Vibhu Tyagi, 2026, 14:4
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Ai Chatbot with Real Time Voice Communication

Vibhu Tyagi, Utkarsh Bhardwaj, Rupesh Kumar
Department of Computer Applications & Technology (SCAT),
Galgotias University, Greater Noida, Uttar Pradesh, India

Abstract- Virtual Assistant is significant for people to interact with computers and perform tasks on different environment in the today's world. Old chatbot systems are not much efficient at handling many users simultaneously and lack secure mechanisms. In this paper, we proposed Assistant AI Chatbot that can understand the feeling of human and response the answer quickly. AI Chatbot's Process Flowcharts and Diagrams have been designed. We have conducted comprehensive analyse of Technology Stack, Security Hardening and User Interface Implementation. In the Evaluation and result we have examined the result of Performance Analysis, System Efficiency Evaluation Security and Reliability Analysis. My proposed AI Chatbot performance is better than Tradition AI Chat Bot.

Keywords: Natural Language Processing (NLP), Voice Recognition, Real-Time Communication.

I. INTRODUCTION

Background: Virtual assistants and AI chatbots are really important in systems. They are used in lots of industries for example customer service and healthcare and education [1]. People want to talk to these systems and get answers away. So, companies are making agents that can understand what people are saying and respond in a way that makes sense. This is a deal because people want to be able to ask questions and get good answers. Naturals Language Processing and machine learning have made it possible for chatbots to be a lot better at understanding what people mean and giving answers [3][5]. This is because Natural Language Processing and machine learning are tools that help Virtual assistants and AI chatbots work well. However old systems have some problems. They are not always able to respond away and they can be hard to use. Virtual assistants need to be able to understand voices and communicate on time. They also need to be secure so people can trust assistants and AI chatbots [1].

Problem Statement: Despite the improvement of AI-based chatbots many virtual assistant systems still struggle to provide timely responses. Traditional chatbots usually rely on set rules or basic processing power [1]. This results in them not understanding what users want and making interactions inefficient. The voice recognition, live communication and secure login systems do not work well together. This limits how useful virtual assistants are. Many existing virtual assistants still have these problems. Virtual assistants need to improve in areas like voice recognition and secure login. AI-based chatbots are advancing rapidly.

Virtual assistants are not improving at the same rate. They often do not understand what users want. Interactions with assistants are often inefficient. The lack of integration between voice recognition and secure authentication limits how useful virtual assistants are. Virtual assistants have limitations in delivering responses that make sense. They are not able to provide responses in time. The performance of assistants is limited by what they can do. They struggle to deliver what users expect from them [1]. Virtual assistants have a lot of room, for improvement.

Proposed Solution: To fix the problems with systems we have an Assistant AI Chatbot. This Virtual Assistant AI Chatbot gives us answers away and it understands what we are talking about. The Virtual Assistant AI Chatbot works with both voice and text which makes it easy for users to talk to it. It uses React.js to make the user interface interactive and fun to use. The Virtual Assistant AI Chatbot uses Node.js and Express.js for the backend so it can process information quickly. The Virtual Assistant AI Chatbot uses WebSocket technology so it can talk to us in time. The system uses MongoDB to store data in a way that can grow with the Virtual Assistant AI Chatbot. For security the system uses OAuth 2.0 only authorized users can use the Virtual Assistant AI Chatbot.

The Gemini API uses advanced Natural Language Processing techniques to understand user queries and generate meaningful responses [3][4][5]. The Gemini API is part of the Virtual Assistant AI Chatbot to help it understand what we say. This way the Virtual Assistant AI Chatbot can give us answers that make sense. This approach makes the Virtual Assistant AI Chatbot work well. It can be used in many different situations. It can be used in life because the Virtual Assistant AI Chatbot and the Gemini API work well together. The Virtual Assistant AI Chatbot is useful because it can understand what we mean and give us answers. The system is good, for applications because it works well and can grow with the Virtual Assistant AI Chatbot.

Core Contributions:

This paper presents a design for a Virtual Assistant AI Chatbot by combining the following main parts:

Real-Time Communication Framework: We use WebSocket technology to allow fast communication between the user and the server. This way the Virtual Assistant AI Chatbot can respond instantly. Interact with the user better. The chatbot relies on real-time communication to work well. Real-time communication is key to Assistant AI Chatbot.

AI-Powered Natural Language Processing: The Gemini API helps the Virtual Assistant AI Chatbot understand and create human-language. This enables the chatbot to know what the user wants and provide answers that make sense in the conversation. The Virtual Assistant AI Chatbot uses AI-powered natural language processing to understand the user and provide answers. Natural Language Processing is important for Assistant AI Chatbot [3][4][5].

Voice Recognition and Multimodal Interaction: The Virtual Assistant AI Chatbot can. Understand voice commands in addition to text-based input. This makes it easier for users to communicate with the chatbot and improves their experience with the Virtual Assistant AI Chatbot. Voice recognition is a feature of Virtual Assistant AI Chatbot. Voice input processing follows modern speech recognition principles described in speech and language processing literature [5].

Scalable Full-Stack Architecture: The Assistant AI Chatbots frontend is built with React.js for a user interface. The backend is built with Node.js and Express.js for processing and MongoDB is used for data storage. The Virtual Assistant AI Chatbot uses a full-stack architecture to handle users. This architecture helps Assistant AI Chatbot to handle user traffic.

Secure Authentication and Authorization: OAuth 2.0 is used to keep user data safe and secure. This ensures that authorized users can access the Virtual Assistant AI Chatbot and protects user information. The Virtual Assistant AI Chatbot uses authentication to protect user data. Authentication is important, for Assistant AI Chatbot security.

II. LITERATURE REVIEW

Modern chatbot systems, such as Google Assistant and Microsoft Copilot, leverage Machine Learning (ML), Artificial Intelligence (AI), and Natural Language Processing (NLP) to understand user queries and generate intelligent, human-like responses [1][5]. These systems typically use cloud-based architectures to process large volumes of data, continuously improve their performance, and support scalable deployments. REST APIs are commonly integrated with chatbot frameworks and external services to enable seamless communication between different system components [1].

Recent advancements in Transformer-based language models have significantly enhanced chatbots by enabling them to understand user intent, conversational context, and linguistic nuances more effectively, resulting in accurate and context-aware responses [3][4]. AI services such as the Gemini API further strengthen chatbot capabilities by providing advanced NLP features for language understanding and response generation [3][5]. Real-time communication is another essential aspect of modern chatbot systems, where technologies like WebSocket enable persistent, bidirectional communication between clients and servers, reducing latency compared to traditional HTTP-based communication.

Backend frameworks such as Node.js and Express.js efficiently manage asynchronous operations and multiple concurrent connections, ensuring that the system remains responsive even under heavy user traffic. Despite these advancements, traditional chatbot architectures often face challenges in handling a large number of simultaneous users and may struggle to accurately interpret complex user requirements, highlighting the need for more scalable, intelligent, and context-aware chatbot solutions [1][5].

III. SYSTEM ARCHITECTURE

High-Level Architecture Overview: We made a Client Layer using React.js. we create a user interface in this layer. It's easy to use and look good for user Experience. AI have two types of features typing and talking. This Layer uses browser-based voice recognition to understand what user is saying. We are used Tailwind CSS to make a good interface and work well on different devices. We are Build a Server Layer using the Node.js and Express.js. This layer handles what the user asks for manages the parts of the system and helps them work together. The Server Layer uses WebSocket to let the Virtual Assistant AI Chatbot talk to the user in time. This means the user gets answers quickly. The Server Layer also uses the Gemini API to understand what the user means and give answers.

Architecture Components: The User Interaction Module enables users to communicate with the Virtual Assistant AI Chatbot through both text and voice inputs in a simple and user-friendly manner. When a user speaks, the chatbot uses voice recognition technology to convert the speech into text before processing the request. Communication between the client and server is handled using WebSocket, which enables real-time message exchange and ensures that users receive instant responses with minimal delay. The module is designed to provide a seamless experience across different devices, making the chatbot accessible, responsive, and easy to use. The AI Processing and Natural Language Understanding (NLU) Module is responsible for analysing user queries, understanding their intent, and generating meaningful responses. This module leverages the Gemini API to perform Natural Language Processing (NLP), allowing the chatbot to interpret the context of user input, identify key information, and determine the user's intent. Based on this analysis, the AI generates accurate, context-aware, and human-like responses. By integrating the Gemini API, the chatbot can effectively understand natural language, improve conversational quality, and deliver intelligent and relevant responses to users.

Backend and Communication Management: We build a backend system used Node.js and Express.js. They are handling API requests, system management and business logic. And WebSocket help us to manage a connection between the client and server. And also allow for two-way communication. User queries are processed and responses are delivered instantly. The backend also manages user sessions and works securely with outside the APIs.

Data Storage and Security: We are used MongoDB to store and manage user data chat history and session data in data layer. And system is used OAuth 2.0 for user authentication and authorization. They have also access control to protect data. The database is used to handle a lot of operations and perform well.

Processing Flow: The user submits their input in the form of text or voice through the client application. The input is then pre-processed and converted into a well-structured format suitable for analysis. After preprocessing, the server sends a request to the Gemini API, where the AI model analyses the user's query, understands the context, and generates an appropriate solution. The generated response is then returned to the server, which processes it and forwards it back to the client in real time, ensuring that the user receives a fast and seamless AI-powered response. [1]

IV. METHODOLOGY

Data Collection and Integration Objective: The Data Collection and Integration phase focuses on gathering the necessary information and defining the chatbot's functionality. The primary objective is

to understand the system requirements and ensure that the chatbot can efficiently handle multiple user requests simultaneously while providing fast, accurate, and contextually relevant responses. During the requirement analysis, the chatbot is designed to support concurrent interactions and maintain seamless communication with users.

Dataset collection involves gathering user queries and inputs to test the chatbot's performance, improve response quality, and enhance conversational accuracy. To enable advanced natural language understanding and response generation, the Gemini API is integrated into the system. Additionally, a suitable technology stack consisting of React.js, Node.js, Express.js, MongoDB, and WebSocket is selected to ensure high performance, scalability, and real-time communication. Finally, performance baselines are established by measuring key metrics such as response time, latency, and accuracy to evaluate the chatbot's overall efficiency and effectiveness.

System Design and Architecture Development Objective: The System Design phase focuses on developing an efficient and scalable architecture for a real-time chatbot that delivers responsive and reliable performance. During this phase, the system components are carefully designed, including the User Interface, Processing Module, Server, and Database, to ensure smooth interaction and data management. The interaction flow is planned to provide seamless communication between users and the chatbot, enabling efficient processing of user requests and AI-generated responses. To support real-time communication, WebSocket technology is implemented, allowing instant message exchange between the client and server with minimal latency.

Additionally, the database is designed to securely store user information, conversation history, and other essential data required for chatbot functionality and future analysis. This architecture ensures that the chatbot is scalable, efficient, and capable of handling multiple users while maintaining fast response times. The proposed system integrates Natural Language Processing techniques and language models for query understanding and response generation [3], [5].

V. SYSTEM DESIGN AND COMPONENTS

The proposed Virtual Assistant AI Chatbot is an intelligent conversational system designed to provide users with quick, accurate, and real-time assistance through both text and voice interactions. The chatbot leverages Natural Language Processing (NLP) and the Gemini API to understand user queries, identify their intent, and generate context-aware, human-like responses. By supporting voice input through speech-to-text conversion, the system enables natural and convenient communication, making it accessible to a wider range of users. The chatbot is designed to simplify interactions with web applications and digital platforms by delivering relevant information instantly and improving the overall user experience.

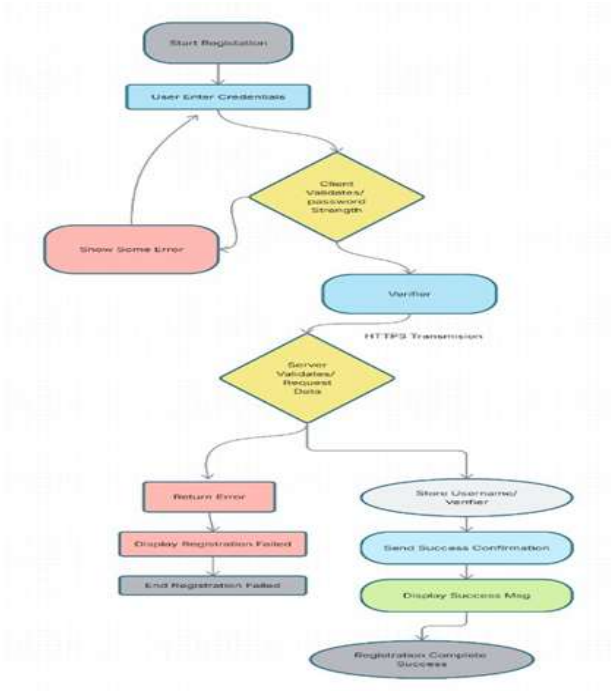
The proposed system is built on several key design principles to ensure high performance and usability. It emphasizes real-time interaction by utilizing WebSocket technology for low-latency, bidirectional communication between the client and server, allowing users to receive immediate responses. The system incorporates AI-driven intelligence through advanced NLP techniques and the Gemini API, enabling it to understand conversational context, interpret user intent, and generate meaningful responses. A scalable architecture is implemented using Node.js and Express.js, allowing the chatbot to efficiently handle multiple concurrent users without performance degradation.

To ensure data privacy and protection, the system adopts secure communication practices by implementing OAuth 2.0 for authentication and HTTPS/TLS encryption for secure data transmission. Additionally, the chatbot follows a user-centric design approach, providing an intuitive, responsive, and accessible interface that delivers a seamless experience across different devices. The architecture of the proposed system consists of three core components. The User Interaction Module serves as the interface between the user and the chatbot, capturing both text and voice inputs. Voice inputs

are converted into text using speech-to-text technology before being processed. Built with React.js and Tailwind CSS, the module provides a responsive and interactive user interface that works efficiently across various devices. User requests are transmitted to the server through WebSocket, enabling real-time communication and immediate feedback.

The AI Processing Module is responsible for analysing user inputs and generating intelligent responses. It utilizes the Gemini API to perform Natural Language Processing, understand user intent, extract relevant information, and produce accurate, context-aware, and human-like responses. Finally, the Backend and Communication Management Module, developed using Node.js and Express.js, manages application logic, API requests, and communication between the client and the AI service. By integrating WebSocket technology, the backend ensures continuous, real-time message exchange while efficiently handling multiple simultaneous requests, resulting in a fast, reliable, and scalable chatbot system

VI. PROCESS FLOWCHARTS AND DIAGRAMS



VII. TECHNOLOGY STACK

Backend: We used Node.js and Express.js to create APIs and also handle the logic on the server in our backend. We use WebSocket to allow client side and server side both talk each other in real time. That means we can send the response back to the client side very fast.

Frontend: We are using React.js in frontend for making a user interface. And this is dynamic mode and works well all device. Also used Tailwind CSS to make a styling easier and more well. We used Axios to send HTTP requests at a same time. And we use the Web Speech API to recognize voice and convert speech into text.

Database: We used a MongoDB where we can store user data and others information relate to user. And we designed database as a scalable and we can retrieve data quickly.

AI Integration: We used a Gemini API for Natural Language processing. And Gemini helps us to understand user input and gives the well output. The Gemini API was used for language understanding based on transformer-based Natural Language Processing concepts [3], [4].

Communication & Security Specifications: We used WebSocket for communication on real time. Also use the OAuth 2.0 to authentication and authorization user. We use the JSON for exchanging the data one formats to other formats between the client and server. We used the tokens to handles time sessions. Data upload on internet so we want its work safely to we can use the HTTP.

Security Hardening: The Security Hardening phase focuses on strengthening the overall security of the chatbot system to ensure safe and reliable operation. Security policies and controls are implemented to protect the application from unauthorized access and potential cyber threats. For secure authentication, the system uses OAuth 2.0, which provides a secure and standardized method for user authentication and authorization while effectively managing access control. To ensure encrypted

communication, all data exchanged between the client and the server is protected using HTTPS/TLS (Transport Layer Security). This encryption safeguards sensitive information during transmission, preventing unauthorized interception, data tampering, and other security threats. By implementing these security measures, the chatbot ensures the confidentiality, integrity, and protection of user data throughout the communication process.

User Interface Implementation

We use React.js for User Interface. This is easy to use and well structure. We design this chatbot run on all devices. We used many components to design this chatbot for user experience. User use chatbot step by step. Tailwind CSS used in frontend for making a responsive website. We add a WebSocket tool that help us up to date in real time. Chatbot interface design same as user want like easy to use with any extra sections.

Figure 7.1: Sign Up Interface

Description: User can Sign-up here entering by the credentials, with the secure and authentication and validation.

Figure 7.2: Login Interface

Description: User can Login here entering by the credentials and used the AI chatbot system.

Figure 7.3: Select your Assistant Image

Figure 7.4: Enter Your Assistant Name

Description: In the input section, you can enter the name of the AI.

Figure 7.5: AI Communication Interface

Description: This is the chatbot screen where users can ask question and receives answer through text and voice in the real time.

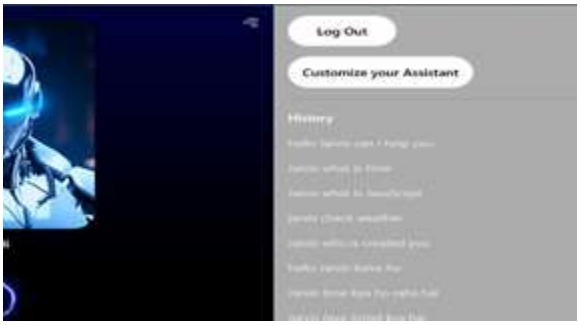


Figure 7.5: History

Description: User can check previous conversation of the History Section.

VIII. EVALUATION AND RESULTS

Performance Analysis: We evaluated the system based on the gives responds and how much accurate it. We know about the WebSocket, it is reduced delays compared to other communication methods. It is powered by AI and using Gemini API well-handled user queries [1].

System Efficiency Evaluation: The tests were done to check the system efficiency.

Metric	Traditional Chatbot	Proposed AI Chatbot	Improvement
Response Time	~500 ms	~200 ms	Faster
Real-Time Communication	Limited	Enabled (Webstock)	Improved
Query Accuracy	Moderate	High (AI based)	Enhanced
Scalability	Limited	High	Improved
User Interaction	Basic	Interactive (Voice + Text)	Enhanced

Security and Reliability Analysis: The system more confident that communication is safe by using OAuth 2.0 authentication and sending the data with HTTPS. System can also check what people want to search and put into it. And provide full security using many tools. The observed improvements in response quality are consistent with recent developments in transformer-based language models [3], [4].

IX. CONCLUSION

The chatbot is really good at showing how Artificial Intelligence and the internet work together and it make easy for people to talk to computers. It does this by using Natural Language Processing from the Gemini API. The Virtual Assistant AI Chatbot also talks to people in time using WebSocket. The Chatbot has a system that can handle a lot of work that means the Chatbot can give people answers that make sense and well structure. The Virtual Assistant AI Chatbot is easy to use because people can talk to Virtual Assistant AI Chatbot by both typing or speaking. The Virtual Assistant AI Chatbot is very quick responsive. Does not take long to answer and understand. Tests have shown this. This makes the Virtual Assistant AI Chatbot good for things, like helping customers being an assistant and finding information.

Future Work:

Advanced AI Model Enhancement: We will add some model to improvement in future our AI. This is helping the chatbot to understand the context better. It will also make conversations flow naturally in many languages our goal is to make a chatbot to gives a response very fast and accurate. When you use datasets to tune models the models can do certain jobs a lot better. This is really important for things that need to know a lot about a field. [6]

Multilingual Support: We will add more Multiple language then user can use chatbot in rural and urban.

Emotion and Sentiment Analysis: Chatbot talk with user and understand emotion and respond more naturally.

Voice Interaction Enhancement: System improved speech recognition and adds this feature text to speech, it will enable more natural voice-based conversations.

Mobile and Cross-Platform Integration: Developing a system for work on different mobile application and cross platform support.

Personalization and Learning: Implementing system for user behaviour tracking and adaptive learning can enable personalized responses and improved user experience, Future versions of the chatbot may incorporate Reinforcement Learning to improve adaptive conversational behaviour [6].

REFERENCES

1. A. Adamopoulou and L. Moussiades, "An Overview of Chatbot Technology," Artificial

Intelligence Applications and Innovations, Springer, 2020.
2. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
3. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL, 2019.
4. T. Brown et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.
5. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Pearson, 2023.
6. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.