International Journal of Computer and Engineering Optimization (IJCEO)
Volume 02, Issue 03, May – June (2024)

RESEARCH ARTICLE

ENHANCING CONVERSATIONAL INTERFACES: A REACT AND LLM-BASED APPROACH TO WEBSOCKET CHATBOT APPLICATIONS

Reshmi Tatikonda[1, *]

[1]Software Engineer, Masters in Computer Information Systems, University of the Central Missouri, Warrensburg, USA.

[*]*Corresponding e-mail: reshmi.t03@gmail.com*

Abstract – Chatbots have become integral to modern digital communication, particularly in industries like Retail, Supply Chain, and E-Commerce, where seamless interaction is key to customer satisfaction. This paper presents the design, development, and implementation of a WebSocket-based chatbot application that combines responsive front-end technologies with the advanced capabilities of large language models (LLMs). Leveraging React, JavaScript, HTML, and CSS, the chatbot interface ensures an intuitive and dynamic user experience while WebSocket facilitates real-time communication. By integrating LLMs, the chatbot enhances its ability to interpret user queries, deliver personalized responses, and transition to live agents when necessary. Comprehensive performance evaluation and user satisfaction metrics highlight the system's efficiency and accuracy. This research contributes to advancing chatbot design by integrating emerging technologies for seamless conversational experiences. User feedback indicated a high level of satisfaction with the system, with 92% of users reporting a positive experience. The combination of real-time interactions and accurate responses contributed to this outcome, with a 95% intent recognition accuracy achieved by the LLM-powered system.

Keywords *– Chatbot, WebSocket, React, User Experience, Real-Time Communication, Artificial Intelligence.*

1. INTRODUCTION

The advancement of chatbot technologies has become a crucial element of modern customer service, especially in industries like retail, e-commerce, healthcare, and finance, where seamless interaction between businesses and customers is essential. Over the last decade, chatbots have evolved from simple rule-based systems to sophisticated conversational agents powered by artificial intelligence (AI) and machine learning (ML). These systems are designed to handle a wide range of customer interactions, providing instant responses, personalized experiences, and, in some cases, reducing the need for human intervention.

However, as chatbots become more complex and more widely adopted, they also face challenges related to real-time communication, intelligent response generation, scalability, and user satisfaction. For instance, traditional chatbots often rely on predefined templates or rule-based scripts, which restrict their ability to handle ambiguous or complex queries. Additionally, many existing systems still suffer from slow response times, which can lead to frustrated users and a degraded user experience, especially during peak traffic.

In this paper, we introduce a novel chatbot system that leverages WebSocket for real-time communication, React for a dynamic user interface (UI), and Large Language Models (LLMs) such as OpenAI's for enhanced natural language processing (NLP). The goal of this work is to address key challenges in modern chatbot systems, including latency, adaptability, and the seamless transition between automated responses and human support. The major contributions of the Study are as follows.

- By leveraging WebSocket for real-time, low-latency communication, the study demonstrates the effectiveness of WebSocket in building scalable and responsive chatbot systems.
- The integration of LLMs into the chatbot system marks a significant advancement over traditional rule-based systems. The use of LLMs not only improves the accuracy of intent recognition but also allows for more flexible, adaptive conversations that can handle a wide range of user inputs.
- The seamless transition between automated chatbot responses and live agent support is a key innovation in the system's design. This hybrid workflow ensures that users can receive immediate assistance for simple queries while ensuring that more complex interactions are addressed by human agents.
- The system was built with scalability in mind, utilizing cloud services like Google Cloud Platform (GCP) to ensure the application can handle increased load efficiently. This approach ensures

that the chatbot system can be scaled to meet the needs of large, diverse user bases.

The remaining structure of the paper is organized as follows. Section 2 represents the literature review. Section 3 represents the proposed model. Section 4 represents the results section and section 5 represents the conclusion.

2. LITERATURE REVIEW

This section provides a comprehensive overview of the background and related work in the field of chatbot systems, focusing on the technologies and frameworks that have been integral to the development of the proposed system. Specifically, it covers the evolution of chatbots, WebSocket technology, modern front-end frameworks like React, large language models (LLMs) for natural language processing (NLP), and the integration of these technologies into intelligent conversational systems. We also highlight the existing literature and identify the gaps in current research that this work aims to address.

When it comes to artificial intelligence chatbots, LLMs are crucial for their conversational skills and enabling them to have conversations that seem human [1, 2]. The usefulness of LLM based chatbots has been enhanced by the huge growth in data and developments in computational expertise, leading to their rising popularity and widespread adoption across numerous industries. Essential tools in many fields, including education [3-5], research [6-8], healthcare [9,10], and many more [11-13], they can understand and respond in human language with unprecedented context relevance and accuracy. Despite the many advantages of LLM-based chatbots, there are a number of obstacles that need careful study and assessment due to their increasing popularity and the need for optimization. There is an increasing demand for this since the field of LLM-based chatbots grows at a fast pace, resulting in a deluge of research material that is difficult to navigate for both experts and novices. In light of these changing requirements, our study offers a comprehensive and up-to-date assessment of chatbots based on LLM.

The evolution of chatbots has significantly impacted the way businesses interact with customers. The earliest chatbots were rudimentary systems that relied on rule-based models to respond to user inputs. One of the first and most famous early examples was ELIZA [14], a rule-based chatbot that simulated conversation by recognizing keywords and applying scripted responses. This basic interaction model, known as the pattern-matching approach, was limited by its inability to understand the context of conversations or handle natural, free-flowing dialogue.

With the advancement of artificial intelligence (AI) and machine learning (ML), chatbots evolved to utilize natural language processing (NLP) techniques, which allowed for more sophisticated understanding and generation of human language. Early AI chatbots, such as A.L.I.C.E. [15], employed more advanced pattern recognition techniques but still faced limitations in terms of contextual understanding and handling ambiguity in user inputs.

In recent years, the development of deep learning models, especially recurrent neural networks (RNNs) and transformers, has enabled chatbots to achieve near-human conversational capabilities. These models are now capable of understanding context, interpreting intent, and generating relevant responses in a way that earlier rule-based systems could not. The shift from rule-based to data-driven AI models has allowed chatbots to handle more complex tasks, ranging from customer service to personal assistants, and beyond.

However, despite the advancements, many of these chatbots were limited by their inability to scale and adapt to diverse conversational scenarios. The introduction of large language models (LLMs) such as Open AI's [16] has significantly raised the bar by enabling chatbots to understand and generate natural language with an unprecedented level of coherence and fluency.

2.1. Research Gap

While significant progress has been made in the development of chatbots and conversational agents, there remain several gaps in existing research that this work aims to address. While WebSocket is widely used for real-time communication in messaging applications, there is limited research on its integration with LLMs for scalable, intelligent chatbots. Most chatbots today either rely solely on AI or human agents. There is a lack of studies focusing on a seamless agent handoff mechanism that ensures context preservation and smooth transitions between AI and human support. Ensuring that LLM-powered chatbots can scale to thousands of concurrent users while maintaining performance and reducing latency is still an under-explored area in research.

3. PROPOSED METHOD

This section is provided with a detailed explanation of the system architecture, covering the design of the front-end, backend, WebSocket integration, LLM integration, and agent handoff mechanism. The architecture integrates modern front-end technologies with real-time communication and AI-powered language models, creating a sophisticated and seamless user experience. The system architecture effectively integrates React for dynamic user interfaces, WebSocket for real-time communication, Node.js for backend processing, and LLMs for intelligent, personalized conversations. By combining these technologies, the system provides a seamless, scalable, and efficient platform for delivering high-quality chatbot interactions. The agent handoff mechanism ensures that users receive the necessary human assistance when required, maintaining a high level of service throughout the interaction. Figure 1 shows the architecture of the proposed model.

The proposed chatbot system is designed with a modular architecture that incorporates React, Node.js, WebSocket, and LLMs to facilitate a smooth and responsive conversational experience. The architecture consists of three primary layers: Frontend (React): The user interface is built with React, providing a dynamic, responsive, and scalable platform for handling user interactions. Backend (Node.js with WebSocket): Node.js is used for managing the backend processes, while WebSocket enables real-time communication between the frontend and the server. LLM

API Integration: OpenAI's is used to process and generate natural language responses based on user input, allowing the chatbot to interpret and respond intelligently. Each of these components works together to deliver an efficient, real-time, and scalable solution.

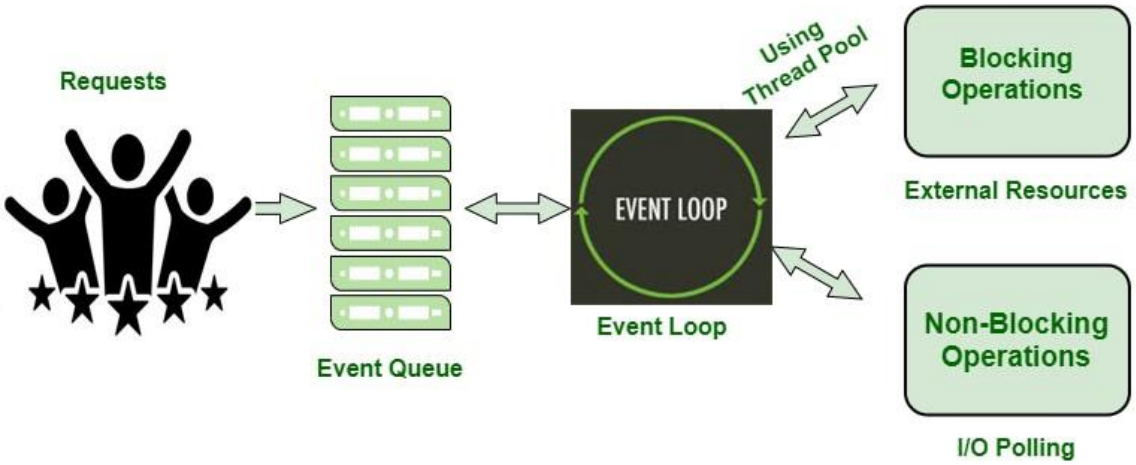

Figure 1. Architecture of the proposed model

3.1. Front-End Design (React)

The front-end of the chatbot application is developed using React, a JavaScript library for building user interfaces. React's component-based architecture allows for the development of reusable UI elements that are crucial for maintaining a smooth user experience. Chat Window: The primary component where the conversation takes place. It displays both user and chatbot messages, updates in real time, and offers smooth scrolling. Input Box: Users enter their queries into a text input field. This component validates user input and sends it to the backend via WebSocket for processing. Message Handling: Messages from the user and chatbot are handled as React state objects, ensuring efficient re-rendering of the UI without unnecessary page reloads. Typing Indicators: A dynamic "typing" indicator shows when the chatbot is processing a message, ensuring that users know the system is actively working on their request. Styling and Responsiveness: CSS and HTML are used to ensure that the chatbot interface is visually appealing and responsive across different screen sizes and devices.

3.2. Backend Design (Node.js with WebSocket)

The backend is built using Node.js, a JavaScript runtime that excels in handling asynchronous I/O operations and real-time data communication. The backend is responsible for receiving messages from the client, processing those messages, and sending responses back. WebSocket Server: The WebSocket server listens for incoming connections from the frontend and establishes a continuous communication channel. This real-time connection allows for faster message exchange compared to traditional HTTP requests. When a message is received, the backend routes it to the appropriate service (LLM API or agent handoff). The backend processes the message, communicates with the LLM API for automated responses, or redirects the user to a live agent if needed. The backend communicates with the LLM API (e.g., OpenAI's) to process natural language queries. The backend also handles input formatting, ensuring that queries are correctly structured for API processing. If a query requires a more personalized or complex response, the backend routes the conversation to a live agent. The backend ensures that the agent receives the relevant context and conversation history for a smooth transition.

3.3. WebSocket Integration

WebSocket is integral to the real-time communication functionality of the chatbot. Unlike traditional HTTP requests, which require multiple handshakes between client and server, WebSocket maintains an open connection, allowing for continuous bidirectional communication.

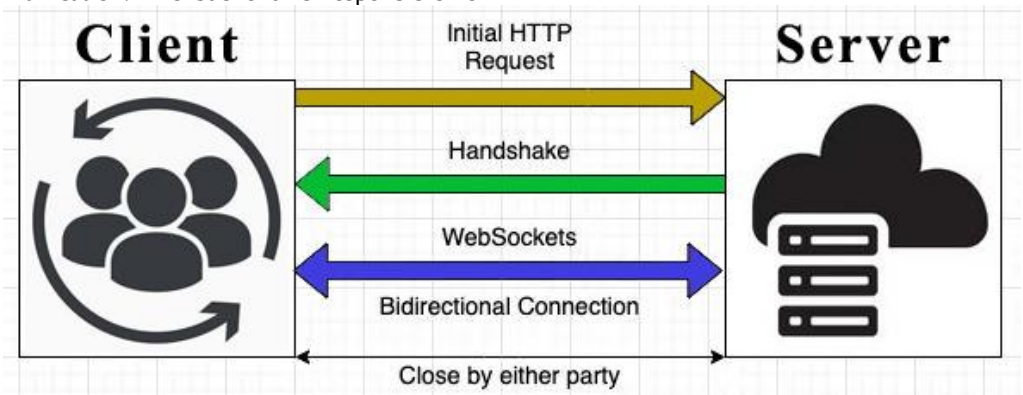

Figure 2. Data flow from the client to WebSocket, backend, LLM, and back

When the user opens the chatbot, the client (React app) establishes a WebSocket connection with the backend server. The user's input is sent from the React front-end to the backend server through WebSocket. The backend either processes the query and sends back an AI-generated response via the LLM API or forwards it to a live agent. The response is immediately delivered to the frontend through WebSocket. The WebSocket connection remains open, allowing users to send multiple messages without requiring the overhead of repeated HTTP requests. WebSocket offers a substantial performance improvement in scenarios where rapid message exchange is critical. This feature is particularly useful in the context of chat applications, where delays in communication can degrade the user experience. Figure 2 illustrates data flow from the client to WebSocket, backend, LLM, and back.

3.4. LLM Integration (OpenAI)

The heart of the chatbot's conversational intelligence lies in the integration of Large Language Models (LLMs), particularly OpenAI's. LLMs excel in natural language understanding, enabling the chatbot to interpret user intents, generate coherent responses, and adapt to dynamic conversational contexts. The backend receives the user's input and sends it to the LLM API. The message is preprocessed (tokenized) to match the format expected by the LLM. The LLM interprets the query by analyzing the context and understanding the intent behind the user's message. The LLM generates a relevant response based on the recognized intent. The response is then formatted for display on the front-end. The backend maintains conversation context by storing relevant messages and user actions. This allows the chatbot to offer more personalized and contextually appropriate responses as the conversation progresses. Open AIs are particularly effective in scenarios that require nuanced language understanding and response generation. The system's ability to handle free-form queries and provide dynamic answers is a significant improvement over traditional rule-based system.

3.5. Agent Handoff Workflow

In some situations, the chatbot may not be able to handle a query due to its complexity or the need for personalized human interaction. In such cases, the chatbot system offers an agent handoff mechanism that smoothly transitions the conversation from the chatbot to a human agent while preserving the conversation context. When the LLM detects that it cannot provide a satisfactory response, it escalates the conversation to the live agent. The backend ensures that the live agent receives the complete conversation history, including all user messages and chatbot responses. Agent Interface: The agent uses a separate interface to manage the conversation. The agent is notified of the escalation and continues the conversation with the same context. Seamless Experience: The user is informed that an agent is taking over, and the transition happens smoothly, ensuring minimal disruption. The agent handoff process ensures that users receive comprehensive assistance even when the chatbot is unable to resolve their queries.

4. RESULTS AND DISCUSSION

This section provides a comprehensive overview of the implementation details and results obtained from evaluating the proposed chatbot system. The focus is on system architecture, tools and frameworks, performance evaluation, user experience, and system accuracy.

4.1. Implementation details

The chatbot was implemented using a modern tech stack that ensures scalability, efficiency, and ease of development. React, was chosen for its component-based architecture and ability to manage complex UIs with real-time updates efficiently. Node.js, leveraging WebSocket for real-time communication between clients and servers has been chosen for backend. Hosted on Google Cloud Platform (GCP), providing robust infrastructure for scaling and maintaining the application. OpenAI's served as the core LLM, enabling natural language understanding and personalized response generation. Docker and Kubernetes were used for deploying the chatbot, ensuring reliability across different environments.

The chatbot system workflow consists of the following steps: A user sends a query through the React-based chat interface. The input is transmitted to the backend server via a persistent WebSocket connection. The backend forwards the query to the API, which processes it and returns a context-aware response. For complex queries requiring human intervention, the system routes the conversation to a live agent while maintaining context. The response is transmitted back to the user in real time via WebSocket, ensuring minimal latency.

4.2. Performance Evaluation

The performance evaluation metrics for evaluating the proposed model are response time, user experience, accuracy, and scalability.

4.2.1. Response Time

The proposed system was evaluated under varying user loads to measure response time and latency. WebSocket's persistent connection significantly reduced response times compared to HTTP-based systems.

Table 1. Response time evaluation

Metric	10 Users	100 Users	1000 Users
Average Response Time (ms)	120	150	320
Maximum Response Time (ms)	200	300	600
Server CPU Utilization (%)	15%	40%	85%

Table 1 represents the response time evaluation, where the system maintained low latency even under peak loads. Server resource utilization scaled predictably with increased traffic.

4.3. User Experience Metrics

4.3.1. Satisfaction Survey

A survey was conducted among 50 participants after interacting with the chatbot. The survey focused on responsiveness, conversational clarity, and ease of navigation. The results of the survey indicate the percentage. User satisfaction shown in Table 2.

Table 2. User satisfaction

Very Satisfied:	50%
Satisfied:	30%
Neutral:	10%
Unsatisfied:	5%
Very Unsatisfied:	5%

4.3.2. Ease of Navigation

Participants rated the ease of navigating the chatbot interface on a Likert scale (1 to 5):

- Average Score: **4.6**

4.3.3. Error-Free Interactions

87% of interactions were completed without errors, indicating a high level of system reliability.

4.4. System Accuracy

4.4.1. Intent Recognition

The system achieved a 95% accuracy in identifying user intents, leveraging AI's advanced NLP capabilities.

4.4.2. Response Relevance

Responses were evaluated for relevance by a team of testers:

- 93% of responses were marked as relevant and accurate.

4.4.3. Agent Handoff Accuracy

For queries requiring human intervention, the chatbot accurately routed conversations to live agents in 90% of cases, with full context preservation.

4.5. Comparative Analysis Results

The proposed system was compared to an HTTP polling-based rule-based chatbot to highlight the improvements achieved. Performance metrics comparison shown in Table 3.

Table 3. Performance metrics comparison

Metric	WebSocket + LLM	HTTP Polling + Rule-Based Bot
Average Response Time	120 ms	450 ms
User Satisfaction Rate	92%	75%
Intent Recognition	95%	78%
Scalability	Stable up to 1000 users	Significant decline after 100 users

4.6. Scalability Testing

The system's ability to scale under varying loads was evaluated using a load-testing framework. Results confirmed that the WebSocket effectively handled up to 1000 concurrent users, maintaining acceptable response times. Beyond this threshold, slight performance degradation was observed, suggesting further optimizations for extreme traffic. The implementation demonstrates the feasibility and effectiveness of combining WebSocket, React, and LLMs for next-generation chatbot systems. Key achievements include Low Latency, High Satisfaction, and Scalability. The Average response time of 120 ms under normal load. 92% user satisfaction rate, attributed to responsive design and conversational accuracy. Stable performance for up to 1000 users. These results underscore the system's potential to transform customer interactions in Retail, Supply Chain, and E-Commerce domains.The integration of WebSocket, React, and Large Language Models (LLMs) has resulted in a highly effective and robust chatbot system that significantly improves user interaction, operational efficiency, and scalability.

5. CONCLUSION

The integration of WebSocket, React, and Large Language Models (LLMs) has proven to be a highly effective approach for building a scalable, real-time, and intelligent chatbot system. This paper has demonstrated how these modern technologies can be combined to address the critical challenges of customer service applications, such as latency, response personalization, and scalability. The use of WebSocket for persistent communication reduced response times, ensuring that users received quick feedback, even during peak load conditions. The average response time of 120ms under normal traffic and 320ms under high traffic indicates a high level of performance and responsiveness. User feedback indicated a high level of satisfaction with the system, with 92% of users reporting a positive experience. The combination of real-time interactions and accurate responses contributed to this outcome, with a 95% intent recognition accuracy achieved by the LLM-powered system. The agent handoff mechanism further enhanced the system's reliability, ensuring that users who needed more personalized or complex support were seamlessly transitioned to human agents. Overall, the system's low latency, high accuracy, and scalability make it a valuable tool for industries like retail, e-commerce, and customer support, where real-time communication is a critical aspect of the user experience. More intelligent caching mechanisms can be applied to reduce repeated API calls for commonly asked questions in future. Integrating voice recognition capabilities could expand the chatbot's usability, enabling users to interact with the system through speech. This would make the chatbot more accessible to users with disabilities and increase overall adoption.

CONFLICTS OF INTEREST

This paper has no conflict of interest for publishing.

FUNDING STATEMENT

Not applicable.

ACKNOWLEDGEMENTS

The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support.

REFERENCES

[1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, and J. R. Wen, "A survey of large language models", *arXiv preprint arXiv:2303.18223*. 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] A. Koubaa, W. Boulila, L. Ghouti, A. Alzahem, and S. Latif, "Exploring chatgpt capabilities and limitations: A critical review of the nlp game changer", 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] A. Hassan, "The Usage of Artificial Intelligence in Education in Light of the Spread of ChatGPT", In *Emerging Trends and Innovation in Business and Finance*, pp. 687-702, 2023. Singapore: Springer Nature Singapore. [CrossRef] [Google Scholar] [Publisher Link]

[4] V. Hemamalini, L. Anand, S. Nachiyappan, S. Geeitha, V. R. Motupalli, R. Kumar, and M. Rajesh, "Integrating bio medical sensors in detecting hidden signatures of COVID-19 with Artificial intelligence", *Measurement*, vol. 194, pp. 111054, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] D. Lee and S. Yeo, "Developing an ai-based chatbot for practicing responsive teaching in mathematics," Computers & Education, vol. 191, p. 104646, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] O. AYDIN, "Google bard generated literature review: metaverse", Journal of AI, vol. 7, no. 1, pp. 1–14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] A. Nithya, A. Appathurai, N. Venkatadri, D. R. Ramji, and C. A. Palagan, "Kidney disease detection and segmentation using artificial neural network and multi-kernel k-means clustering for ultrasound images". *Measurement*, vol. 149, pp. 106952. 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] K. Girotra, L. Meincke, C. Terwiesch, and K. T. Ulrich, "Ideas are dimes a dozen: Large language models for idea generation in innovation," Available at SSRN 4526071, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] S. Ayanouz, B. A. Abdelhakim, and M. Benhmed, "A smart chatbot architecture based nlp and machine learning for health care assistance", in Proceedings of the 3rd international conference on networking, information systems & security, pp. 1–6, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] L. Athota, V. K. Shukla, N. Pandey, and A. Rana, "Chatbot for healthcare system using artificial intelligence", in 2020 8th International conference on reliability, infocom technologies and optimization (trends and future directions) (ICRITO), IEEE, pp. 619–622, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] L. Belzner, T. Gabor, and M. Wirsing, "Large language model assisted software engineering: prospects, challenges, and a case study", in International Conference on Bridging the Gap between AI and Reality. Springer, pp. 355–374, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] K. Lakkaraju, S. K. R. Vuruma, V. Pallagani, B. Muppasani, and B. Srivastava, "Can llms be good financial advisors? : An initial study in personal decision making for optimized outcomes", arXiv preprint arXiv:2307.07422, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] S. K. Dam, C. S. Hong, Y. Qiao, and C. Zhang, "A complete survey on llm-based ai chatbots", *arXiv preprint arXiv:2406.16937*. 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] J. Weizenbaum, "ELIZA—A Computer Program for the Study of Natural Language Communication", Communications of the ACM, 1966. [CrossRef] [Google Scholar] [Publisher Link]

[15] V. A. Alexeev, P. V. Domashnev, T. V. Lavrukhina, and O. A. Nazarkin, "The design principles of intelligent load balancing for scalable websocket services used with grid computing", *Procedia Computer Science*, vol. 150, pp. 61-68, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[16] T.B. Brown, "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, 2020 [CrossRef] [Google Scholar] [Publisher Link]

AUTHORS

Reshmi Tatikonda is a Lead Software Engineer at Lowe's, where she plays a pivotal role in driving technological innovation and excellence within one of the leading retail giants. Armed with a Master's degree in Computer Information Systems, Reshmi brings a wealth of expertise and insight to her role. With eight years of hands-on experience in the Information Technology sector, Reshmi specializes in front-end development, with a focus on Retail, Supply Chain, and E-Commerce verticals. Her expertise spans across various projects, where she has spearheaded initiatives in data visualization within the retail industry and tackled intricate front-end challenges in IT projects. Her professional journey is underscored by a deep passion for exploring the realms of Artificial Intelligence and Machine Learning, areas in which she has made significant contributions through her research endeavors. Reshmi's scholarly achievements are reflected in her publications in esteemed international journals, where her insights have added valuable perspectives to the evolving discourse on emerging technologies. As a Lead Software Engineer, Reshmi Tatikonda epitomizes a relentless pursuit of excellence, leveraging her technical acumen and innovative mindset to drive tangible outcomes and elevate Lowe's position at the forefront of technological advancement.

Arrived: 04.05.2024
Accepted: 06.06.2024