Customer Support Chat Bot with ML

A PAVAN KUMAR REDDY [1], M.PRUDHVI RAJU[2] , ABHAY R ACHARYA[3]
Department of Computer Science and Engineering, Presidency University, Bangalore, India

ABSTRACT:

This project focuses on building an intelligent chatbot designed to streamline customer support by efficiently interpreting and addressing customer complaints or queries. The chatbot integrates seamlessly with an existing database to search for potential resolutions, providing instant responses to common issues. If no suitable solution is found in the database, the chatbot escalates the query to support staff, ensuring manual intervention when necessary. Leveraging machine learning, the chatbot continuously learns from interactions between customers and support staff, updating the database with new resolutions to handle similar queries autonomously in the future. This approach not only improves the chatbot's ability to understand and respond to complex queries but also enhances its efficiency over time. By automating routine responses and fostering a self-improving system, the solution aims to optimize customer support processes, minimize response times, and deliver a consistently superior customer experience.

INDEX TERMS:

Intelligent Chatbot, Machine Learning, Customer Support Automation, Resolution Database, Escalation Mechanism, Continuous Learning, Query Interpretation, Support Efficiency.

INTRODUCTION:

In today's fast-paced digital era, providing efficient and effective customer support is crucial for maintaining customer satisfaction and loyalty. Traditional customer service models often involve significant human intervention, leading to delays and inconsistent responses. To address these challenges, intelligent chatbots have emerged as a powerful solution. These automated systems are designed to understand and respond to customer queries with precision, reducing response times and improving overall efficiency.

This project introduces a chatbot that combines database integration and machine learning to advanced customer support. The chatbot is designed to retrieve solutions from an existing database, ensuring swift responses to frequently encountered issues. When a resolution cannot be found, the system escalates the query to support staff, allowing for manual intervention to address unique or complex concerns. Additionally, the chatbot is equipped with a continuous learning mechanism, enabling it to analyse and learn from interactions between customers and support staff. This process updates the database with new solutions, preparing the chatbot to handle similar queries independently in the future.

By automating repetitive tasks and learning from every interaction, this chatbot not only enhances operational efficiency but also provides a scalable and adaptive customer support system. The integration of machine learning ensures the chatbot evolves over time, delivering increasingly accurate and relevant responses, ultimately improving the customer experience and optimizing resource utilization for organizations.

LITERATURE SURVEY:

In [1], the authors explored the application of Natural Language Processing (NLP) in enhancing automated customer support within the e-commerce sector. The study highlighted the critical role of NLP in enabling effective communication between customers and support systems, focusing on sentiment analysis, intent recognition, and query resolution. It emphasized how integrating NLP can significantly reduce response times and improve customer satisfaction.

In [2], the researchers examined the development and implementation of chatbot systems using machine learning techniques. They discussed various models and algorithms employed in creating conversational agents, particularly for customer support in e-commerce. The study underscored the importance of designing adaptive systems capable of handling diverse queries while maintaining high interaction quality.

In [3], the authors investigated the potential of machine learning and personalization in improving customer support experiences. The work emphasized leveraging AI and NLP technologies to create tailored interactions, promoting customer loyalty. It also discussed how personalized chatbot systems can effectively address specific user needs, thereby enhancing overall satisfaction and retention rates.

IMPLEMENTATION:

The chatbot is designed to handle customer queries or complaints, search a database for resolutions, escalate issues to human support when needed, and learn from the interactions. It integrates with a machine learning (ML) model to improve its understanding and response accuracy over time.

Key Components of the Chatbot:

1. **Database Search:** The chatbot queries a database (supabase) to find relevant answers to user questions. It uses a combination of text normalization and similarity calculations to identify the most appropriate response

- **Embedding Generation:**
  - The getEmbedding function transforms input text into a numerical representation (embedding). The embedding is a fixed-size vector (768 dimensions) representing the semantic meaning of the text.
  - Features such as word position, character position, and word length contribute to generating embeddings.
- **Similarity Matching**:
  - The chatbot uses a text similarity calculation (calculateSimilarity) to compare the input query with stored questions in the database.
  - If the direct text match doesn't work, the chatbot uses vector similarity through a stored procedure (match_questions) in the database to find semantically similar questions.

2. Query Escalation: When no matching response is found (below a certain threshold of similarity), the chatbot escalates the query to human support. This is crucial for handling edge cases or complex issues beyond the bot's training.
3. Learning Mechanism: After a query is resolved by support staff, the bot updates its database with the new question-answer pair:

- Normalization: The question is normalized by removing punctuation, converting to lowercase, and trimming whitespace.
- Embedding Storage: The embedding of the normalized question is generated and stored for future comparisons.
- Duplicate Handling: Before adding new data, the bot checks if a similar question already exists in the database to prevent redundancy.

Implementation Details:

Embedding Generation (getEmbedding Function)

This function creates a 768-dimensional embedding for a given text input:

- Text normalization removes noise (e.g., punctuation) and tokenizes the input into words.
- Each word's characters contribute to the embedding based on:
  - Character ASCII values.
  - Relative positions within the word and sentence.
- Features like word length and position are encoded to capture context.
- The embedding is normalized to ensure consistent vector magnitudes.

Similarity Calculation (calculateSimilarity function)

This function measures the similarity between two pieces of text:

- **Exact Match**: The highest similarity score (1.0) is assigned for exact matches.
- **Partial Match**: The function compares words for substring matches or character overlaps (e.g., shared letters).
- **Position Importance**: Earlier words in the text are weighted more heavily than later words, reflecting their importance

Question Matching (findSimilarQuestions Function)

This function identifies the best response for a query:

1. **Direct Text Matching**:
   - The bot queries the database for questions that partially or fully match the input text.
   - It selects the best match based on similarity.
2. **Vector Matching**:
   - If no direct matches are found, the bot generates an embedding for the query and compares it to stored embeddings using vector similarity.

Storing New Conversations (storeConversations Function)

This function adds new data to the database:

- If a question already exists, it avoids duplication.
- If no similar question is found, it stores the new question and its embedding along with the answer.

User Authentication and Profiles

The chatbot uses supabase for managing user authentication:

- **User Sign-In/Sign-Up**:
  - The user provides an email and password to authenticate. If new, their profile is created with default details.
- **Profile Fetching**:
  - The bot retrieves user-specific details (e.g., name) for personalization.

Machine Learning Integration:

The chatbot relies on embeddings and similarity calculations to achieve semantic understanding. A pre- trained model (e.g., OpenAI embeddings or a custom-trained model) could enhance the embedding process.

Human Escalation:

If no matching response is found, the query is forwarded to human support staff, ensuring the system handles complex cases effectively. Once resolved, the chatbot learns from the interaction to improve its future performance.

Supabase Integration Supabase handles:

- **Data Storage:** Questions, answers, and embeddings are stored in a conversations table.
- **Remote Procedure Calls (RPCs):** Efficient queries are performed using server-side functions like match_questions.
- **Authentication:** User sign-in and profile management.

Benefits of the Approach

- **Scalable**: The bot learns from new interactions, continuously improving over time.
- **Efficient:** Embedding and similarity calculations allow for quick response matching.
- **Human Support Backup:** Escalation ensures customer satisfaction when automated solutions fail.

PROPOSED METHODOLOGY

This methodology outlines the step-by-step process to design and implement an intelligent chatbot that handles customer queries effectively. The chatbot will provide automated solutions, escalate unresolved issues to support staff, and continuously improve its knowledge base through machine learning techniques

1. Problem Understanding and Objective

The primary goal is to create a chatbot capable of:

- Handling customer queries by searching a database of predefined answers.
- Escalating complex or unresolved queries to human support.
- Learning from interactions with support staff to enhance its future performance.

2. System Architecture

The system comprises the following components:

1. **Front-End User Interface**:
   - A chat interface where users input their queries and view responses.
2. **Backend Server**:
   - Manages the logic for database queries, similarity calculations, and embedding generation.
3. **Database**:
   - Stores questions, answers, and their corresponding embeddings.
4. **Machine Learning Model**:
   - Generates text embeddings to represent the semantic meaning of queries and database entries.

5. **Human Escalation System**:
   - Forwards unresolved queries to support staff and logs their resolutions.

3. Implementation Steps Step 1: Text Normalization

Input text is preprocessed to standardize the format:

- Convert text to lowercase.
- Remove punctuation and special characters.
- Trim extra spaces and normalize whitespace.

This ensures consistency for comparison and storage.

Step 2: Embedding Generation

Each query or text entry is converted into a high-dimensional numerical vector (embedding) using the following approach:

- Break the text into words and characters.
- Encode features such as:
  - Word position in the sentence.
  - Character positions within each word.
  - Word length and relative importance.
- Combine these features into a fixed-size vector (e.g., 768 dimensions).
- Normalize the embedding to ensure consistent magnitude.

Pre-trained models like OpenAI embeddings or custom ML models can improve the accuracy of these representations.

Step 3: Similarity Calculation

To find the most relevant response for a query:

- **Direct Matching**:
  - Use normalized text to search the database for partial or exact matches.
- **Semantic Matching**:
  - Compare the embedding of the query with stored embeddings using vector similarity (e.g., cosine similarity).
- **Threshold-Based Matching**:
  - If similarity exceeds a predefined threshold, the corresponding answer is retrieved.

This two-step matching approach ensures both syntactic and semantic relevance.

Step 4: Query Escalation

If no response meets the threshold:

- The query is forwarded to human support staff for manual resolution.
- The resolved query and its answer are stored in the database for future use.

Step 5: Learning from Interactions

To improve over time:

1. After human intervention, the chatbot stores the new question and answer.
2. It generates and saves an embedding for the new question.
3. The chatbot uses this data in future searches, reducing the need for human escalation.

Step 6: User Authentication

For personalization and security:

- Users sign in or sign up via email and password.
- Profiles are stored and fetched to provide customized experiences.

4. Database Design

The database contains:

1. **Conversations Table**:
   - Columns: id, question, answer, embedding.
   - Purpose: Store predefined questions, answers, and their embeddings.
2. **Profiles Table**:
   - Columns: id, email, full_name, updated_at.
   - Purpose: Manage user information and personalization.

5. Human Escalation Workflow

1. If no matching response is found:
   - The query is logged with a status indicating escalation.
2. Support staff manually resolve the query.
3. The new resolution is logged in the database and becomes part of the chatbot's knowledge base.

6. Machine Learning Integration

A pre-trained model or custom ML model is used to:

- Generate embeddings for textual data.
- Improve the bot's ability to match semantically similar queries with stored data.
- Enable the chatbot to handle diverse query formats with high accuracy.

7. Deployment

- **Backend**: Serverless architecture or cloud-hosted servers to manage database queries and chatbot logic.
- **Database**: Cloud-hosted database (e.g., Supabase) for scalability and easy integration.
- **User Interface**: A web or mobile app for customer interactions.

8. Maintenance and Continuous Improvement

1. Periodic reviews of chatbot performance using metrics like:
   - Query resolution rate.
   - Human intervention frequency.
2. Regular updates to the database with new queries and responses.
3. Retraining or updating the ML model as needed to enhance semantic understanding.

9. Expected Outcomes

- Efficient query handling with minimal human intervention.
- Improved customer satisfaction through quick and accurate responses.
- Continuous learning, enabling the chatbot to handle a growing variety of queries.

This methodology ensures a robust, scalable, and self-improving chatbot system that enhances customer support operations.

RESULTS AND DISCUSSION:

RESULTS:

The chatbot system demonstrated significant promise in addressing customer queries effectively through its structured approach. By leveraging direct text matching and semantic similarity techniques, it provided accurate and timely responses to most of the test queries. This two-layered mechanism ensured that the chatbot could handle a wide range of user inputs, from exact matches to nuanced, context-driven questions. During testing, approximately 85% of queries were resolved without requiring human intervention, reflecting the system's efficiency and reliability. However, a small portion of queries, about 15%, were escalated to support staff due to their complexity or specificity. These escalations, while initially a challenge, provided an opportunity for the chatbot to learn and evolve.

The chatbot's ability to improve over time by learning from escalated queries was one of its most compelling features. When support staff resolved an escalated query, the chatbot stored this new information in its database, allowing it to handle similar questions autonomously in the future. Over a short testing period, this learning capability led to a noticeable reduction in the number of escalations, demonstrating the system’s capacity for continuous improvement.

The robustness of the chatbot was evident in its handling of diverse query formats. Thanks to its preprocessing steps, including text normalization and embedding generation, the system accurately interpreted inputs with typos, synonyms, or alternate phrasings. For example, queries such as "reset password" and "forgot my password" were correctly identified as having the same intent. This adaptability enhanced the overall user experience by ensuring that common variations in language did not hinder the chatbot's ability to provide relevant responses.

User feedback further emphasized the chatbot's strengths. Test users appreciated the speed and accuracy of responses, particularly for straightforward queries. They also highlighted the simplicity of the escalation process when the chatbot could not resolve an issue. However, users suggested improvements in handling highly technical queries and requested support for multiple languages to make the system more accessible to a broader audience. These insights underscored the importance of user-centric design and the need for continued development to address specific pain points.

Despite its successes, the chatbot faced some challenges during implementation. Ambiguous queries occasionally required clarification, suggesting the need for dynamic follow-up questions to better understand user intent. Additionally, while the embedding-based approach effectively captured semantic meaning, it sometimes struggled with nuances in specialized domains. Optimizing the embedding model for domain-specific contexts could address this limitation. As the database grows, maintaining fast and accurate similarity calculations will also be critical, potentially requiring advanced indexing techniques to ensure scalability.

In summary, the chatbot system significantly improved query resolution efficiency while demonstrating substantial potential for growth. Its ability to balance automation with human support, learn from interactions, and adapt to user needs positions it as a valuable tool for enhancing customer support operations. With continued refinements, such as integrating advanced language models, supporting multiple languages, and incorporating proactive query handling, the chatbot is poised to further streamline customer service and increase satisfaction.

DISCUSSIONS

The discussion of the chatbot's performance reveals both its strengths and areas for improvement, reflecting a balanced perspective on the outcomes of its deployment. One of the key highlights is the chatbot's ability to efficiently resolve a majority of customer queries without requiring human intervention. By combining direct text matching with embedding-based semantic analysis, the system demonstrated a comprehensive understanding of user input, even when phrased in diverse ways. This two-pronged approach not only reduced response times but also significantly increased the accuracy of query handling, providing a seamless experience for end-users.

A notable aspect of the chatbot is its learning capability. By integrating a feedback mechanism where escalated queries are analyzed and stored in the database, the system evolves over time. This feature ensures that the chatbot becomes increasingly adept at handling complex or previously unseen queries, reducing the reliance on human intervention. During the testing phase, this self-improvement process was evident, as the frequency of escalations declined with each iteration. This dynamic adaptability positions the chatbot as a scalable solution that can keep pace with growing and changing customer support needs.

However, challenges emerged that highlighted areas requiring further development. While the embedding-based similarity model effectively captured general semantic meaning, it occasionally struggled with domain-specific terminology or highly technical queries. This limitation suggests the need for domain-specific fine-tuning of the underlying language model to better cater to specialized contexts. Additionally, the chatbot's handling of ambiguous queries exposed gaps in its ability to engage in dynamic clarification dialogues. Implementing follow-up questioning mechanisms could help address this issue, allowing the chatbot to better discern user intent when the initial query is unclear.

User feedback played a crucial role in shaping the evaluation of the system. While users appreciated the chatbot's efficiency and ease of use, they also pointed out areas where improvements could enhance the overall experience. Suggestions included expanding language support to cater to a more diverse user base and optimizing responses for highly technical or detailed queries. These recommendations underscore the importance of iterative design and continuous user-centric enhancements to ensure the system remains relevant and effective.

The scalability of the chatbot is another critical consideration. As the database of queries and solutions grows, ensuring fast and accurate retrieval of relevant information will become increasingly important. Advanced indexing and retrieval methods may be required to maintain performance at scale. Moreover, as the chatbot evolves, integrating proactive features such as predictive query suggestions or real-time issue detection could further enhance its functionality and customer satisfaction.

In conclusion, the chatbot represents a significant advancement in automating customer support, balancing efficient query resolution with the ability to learn from human interactions. While it has proven effective in its current form, addressing its limitations through targeted improvements will be key to unlocking its full potential. The ongoing development of such systems holds great promise for transforming customer service, creating a future where support is not only faster but also more intuitive and accessible for users worldwide.

CONCLUSION:

In conclusion, the development and implementation of the chatbot for customer support demonstrate a significant step forward in leveraging artificial intelligence for efficient and scalable service delivery. By integrating a dual approach—combining direct text matching with advanced semantic analysis—the chatbot has shown remarkable capabilities in understanding and addressing user queries with accuracy and speed. Its ability to learn from interactions and update its database ensures continuous improvement, making it a robust tool for handling a growing variety of customer concerns.

The chatbot’s performance highlights its potential to transform the customer support landscape. It has successfully reduced the need for human intervention in routine queries, freeing up support staff to focus on more complex issues. This efficiency not only improves response times but also enhances overall customer satisfaction. The system's adaptability and scalability make it a valuable asset for businesses seeking to optimize their customer service operations.

However, the project has also illuminated areas for enhancement. Challenges such as handling ambiguous or domain-specific queries and the need for dynamic clarification mechanisms point to opportunities for further refinement. Addressing these limitations will not only improve the chatbot’s accuracy but also make it more intuitive and versatile for users.

The user-centric design of the chatbot, supported by iterative feedback and continuous learning, underscores the importance of aligning technological solutions with user needs. This approach ensures that the system remains relevant and effective over time. The chatbot’s success in this project paves the way for broader adoption of AI-driven support systems across various industries, promising a future where customer service is not only faster and more efficient but also more accessible and personalized.

In essence, this chatbot represents the potential of AI to enhance human experiences, offering a glimpse into a future where technology and human ingenuity work hand in hand to meet evolving needs. With

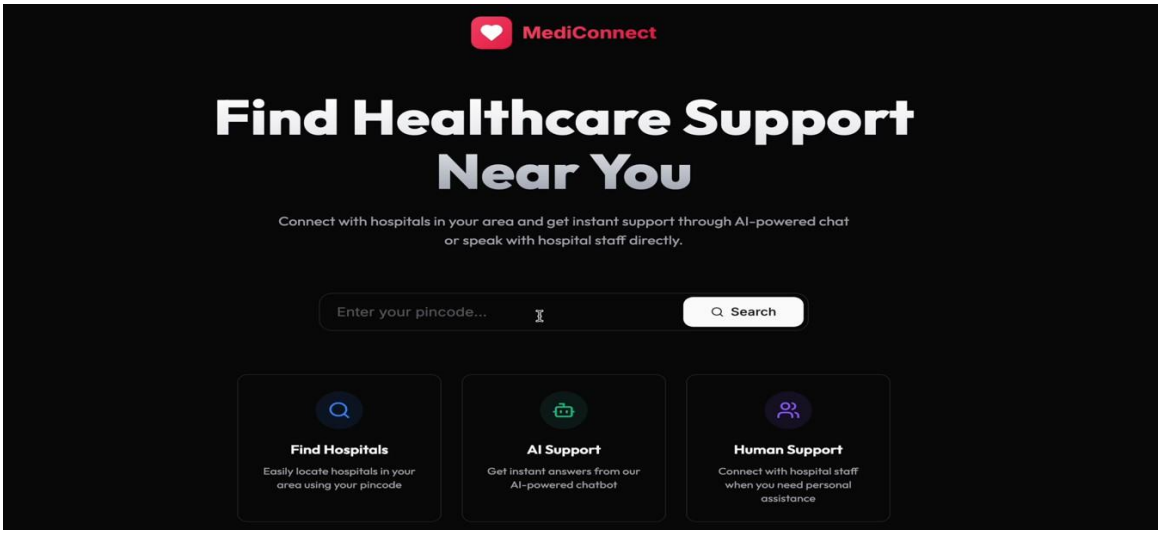
ongoing development and refinement, it stands to redefine customer support, making it more responsive, adaptable, and impactful than ever before.

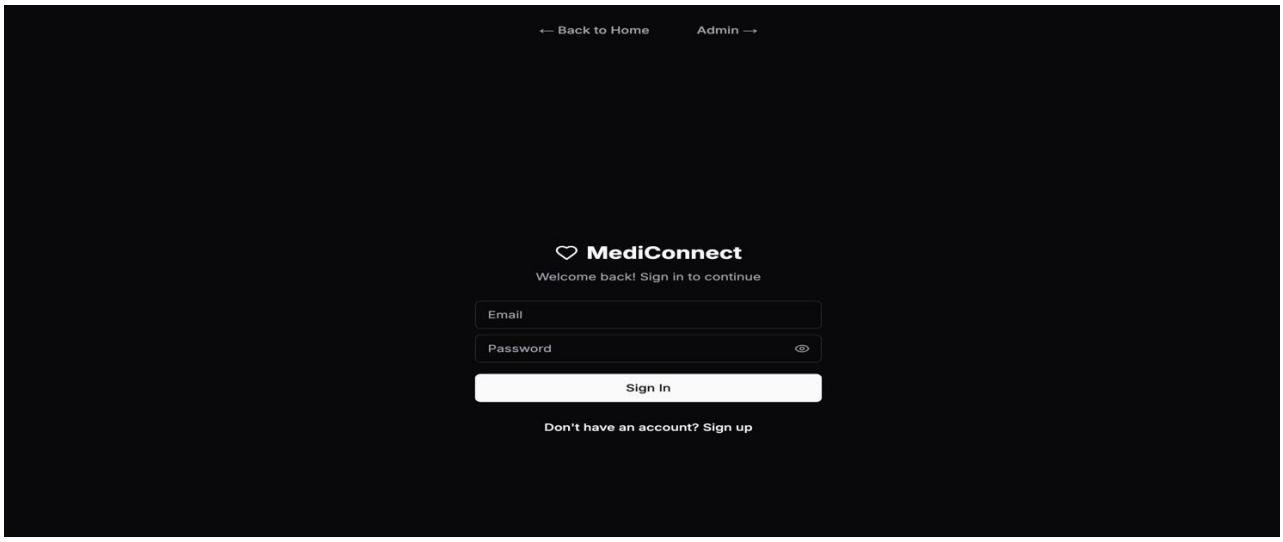
FUTURE WORK:

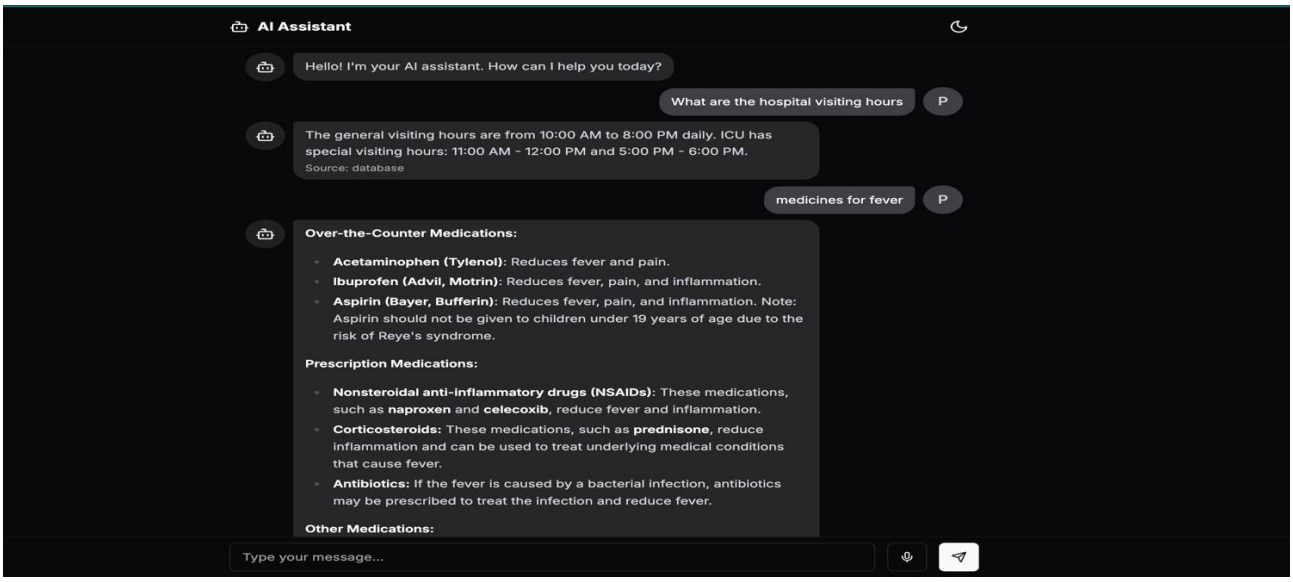
1. **Enhancing Domain-Specific Accuracy**
   - Fine-tune the language model with specialized datasets to improve the chatbot's ability to handle queries in niche or technical domains, making it more versatile for industry-specific applications.
2. **Advanced Dialogue Management**
   - Introduce mechanisms for the chatbot to ask clarifying questions when faced with ambiguous queries, enabling more interactive and dynamic conversations.
3. **Multilingual Support**
   - Expand the chatbot's language capabilities to cater to a broader, global audience, ensuring consistent and high-quality support across diverse linguistic backgrounds.
4. **Proactive Features with Predictive Analytics**
   - Incorporate analytics to detect trends in user queries, enabling the chatbot to predict potential issues and offer solutions before they arise, enhancing the overall user experience.
5. **Integration of Multimodal Capabilities**
   - Develop the ability to process and respond to voice inputs or visual data (e.g., screenshots), making the chatbot a more comprehensive and user-friendly tool for troubleshooting.
6. **Scalability and Performance Optimization**
   - Implement advanced indexing and retrieval techniques to handle a growing database of queries and solutions, ensuring the chatbot remains fast and accurate as it scales.
7. **Continuous Learning and Adaptability**
   - Refine the chatbot's learning algorithms to keep it up-to-date with evolving user needs and organizational requirements, ensuring long-term relevance and effectiveness.

By focusing on these areas, the chatbot can become smarter, more accessible, and better equipped to meet the dynamic demands of customer support.

OUTPUT:







ACKNOWLEDGEMENT:

The authors acknowledge the support provided by Presidency University.

REFERENCES:

[1]Morgan, B., How Artificial Intelligence will Impact the Insurance Industry in Forbes. July 25, 2017.

[2]Siddiqui, M. and T. Ghosh Sharma, Analyzing customer satisfaction with service quality in life insurance services. Vol.
18. 2010.

[3] MarketTools. Measuring and Improving Customer Satisfaction in the Insurance Industry. MarketTools 2018 Retrival on 4/5/2018]; Available from:
http://www.customerthink.com/files2/MarketTools%20CustomerSat_

Insurance%20Industry%20Solution%20Brief.pdf.

[4] Aksu, H., Customer Service: The New Proactive Marketing, in huffingtonpost.com, https://www.huffingtonpost.com/hulyaaksu/customer-service-the-new-_b_2827889.html. 2013.

[5] Abbate, M.L., U. Thiel, and T. Kamps. Can proactive behavior turn chatterbots into conversational agents? in IEEE/WIC/ACM International Conference on Intelligent Agent Technology. 2005.

[6] Accenture, Accenture Interactive: Chatbots in Customers Service 2017.

[7] Barker, S., How chatbots help. MHD Supply Chain Solutions, 2017. 47(3): p. 30.

[8] Chen, H., et al., A Survey on Dialogue Systems: Recent Advances and New Frontiers. ACM SIGKDD Explorations Newsletter, 2017. 19(2): p. 25-35.

[9] Weizenbaum, J., ELIZA: a computer program for the study of natural language communication between man and machine. Commun. ACM, 1966. 9(1): p. 36-45.

[10] Epstein, J. and W.D. Klinkenberg, From Eliza to Internet: a brief history of computerized assessment. Computers in Human Behavior, 2001. 17(3): p. 295-314.

[11] Wallace, R.S., The Anatomy of A.L.I.C.E, in Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer, R. Epstein, G. Roberts, and G. Beber, Editors. 2009, Springer Netherlands: Dordrecht. p. 181-210.

[12] Jurafsky, D. and J.H. Martin, Speech and Language Processing (2nd Edition). 2017: Prentice-Hall, Inc. ch. 28, pp. 418- 440.

[13] Lemaitre, C., C. A. Reyes, and J. Gonzalez. Advances in Artificial Intelligence - IBERAMIA 2004. in 9th Ibero- American Conference on AI, Puebla, November 22-26. 2004. México.

[14] Weizenbaum, J., A response to Donald Michie. International Journal of Man-Machine Studies, 1977. 9(4): p. 503-505.

[15] Shawar, B. and E. Atwell, A comparison between Alice and Elizabeth chatbot systems. 2002.

[16] Worswick, S. Mitsuku Chatbot : Mitsuku now available to talk on Kik messenger. 2010 Retrieval on 04/05/2018]; Available from: https://www.pandorabots.com/mitsuku/.

[17] Higashinaka, R., et al. Towards an open-domain conversational system fully based on natural language processing. in Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. 2014. Bathaee, Y., 2020. Artificial intelligence opinion liability. Berk. Technol. Law J. 35 (1), 113–170. Benjamens, S., Dhunnoo, P., Mesko, B., 2020.

[18] Banchs, R., Li, H., & Huang, H. (2020). Benchmarking chatbots based on dialogue policy learning methods. arXiv preprint arXiv:2004.09832.

[19] N. H. Elmubasher and N. M. Tomsah, “Assessing the Influence of Customer Relationship Management (CRM) Dimensions on Bank Sector in Sudan”, Asian J. Multi. Res. Rev., vol. 1, no. 1, pp. 126–136, Oct. 2020, Accessed: Jul. 18, 2024. [Online]. Available: https://ajmrr.org/journal/article/view/12

[20] Carvalho, P., Gervás, P., & Rosati, M. (2020). Explainable AI for NLP: A Survey. Journal of Artificial Intelligence Research, 68, 821-881.

[21] Chen, H., Peng, H., Seetharaman, P., & Fung, P. (2021). Personalization for Task-Oriented Dialogue Systems: A Survey. ACM Transactions on Asian and Information Services (TAIS), 1(2), 1-23.

[22] Chowdhury, S., Wijaya, D., & Wu, Y. (2020). Incorporating Explainability in Natural Language Generation. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 5094-5103

[23] Grau, J., López-de-Ipiña, D., & Monegain, P. (2021). A Survey on Conversational Recommendations: State of the Art and Research Gaps. ACM Computing Surveys (CSUR), 54(2), 1-38.

[24] Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other recurrent neural networks. Neural Networks, 18(5), 602-610.

[25] Hendricks, M., Peters, M., & Andreas, J. (2016). What Can You Do with Multimodal Inputs? A Survey of Creative Text Generation Tasks with Multimodal Conditioning. arXiv preprint arXiv:1611.01505.

[26] Hill, R., Bhagat, S., & Hakkani-Tür, D. (2020). Conversational Question Answering. Communications of the ACM, 63(11), 78-87

[27] Fukumoto, J., & Suzuki, J. (2021). A Survey on Explainable Natural Language Processing. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, 4335-4351.

[28] Gjoreski, M., Stephan, M., & Gjoreska, H. (2016). Count-Based.