IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 1, January 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |

Empowering Customer Support: Using Generative AI and Pre-trained LLM's in a Chatbot Revolution

Munesh Kumar B N, Rohini A, Armaan Shaik, Mohammad Basha Shaik, Mohammed Abrar

UG Student, Department of Computer Science and Engineering, Presidency University, Bengaluru, India

Assistant Professor, Department of Computer Science and Engineering, Presidency University, Bengaluru, India

UG Student, Department of Computer Science and Engineering, Presidency University, Bengaluru, India

UG Student, Department of Computer Science and Engineering, Presidency University, Bengaluru, India

UG Student, Department of Computer Science and Engineering, Presidency University, Bengaluru, India

ABSTRACT: This paper addresses the challenge of efficiently handling a diverse array of customer queries by proposing the development of an innovative web-based customer support chatbot. The objectives encompass creating a versatile system capable of interpreting and resolving a spectrum of customer complaints, enhancing support staff efficiency, and facilitating knowledge base updates. The proposed methodology employs the MERN stack for web app development and integrates Generative AI and pre-trained Large Language Models (LLMs), specifically OpenAI's pre-built models, for intelligent responses. The pseudo code outlines the interactions between the chatbot, support staff, and administrators, ensuring a seamless process. Results indicate the chatbot's commendable ability to accurately interpret queries, showcasing language versatility in various Indian and foreign languages. Efficient handling of varied phrasings and also accommodating grammatical errors. The use of prebuilt models is highlighted as a practical and efficient solution, reducing implementation time compared to custom models. Additionally, the implementation is recognized for its scalability, positioning it as a practical and efficient solution for evolving customer support needs in the near future.

KEYWORDS: Customer support chatbot, Large Language Models (LLMs), Generative AI, Scalability, Real-time communication, Multilingual support, AI assistants, MERN stack, Socket.IO

I. INTRODUCTION

As the world of web app development expands, so does the challenge of handling a growing number of customer queries. Traditional approaches, like hiring more support staff, are proving slow and expensive for companies. To tackle this, our project aims to revolutionize customer support by introducing a cutting-edge chatbot. This intelligent chatbot is designed to swiftly interpret and respond to customer queries by searching through a comprehensive database for solutions.

Our goal is to create a dynamic and accessible solution that not only caters to the increasing volume of queries but also ensures quick and cost-effective resolutions. The envisioned web app goes beyond typical customer support – when the chatbot encounters a query, it autonomously delivers a solution from its extensive database, ensuring a seamless experience. If a query exceeds the chatbot's capabilities, it seamlessly hands it over to human support staff for personalized assistance.

Key to our solution is the continuous updating of the database based on customer interactions, allowing the system to evolve and become more effective over time. This intersection of web app development and customer support promises not only efficiency and cost-effectiveness but also a forward-thinking approach to database management. In essence, our project seeks to redefine the customer support paradigm in the ever-evolving landscape of services and technology.

II. RELATED WORK

1. Existing Methods

In the exploration of conversational systems, various approaches have been proposed. One study introduced a chatbot incorporating Reinforcement Learning (RL) strategies to enhance user interactions. Employing sentiment analysis and Natural Language Processing (NLP), the chatbot gauges user moods, utilizing algorithms like DistilBERT for emotion

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |

detection, Tacotron2-ljSpeech for Text-to-Speech (TTS), and a combination of ChatGPT and BERT for financial advice.

Another paper presented a multifaceted NLP-based chatbot utilizing TensorFlow for image recognition, Flask for the backend, ReactJS for user interfaces, and Dialogflow for natural language understanding. The system employs a database (MongoDB/Firebase) and a deep learning model for processing customer queries, demonstrating continuous learning through ML and NLP engines.

A distinct approach involved the development of a banking-related chatbot named Jarvis using the Rasa Framework. The system's architecture encompasses backend, ML model, and frontend components. Rasa NLU and Rasa Core were implemented in the backend for user input handling, intent identification, and end-to-end conversation management. A separate ML model, utilizing Logistic Regression, achieved an 80% accuracy in predicting loans based on various parameters.

Conversational Recommender Systems (CRS) were a focus in another study, emphasizing the importance of understanding user intent for effective task-oriented systems. The proposed approach introduced a novel intent recognition method within an interactive pipeline, employing reverse feature engineering and contextualizing input utterances for enhanced intent recognition.

User authentication and input processing were central to a different investigation. The paper highlighted the utilization of AIML patterns for conversation mapping, keyword extraction from user input, and a response system incorporating NLP algorithms for sentence similarity. The system verified user authenticity, saved user data, and responded intelligently to user queries, with a particular focus on college-related information.

Finally, a study conducted at a Brazilian commercial bank's Analytical Intelligence Unit (AIU) focused on the integration of AI, particularly chatbots, for customer service. The research, based on 181 million interactions in 2020, utilized AI integrated with IBM's Watson system. The study highlighted the technological innovation in process management and the substantial contribution of AI, specifically chatbots, to customer service efficiency.

In summary, these diverse approaches encompass RL strategies, NLP-based chatbots, frameworks like Rasa, and innovative methods for intent recognition and user interaction, showcasing the multifaceted nature of current conversational system research.

2. Drawbacks

The implementation of conversational systems introduces various challenges. Privacy and bias concerns, alongside ethical dilemmas, underscore the need for responsible chatbot creation. Human intervention is crucial for training and optimizing chatbot systems, relying on NLP and ML algorithms for effective query analysis and response generation. Algorithmic dependencies, particularly with transformer models like BERT, pose potential computational challenges, scalability issues, and concerns about generalizability across diverse contexts. The reliance on predefined knowledge bases limits responses and scalability, while Rasa NLU faces challenges with insufficient or non-diverse training data.

Specific issues include the potential bias introduced by data content analysis, limitations in handling unseen texts or keywords, and challenges in generalizability due to small sample sizes. Ambiguities in terminology and difficulties in replicating human-like interactions highlight humanization challenges. Fixed rule-based approaches in current chatbots lead to adaptability challenges, grammatical errors, and limitations in supporting sentiment analysis and accuracy in dynamic conversations. Training and learning challenges, inadequate user interfaces, and limited language support further contribute to the comprehensive understanding of drawbacks in current conversational systems.

Adding to these challenges is the difficulty of acquiring and training chatbots on large datasets. The associated costs in terms of both money and time, as well as the expertise required for satisfactory results, present barriers for many businesses looking to minimize customer support workload. This emphasizes the need for cost-efficient methods in the market that provide satisfactory, accurate, and scalable results, offering an alternative to traditional resource-intensive approaches.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |

III. OBJECTIVES

The primary objective of this project is to address the surging influx of customer queries across diverse services in a highly efficient manner. In pursuit of this goal, the project endeavors to develop an innovative web-based customer support chatbot with multifaceted capabilities. The chatbot is designed to interpret a spectrum of customer queries and complaints, undertaking the crucial task of searching the database for viable resolutions and promptly delivering articulate responses. The overarching aim is to create a versatile system capable of handling a diverse range of inquiries, spanning from product-related questions to troubleshooting issues and general feedback.

Furthermore, the project encompasses the creation of distinct portals tailored for specific roles within the support ecosystem. One such portal is dedicated to customer care executives, providing them with a seamless channel to interact and communicate directly with customers or clients. This targeted approach aims to enhance the efficiency of support staff in addressing user queries. Another critical component is the portal designed for administrators, granting them the authority to review and curate content intended for updates in the knowledge base. This separate administrative interface ensures a meticulous and controlled process for knowledge base management.

The overarching goal is to deliver an intuitive and user-friendly chatbot that goes beyond basic query resolution. The system should be easily navigable, fostering a positive user experience while consistently offering helpful and informative responses to customer inquiries. These objectives collectively underscore the ambition to revolutionize the customer support process, providing not only efficiency and accessibility but also a robust framework for ongoing knowledge base enhancement.

IV. PROPOSED METHODOLOGY

The proposed methodology centers around the MERN (MongoDB, Express.js, React, Node.js) stack, a comprehensive technology suite renowned for building scalable and responsive applications. On the frontend, the project leverages JavaScript, HTML, and CSS, with the React framework ensuring a seamless and interactive user experience. This choice of technologies establishes a robust foundation for developing a modern and user-friendly customer support chatbot interface.

3. **Backend Architecture:**

Node.js is employed as the runtime for server-side operations, facilitating the execution of server-related tasks. Express.js is utilized to create RESTful APIs and manage WebSocket connections. The incorporation of WebSocket, facilitated by Socket IO Libraries, enables real-time, bidirectional communication, enhancing the responsiveness and immediacy of user interactions.

4. **Database Management:**

MongoDB is selected as the database solution, serving a dual purpose. Firstly, it efficiently stores and retrieves business-related data, utilizing its non-relational document database features that support JSON-like storage. Secondly, it maintains a record of chat history, encompassing user queries, summaries, and resolutions. MongoDB's flexible data model, combined with full indexing support and intuitive APIs, aligns seamlessly with the requirements of the chatbot application.

5. **Using Generative AI & Pre-trained LLM's:**

The core intelligence of the chatbot is enriched through the integration of Generative AI and pre-trained Large Language Models (LLMs), specifically leveraging OpenAI's offerings. Large Language Models are adept at understanding human speech and generating contextually appropriate responses based on their extensive knowledge base. Generative AI further augments this capability by enabling the system to create new content, such as text, by learning patterns from existing data through deep learning techniques.

A custom LLM with a Generative AI feature could be built to cater to our application needs, but there are few prebuilt ones like OpenAI assistant API run on well-trained GPT models that has a bigger knowledge base and has better understanding. We are using the gpt-3.5-turbo-1106 model for our application, since this is the updated and better trained model which gives better results than the other models. So a pre-built LLM models like this can be used instead of a custom built LLM as it is easier to use, provides satisfactory results, and is also easy to integrate.

International Journal of Innovative Research in Computer and Communication Engineering

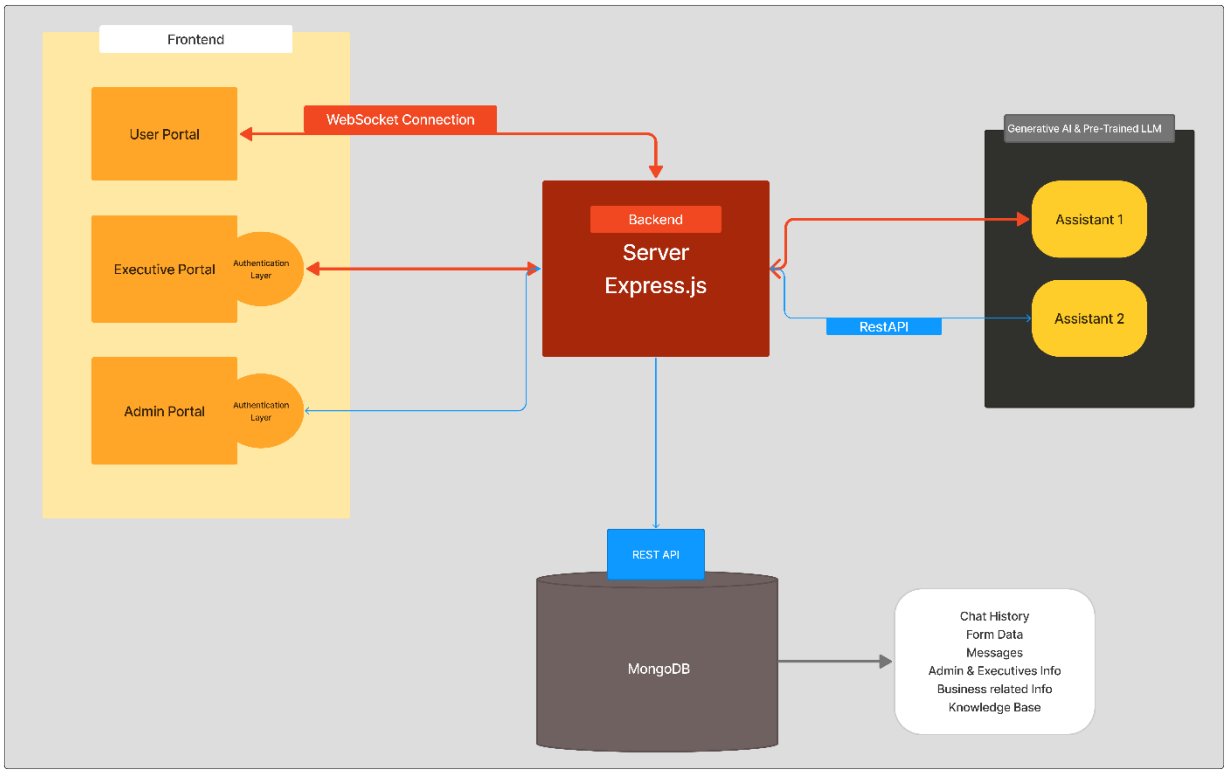
| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

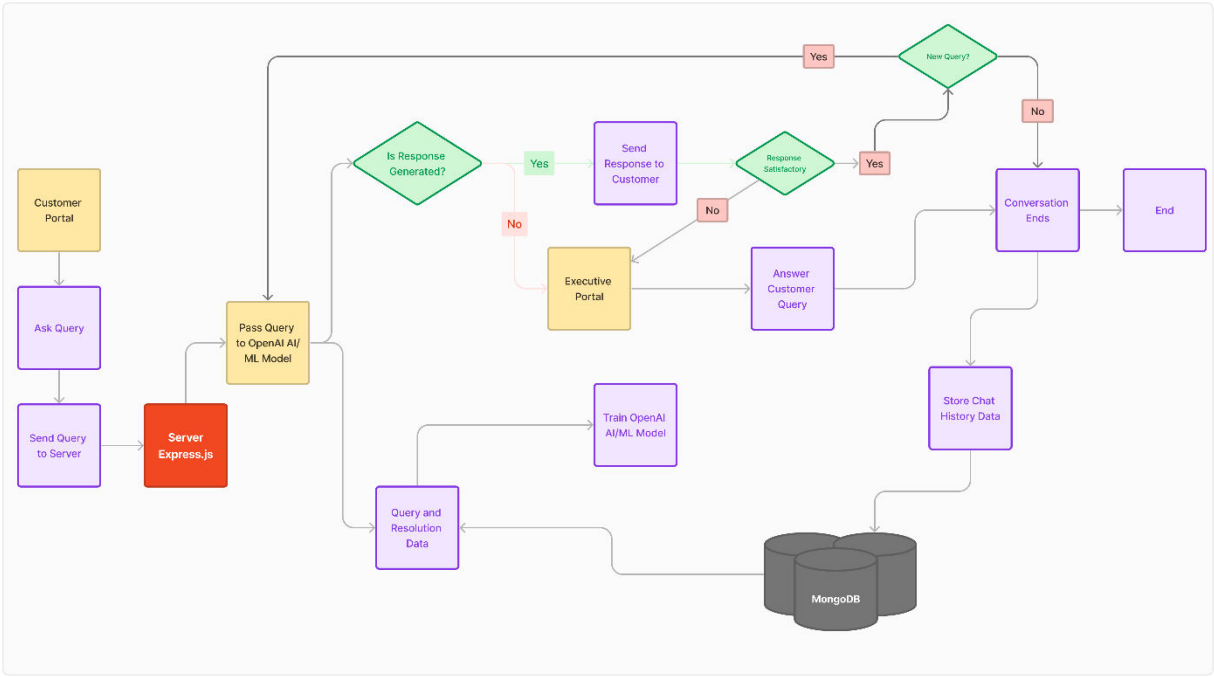
| DOI: 10.15680/IJIRCCE.2024.1201020 |

V. SYSTEM DESIGN

1. Architecture Design



2. Flowchart

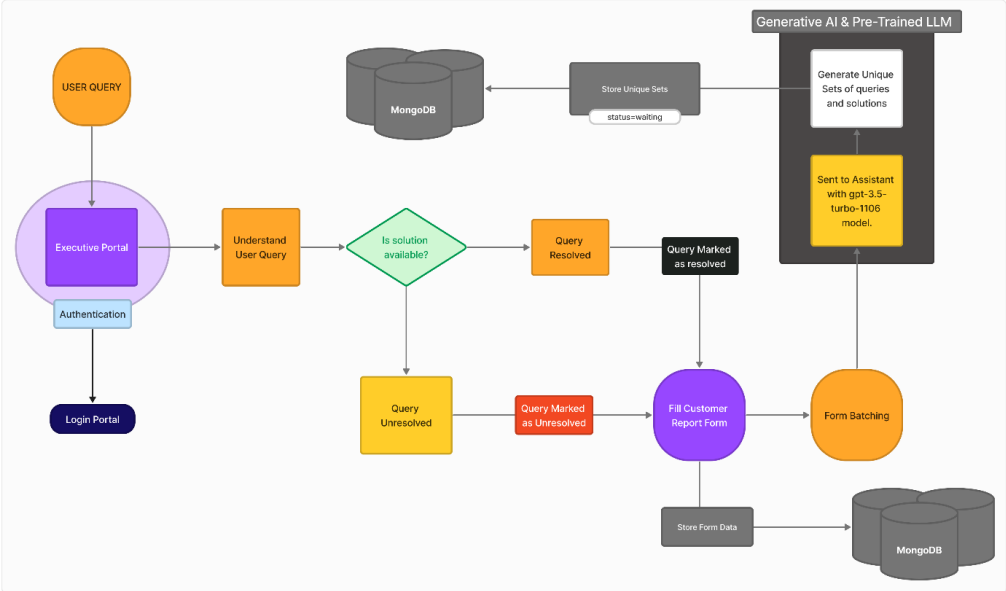


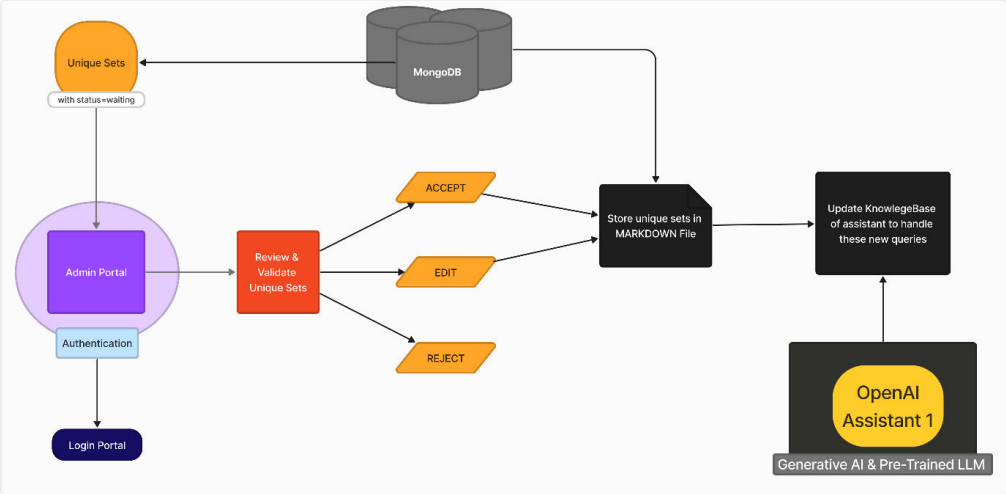
International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |





VI. IMPLEMENTATION

1. Backend

1.1 Node.js and Express.js:

The server-side operations are powered by Node.js, providing a JavaScript runtime environment. Express.js, a robust web framework for Node.js, is utilized to create REST APIs and maintain WebSocket connections. This combination ensures the efficient handling of incoming requests and seamless communication between the server and the application's components.

1.2 API and WebSocket Layers:

The backend is designed with distinct API and WebSocket layers. The API layer manages traditional HTTP requests, including authentication functionalities such as login and logout. WebSocket connections, facilitated by the Socket.IO library, play a pivotal role in establishing full-duplex and bidirectional communication. Socket.IO enhances real-time interactions, allowing for immediate updates and responses.
These layers are actively listening for any incoming requests and then upon receiving any request directs it to the appropriate section in the services layer.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |

1.3 Services Layer:
Upon receiving requests, the backend directs them to the services layer, the business logic segment of the application. This layer s responsible to act upon the incoming data from the requests. Within this layer, it utilizes the models in the database layer, precisely the repository layer to execute essential functionalities based on the data derived from the requests originating from API or WebSocket layers. Once the data undergoes processing within this layer, the refined information is then returned to its point of origin(where it was called). This modular structure ensures flexibility and maintainability, facilitating the seamless flow of information between the backend and other components.

1.4 Authentication with Bcrypt:
To ensure data security and privacy, the backend incorporates the Bcrypt library and its password hashing algorithm. This enables secure password storage and authentication processes, enhancing the overall robustness of the customer support chatbot.

1.5 WebSocket Connections with Socket.IO:
WebSocket Connection Handling:
The WebSocket connections are crucial for enabling real-time communication within the chatbot. When a user opens the chatbot, the frontend initiates a WebSocket connection with the server through the Socket.IO library. This establishes a persistent and efficient channel for bidirectional communication.

Event Handling:
The server is equipped to handle various WebSocket events, enhancing the overall functionality of the chatbot. The connection event is triggered when a new client connects, providing the opportunity to initialize necessary processes. The message event is crucial for processing user messages, storing them in the MongoDB database, and interacting with the AI model to formulate responses. The disconnect event is triggered when a client disconnects, allowing for proper cleanup and maintenance of active connections.

Message Exchange Workflow:
The message exchange workflow involves a seamless process from user interaction to server response. When a user sends a message, the frontend transmits it through the WebSocket connection to the server. The server, in turn, processes the message by storing it in the MongoDB database, consulting the AI model if needed, and constructing a response. This response is then broadcasted to all connected clients, ensuring that both the sender and others receive immediate updates.

Additional Features (Optional):
In addition to fundamental chat functionalities, the WebSocket connections allows us to incorporate optional features to enhance user experience. Presence detection is implemented to track online users and display their status. Typing indicators and file sharing features contribute to a more dynamic and engaging communication environment. These optional features further contribute to the versatility and adaptability of the customer support chatbot.

2. Database

2.1 MongoDB for Efficient Data Handling:
MongoDB is employed as the database solution, effectively storing and retrieving various business-related data. This includes chat history, messages, form data submitted by support staff, FAQs, and solutions. MongoDB's flexibility and scalability make it well-suited for handling the continuous updates and diverse data types associated with customer interactions.
Few models are created in the DB namely admins, executives, chat history, form data, Knowledge Base (FAQ's, Product details, etc), messages. There are a few joins made between these model collections in order to prevent duplication.

2.2 Real-time Data Management:
WebSocket connections facilitate real-time data management within the database, ensuring that information is promptly updated based on user interactions and solutions. This feature contributes to the adaptability and responsiveness of the customer support chatbot.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |

3. Generative AI and Pre-trained LLM's

The customer support chatbot leverages the OpenAI Assistant API to build AI assistants within the application. Two OpenAI assistants are utilized: one for generating responses to user queries and another for analyzing chat history and form data submitted by support staff.

The second assistant employs pre-trained Large Language Models (LLM's) to generate a unique set of queries and solutions. These are then sent to the Admin portal for verification and validation, allowing administrators to edit, accept, or reject queries and solutions. Accepted data is appended to the dataset, enabling the Assistant to adapt and respond effectively to user queries using the updated information.

Upon updating the dataset and passing it to the Assistant, the system autonomously chunks documents, indexes and stores embeddings, and employs vector search to retrieve relevant content for addressing user queries. The model dynamically determines when to retrieve content based on user messages, ensuring a responsive and context-aware chatbot.

4. Frontend

4.1 User, Executive, and Admin Portals:

The frontend is structured into three distinct portals: User, Executive (support staff), and Admin. Users interact with the chatbot through the User portal, while support staff and administrators access the Executive and Admin portals, respectively. Each portal is equipped with its authentication layer to ensure secure access and data protection.

Once logged in, support staff can converse with users, fill out forms with user query summaries and the proposed solutions given by them to the users based on the conversation between them and the user. These forms, transmitted via WebSockets, make their way back to the backend and ultimately into the MongoDB database.

But the journey doesn't end there. In batches, these forms are fed to the second OpenAI assistant, equipped with the power of large language models (LLMs). This assistant meticulously analyzes the chat history and form data, meticulously extracting knowledge and crafting unique sets of queries and solutions. These AI-generated gems are then presented to the admin for careful verification and validation allowing administrators to edit, accept, or reject queries and solutions. Accepted entries are integrated into the main dataset, empowering the first assistant to provide even more insightful responses in the future.

4.2 Seamless Transition and Socket Connections:

The frontend allows users to seamlessly transition from the assistant in the chatbot in the User portal to engaging with support staff in the Executive portal. This transition involves the disconnection of the chatbot and AI assistant and the establishment of a new connection between the chatbot and executive. Both interactions utilize Socket.IO connections, providing a smooth and responsive communication channel.

At its core, the implementation is a well-coordinated team of technologies. Databases act as reliable memory banks, storing past conversations and data. Frameworks act as the conductors, directing the flow of information and ensuring smooth operation. AI assistants, infused with intelligence, provide insightful responses and guidance. WebSockets, like communication highways, seamlessly connect users, staff, and AI in real-time conversations. Meanwhile, large language models analyze and refine knowledge, continuously transforming it into improved solutions. This dynamic collaboration lies at the heart of the customer support chatbot, constantly learning and adapting to serve users better with each interaction.

VII. RESULTS AND DISCUSSION

The implemented chatbot's assistant demonstrates a commendable ability to accurately interpret user queries, utilizing vector search and indexing for precise knowledge base retrieval. The dynamic generation of responses proves to be accurate and satisfactory, facilitated by the retrieval function of the assistant. Leveraging the OpenAI GPT-3.5-turbo-1106 model empowers our chatbot to effectively handle grammatical errors, incomplete queries, and varied phrasings, ensuring consistently accurate results. Moreover, the chatbot showcases remarkable language versatility, proficiently responding to queries in multiple languages, including regional Indian languages and foreign languages.

This choice of utilizing a prebuilt model enhances efficiency, facilitates easy integration, and provides ample room for additional features, making it a favorable option over custom-built Large Language Models (LLM's) with NLP algorithms. The implementation time is notably reduced compared to custom LLMs, establishing the prebuilt model as a practical and efficient solution for our chatbot.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |

VIII. CONCLUSION

In summary, our project aimed to tackle the surge in customer queries by creating an advanced web-based customer support chatbot. The chatbot, with its diverse capabilities, effectively interprets and responds to a wide range of customer inquiries. The development includes specific portals for customer care executives and administrators, ensuring targeted interactions and streamlined knowledge base management.

Using the MERN stack, we established a user-friendly chatbot interface. Technologies like Node.js, Express.js, React, and MongoDB enabled real-time communication through WebSocket connections, enhancing responsiveness. Incorporating Generative AI, particularly the OpenAI GPT-3.5-turbo-1106 model, strengthened the chatbot's ability to handle grammatical errors and respond in multiple languages. Our implementation showcases a well-coordinated integration of databases, frameworks, AI, and WebSocket connections, allowing the chatbot to learn and adapt continuously. The results demonstrate the chatbot's commendable accuracy in interpreting queries and providing satisfactory responses. Opting for a prebuilt model not only improved efficiency but also simplified integration. Overall, our project successfully revolutionizes customer support, offering efficiency, accessibility, and continuous knowledge base enhancement.

IX. FUTURE SCOPE

The future scope of this project involves scaling its capabilities through the integration of additional models from the OpenAI API library. This includes incorporating Text-to-Speech (TTS) functionality for narrating written content in multiple languages and providing real-time audio output. Speech-to-Text capabilities, utilizing the Whisper model, allow for transcription and translation of audio, ensuring multilingual support.

Expanding beyond text-based and voice based interactions, the integration of Image Generation with GPT-4 and Vision enables the chatbot to analyze and respond to questions about images. Can also guide users to navigate through the websites and products by generating images. Answer product related queries with images.

To handle larger datasets more efficiently, the project aims to explore modern vector analytical and schema-free databases like ElasticSearch. These additions aim to elevate the chatbot's performance, versatility, and overall functionality making this approach a much more scalable one.

REFERENCES

[1] Shivom Agarwal, Shourya Mehra, Pritha Mitra (Quantum Dynamics SAS Marseille , France) "Multi-Purpose NLP Chatbot : Design, Methodology & Conclusion**",** Oct 2023 -https://arxiv.org/abs/2310.08977

[2] Hiral Paghadal, Anezka Virani, Apratim Shukla, Dr. G T Thampi "Customer Support Chatbot Leveraging Machine Learning", May 2020 – https://www.irjet.net/archives/V7/i5/IRJET-V7I5217.pdf

[3] Anas Khan, Masood Khan, Rakesh Utekar "Customer Support Chatbot using NLU", May 2021 - https://www.irjet.net/archives/V8/i5/IRJET-V8I5526.pdf

[4] Sahar Moradizeyveh "Intent Recognition in Conversational Recommender System", Dec 2022 - https://arxiv.org/abs/2212.03721

[5] Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Shreya Bisen, Vasundhara Rathod "Implementation of a Chatbot System using AI and NLP", May 2018 https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3531782

[6] Patchara, Vanichvasin "Chatbot Development as a Digital Learning Tool to Increase Students' Research Knowledge", Jan 2021 - https://eric.ed.gov/?id=EJ1284721

[7] Chiara Valentina Misischia, Flora Poecze, Christine Strauss "Chatbots in customer service: Their relevance and impact on service quality" March, 2022 -https://www.sciencedirect.com/science/article/pii/S1877050922004689

[8] Ivan Martins De Andrade, Cleonir Tumelero "Increasing customer service efficiency through artificial intelligence chatbot", July 2022 - https://www.emerald.com/insight/content/doi/10.1108/REGE-07-2021-0120/full/html

[9] Lei Cui , Chuanqi Tan , Shaohan Huang ,Chaoqun Duan "SuperAgent: A Customer Service Chatbot for E-commerce Websites" , January 2017 - https://www.researchgate.net/publication/318738996_SuperAgent_A_Customer_Service_Chatbot_for_E-commerce_Websites

[10] Soufyane Ayanouz, Boudhir Anouar Abdelhakim, Mohammed Benhmed "A Smart Chatbot Architecture based NLP and Machine Learning for Health Care Assistance", April 2020 - https://dl.acm.org/doi/10.1145/3386723.3387897

[11] OpenAI Documentation – https://platform.openai.com/docs/overview

[12] Elastic-Search Documentation - https://www.elastic.co/guide/index.html

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201020 |

[13] Data Tuning Preperation and analysis for chat model fine tuning- https://cookbook.openai.com/examples/chat_finetuning_data_prep
[14] Assistant Tools: Knowledge Retreival -https://platform.openai.com/docs/assistants/tools/knowledge-retrieval
[15] Assistants: How assistants work- https://platform.openai.com/docs/assistants/how-it-works
[16] Best File format recommended for assistant api retrieval - https://community.openai.com/t/whats-the-best-file-format-for-recommendation-by-using-[1] assistant-api/556705
[17] Socket.IO Documentation- https://socket.io/docs/v4/
[18] Bcrypt NPM package- https://www.npmjs.com/package/bcrypt
[19] Getting started with axios: Docs- https://axios-http.com/docs/intro
[20] React DOM Documentation- https://legacy.reactjs.org/docs/react-dom.html
[21] Cross Origin Resource Sharing (CORS) Guide- https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details