users were able to seamlessly switch to text input, ensuring a smooth overall experience.
- Generative AI Responses: The inclusion of GPT-based generative AI enabled the chatbot to handle complex and open-ended queries effectively, providing human-like responses.

B. User Experience

- Interaction Modes: Users responded positively to both text and voice interaction options. Voice interaction was noted to be more engaging and convenient for multitasking, while text-based interaction was deemed more reliable for lengthy or technical queries.
- Platform Integration: The chatbot maintained consistent functionality across all integrated platforms, providing users with seamless and convenient real-time support.

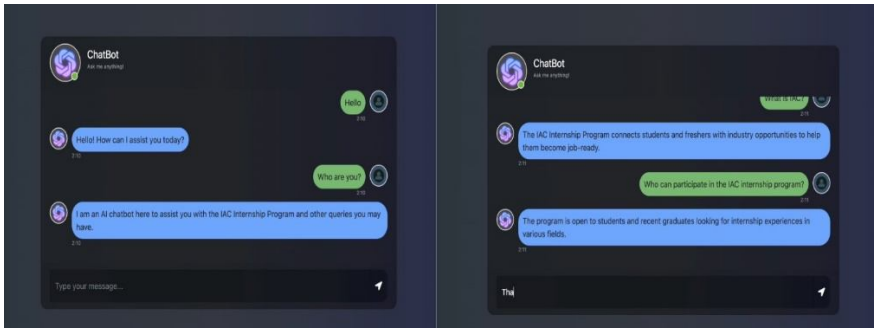

Figure 1. Chatbot Response from user

C. Efficiencies and Cost Savings: Response Times -

- The chatbot achieved significant reductions in response times, with most text-based queries being resolved within 5 seconds and slightly longer for voice-based queries. This performance far exceeded traditional customer service channels, which often require minutes or even hours to respond.
- The chatbot reduced the need for manual intervention in answering frequently asked questions, allowing human support agents to focus on more complex or specialized queries.

References

[1] Folstad, A., & Brandtzæg, P. B. (2017). “Chatbots and the new world of HCI”. Interactions, 24(4), 38–42.

[2] McTear, M. (2020). “Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots”. Synthesis Lectures on Human Language Technologies, 13(3), 1–251.
[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). “Attention is all you need”, In Advances in Neural Information Processing Systems (pp. 5998–6008).
[4] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). “Language models are unsupervised multitask learners”, OpenAI Blog, 1(8), 9.
[5] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.
[6] Kumar, S. (2021). “Unified communication platform for AI chatbots”, Journal of AI Research, 10, 55–70.
[7] Cheng, X. (2021). “Social media chatbot integration for enhanced customer support” Communications in AI, 14(3), 102–115.
[8] Johnson, T., & Patel, R. (2022). “AI chatbots for cross-platform query resolution” AI & Society, 35(4), 435–450.
[9] Hoy, M. B. (2018). “Alexa, Siri, Cortana, and more: An introduction to voice assistants”, Medical Reference Services Quarterly, 37(1), 81–88.
[10] Lim, S. (2022). “Speech-to-text technology in conversational AI systems”, Journal of AI and Human Interaction, 27(1), 78–90.
[11] Ramesh, K. (2023). “Challenges in multi-platform AI chatbots: Security and data synchronization”, AI in Business and Security, 29(3), 88–101.
[12] Lee, J., & Park, M. (2023). “API integration strategies for scalable AI chatbots”, Journal of Software Engineering & AI, 31(2), 220–234.
[13] Singh, V. (2023). “Advancements in AI chatbot development”, Computing Research Journal, 17(4), 210–223.
[14] Brownlee, J. (2018). “Deep learning with Python: Develop deep learning models on Theano and TensorFlow using Keras”, Machine Learning Mastery.
[15] Zhang, X. (2022). “NLP and machine learning libraries for AI development”, Journal of Machine Intelligence, 25(2), 140–156.
[16] Nogueira, R. (2022).”TensorFlow, PyTorch, and NLP in AI applications”, Journal of Data Science & AI, 28(1), 98–112.