**CUSTOMER SUPPORT CHATBOT WITH MACHINE LEARNING**

## A PROJECT REPORT

***Submitted by,***

**20211CAI0081-VIDHYASHREE V**

**20211CAI0094-SALAPAKSHI SAGAR**

**20211CAI0162-CHALLA YOGESH**

**20211CAI0198-TALLA SUNIL KUMAR**

**20211CAI0200-GADE PRATHYUSHA**

### *Under the guidance of,*

**Mr.Afroz Pasha**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING.**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**DECEMBER 2024**

**PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**CERTIFICATE**

This is to certify that the Project report **“Customer Support Chatbot using Machine Learning”** being submitted by “VIDHYASHREE V,SALAPAKSHI SAGAR, CHALLAYOGESH, TALLASUNILKUMAR, GADEPRATHYUSHA” bearing roll number(s)“20211CAI0081,20211CAI0094,20211CAI0162,20211CAI0198,20211CAI0200” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

|  |  |
| --- | --- |
| **Mr.Afroz Pasha**  Assistant Professor  School of CSE&IS  Presidency University | **Dr. Zafar Ali Khan N**  Associate Professor& HoD  School of CSE&IS  Presidency University |

|  |  |  |
| --- | --- | --- |
| **Dr. L. SHAKKEERA**  Associate Dean  School of CSE  Presidency University | **Dr. MYDHILI NAIR**  Associate Dean  School of CSE  Presidency University | **Dr. SAMEERUDDIN KHAN**  Pro-Vc School of Engineering  Dean -School of CSE&IS  Presidency University |

**PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **Customer Support Chatbot using Machine Learning** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **COMPUTER SCIENCE ENGINEERING SPECIALIZATION IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** , is a record of our own investigations carried under the guidance of

**Mr.Afroz Pasha, Assistant Professor,**

**School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

|  |  |
| --- | --- |
|  | **Name(s)Roll num(s): VIDHYASHREE (20201CAI0081)**  **SALAPAKSHISAGAR(20201CAI0094)**  **CHALLA YOGESH (20201CAI0162)**  **TALLA SUNIKUMAR(20201CAI0198)**  **GADEPRATHYUSHA(20211CAI0200)** |

**ABSTRACT**

In the domain of customer support, businesses continually seek ways to improve the speed, efficiency, and quality of service provided to customers. Traditional customer support methods, which primarily rely on human agents, can be costly and often struggle to meet customer expectations in terms of response times, availability, and scalability. This is where machine learning-powered chatbots come into play.A chatbot is an intelligent system which can hold a conversation with a human using natural language in real time. Due to the rise of Internet usage, many businesses now use online platforms to handle customer inquiries, and many of them turn to chatbots for improving their customer service or for streamlining operations and increasing their productivity. However, there is still a gap between existing chatbots and the autonomous, conversational agents businesses hope to implement. As such, this paper will first provide an overview of chatbots and then focus on research trends regarding the development of human-like chatbots capable of closing this technological gap. We reviewed the literature published over the past decade, from 1998 to 2018, and presented an overview of chatbots using a mind-map. The research findings suggest that chatbots operate in three steps: understanding the natural language input; generating an automatic, relevant response; and, constructing realistic and fluent natural language responses. The current bottleneck in designing artificially intelligent chatbots lies in the industry’s lack of natural language processing capabilities. Without the ability to properly understand the content and context of a user’s input, the chatbot cannot generate a relevant response. The customer support chatbot, as a significant player in this sector, is tasked with offering compensation, legal guidance, and support to those affected by vehicular incidents . In recent years, the increasing demand for efficient and personalized customer support has driven organizations like the customer support chatbot to explore innovative solutions to cater to evolving customer needs . This literature review paper aims to provide a comprehensive overview of the existing research and implementations in the realm of developing customer support chatbots within the insurance and compensation sector . Specifically, the focus is on the context of the customer support chatbot and its pursuit of leveraging machine learning technologies to enhance the customer support experience. Customer support chatbots have evolved significantly, transforming from rule-based systems to sophisticated AI driven agents capable of understanding and addressing a diverse array of customer inquiries. As organizations seek ways to improve customer interactions and streamline support processes, chatbots have emerged as a promising avenue. These AI powered systems are designed to provide timely and accurate information to customers, thereby enhancing their overall experience. Within the domain of insurance and compensation, the customer support chatbot is poised to benefit from the adoption of chatbot technology, which can cater to queries related to accident claims, compensation processes, legal requirements, and general inquiries . By reviewing the existing literature and research, this paper aims to shed light on the evolution of customer support chatbots, the machine learning techniques employed in their development, the challenges faced, and the best practices adopted. The ultimate goal is to establish a solid foundation for the development of a customer support chatbot tailored to the unique requirements of the insurance and compensation sector . This study recognizes the potential of chatbots to not only improve customer satisfaction but also enhance the efficiency of support processes, contributing to a more streamlined and effective ecosystem for motor vehicle accident compensation.

**ACKNOWLEDGEMENT**

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer ScienceEngineering& Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair,**School of Computer Science Engineering& Information Science, Presidency University, and Dr. ANANDARAJ S P, Head of the Department, School of Computer Science Engineering& Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Mr. Afroz Pasha** and Reviewer **Dr./Mr.Ms. KOKILA S**, School of Computer Science Engineering, Presidency University for his/her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A and Mr. Md Zia Ur Rahman,** department Project Coordinators and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**VIDHYASHREE (20201CAI0081)**

**SALAPAKSHI SAGAR(20201CAI0094)**

**CHALLA YOGESH (20201CAI0162)**

**TALLA SUNIL KUMAR(20201CAI0198)**

**GADE PRATHYUSHA(20211CAI0200)**

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Figure Name** | **Caption** | **Page No.** |
| 1  2 | Figure 1.1  Figure 1.2 | **System Components and Block Diagram**  **Results and Discussion** | 5  8 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGENO.** |
|  | **ABSTRACT ACKNOWLEDGMENT**  **LIST OF FIGURES** | **ii** |
| 1. | **INTRODUCTION** | 1 |
|  | 1.1 Introduction to the domain of the Problem Statement chosen: |  |
|  | 1.1.1 Types of Domains |  |
|  | 1.1.2. The impacts of predictive modelling, machine learning personalization, AI chatbots, and NLP on customer loyalty |  |
|  |  |  |
|  |  |  |
| 2. | **LITERATURE REVIEW** | **11** |
|  | 2.1 Advantages of ML-powered Customer Support Chatbots |  |
|  | 2.2. Challenges in Implementing ML-powered Chatbots for Customer Support |  |

3 3.1 **PROPOSED METHODOLOGY**17

4. 4.1 **OBJECTIVE 19**

6. 6.1 **BLOCK DIAGRAM AND COMPONENTS 21**

7. 7.1 TIMELINE

8. 8.1 **OUTCOMES** 25

9. 9.1 **RESULTS AND DISCUSSIONS**27

10. 10.1 **CONCLUSION** 28

11. 11.1 **REFERENCES** 29

11.1.1 **APPENDIX A** :PSUEDOCODE 30

**CHAPTER-1**

**INTRODUCTION**

In the domain of customer support, businesses continually seek ways to improve the speed, efficiency, and quality of service provided to customers. Traditional customer support methods, which primarily rely on human agents, can be costly and often struggle to meet customer expectations in terms of response times, availability, and scalability. This is where machine learning-powered chatbots come into play.A chatbot is an intelligent system which can hold a conversation with a human using natural language in real time. Due to the rise of Internet usage, many businesses now use online platforms to handle customer inquiries, and many of them turn to chatbots for improving their customer service or for streamlining operations and increasing their productivity. However, there is still a gap between existing chatbots and the autonomous, conversational agents businesses hope to implement. As such, this paper will first provide an overview of chatbots and then focus on research trends regarding the development of human-like chatbots capable of closing this technological gap. We reviewed the literature published over the past decade, from 1998 to 2018, and presented an overview of chatbots using a mind-map. The research findings suggest that chatbots operate in three steps: understanding the natural language input; generating an automatic, relevant response; and, constructing realistic and fluent natural language responses. The current bottleneck in designing artificially intelligent chatbots lies in the industry’s lack of natural language processing capabilities. Without the ability to properly understand the content and context of a user’s input, the chatbot cannot generate a relevant response. The customer support chatbot, as a significant player in this sector, is tasked with offering compensation, legal guidance, and support to those affected by vehicular incidents . In recent years, the increasing demand for efficient and personalized customer support has driven organizations like the customer support chatbot to explore innovative solutions to cater to evolving customer needs . This literature review paper aims to provide a comprehensive overview of the existing research and implementations in the realm of developing customer support chatbots within the insurance and compensation sector . Specifically, the focus is on the context of the customer support chatbot and its pursuit of leveraging machine learning technologies to enhance the customer support experience. Customer support chatbots have evolved significantly, transforming from rule-based systems to sophisticated AI driven agents capable of understanding and addressing a diverse array of customer inquiries. As organizations seek ways to improve customer interactions and streamline support processes, chatbots have emerged as a promising avenue. These AI powered systems are designed to provide timely and accurate information to customers, thereby enhancing their overall experience. Within the domain of insurance and compensation, the customer support chatbot is poised to benefit from the adoption of chatbot technology, which can cater to queries related to accident claims, compensation processes, legal requirements, and general inquiries . By reviewing the existing literature and research, this paper aims to shed light on the evolution of customer support chatbots, the machine learning techniques employed in their development, the challenges faced, and the best practices adopted. The ultimate goal is to establish a solid foundation for the development of a customer support chatbot tailored to the unique requirements of the insurance and compensation sector . This study recognizes the potential of chatbots to not only improve customer satisfaction but also enhance the efficiency of support processes, contributing to a more streamlined and effective ecosystem for motor vehicle accident compensation.

**1.1 Introduction to the domain of the Problem Statement chosen:**

**1.1.1 Types of Domains:**

* **Machine learning (ML)**-powered chatbots present an innovative solution to these customer support challenges. By leveraging natural language processing (NLP) and ML, these chatbots can provide automated, real-time assistance to customers, offering responses that feel natural and contextually appropriate.
* **Automated Responses to Common Queries:** An ML-powered chatbot can be trained on frequently asked questions (FAQs) and other common customer queries, enabling it to provide instant responses without the need for human intervention
* **Intent Recognition:** Using ML algorithms, the chatbot can identify the purpose or “intent” behind each customer query. For example, a customer asking, "Where’s my order?" might trigger the chatbot to access order-tracking information.
* **Contextual Understanding:** Unlike rule-based bots, ML-powered chatbots can understand the context within a conversation. They can retain information shared earlier in the interaction, making the conversation more natural and efficient.
* **Sentiment Analysis:** Chatbots can analyze the sentiment behind a customer's message, helping to determine when a query may require escalation to a human agent or when the customer is frustrated, enabling more empathetic responses.
* **Personalized Responses:** By using customer data, chatbots can offer personalized responses and recommendations, making the support experience more relevant and enhancing customer satisfaction.

**1.1.2 The impacts of predictive modelling, machine learning personalization, AI chatbots, and NLP on customer loyalty:**

* . **Predictive modelling** :Prescriptive modeling is the use of large data, mathematics, and modeling approaches to create future predictions. It examines both current and historical data to identify trends in order to predict the possibility of that very same behavior occurring once more in the future. Prescriptive modeling is a proactive strategy as opposed to a reactive one. Knowing what potential consumers want or purchase next may provide businesses with a major competitive edge. Predictive analytics may help businesses improve personalization, promote new business and loyalty, and improve client satisfaction.
* **Client Loyalty Evaluation** :The higher the retention rate, the more consumers redeem their incentives, and hence the greater the proportion of loyal customers (Wassouf et al., 2020). Assessing engagement ratio or involvement level is similar in that it shows the proportion of consumers who are actively participating in the loyalty program (Wassouf et al., 2020). This will assist business in determining if its present brand loyalty items are successful offers or not.
* **Chatbots loyalty**: Customers appreciate speedier service provided by chatbots, but a human touch may be the perfect addition. Bots have shown to be effective agents for answering simple inquiries. They are configured with basic principles and AI logic to replicate human written speech. As customer requirements and preferences shift and restructure the corporate structure, practically all clients want client service agents to be available 24 hours a day, seven days a week. If administered by a human operator, this is almost impossible. Chatbots are succeeding in this circumstance by offering the appropriate assistance and service . A human-AI hybrid may transform the customer service, with people totally focused on harnessing AI's speed, efficiency, and reach, and the chatbot trained to provide 100% customer pleasure by addressing their concerns on time.
* **Answering customer questions** :This has been the most basic use of chatbots for small enterprises. A firm would not perform well if it does not include that as one of ways to leverage the chatbot for client acquisition. Bots serve as the initial point of interaction between consumers and businesses. This is especially important if business operate in a sector with a lot of terminology; there is a good probability that people may contact customer services with simple queries.
* **Multilingualism and chatbot:** There are several languages in the globe, and everyone, quite plainly, likes to speak their native language. As a result a company may make better usage chatbot by developing them to carry on a conversation. One would not expect all of business clients to share a common language if they were dispersed all across world. As a result, companies can plan for the clients to be able to connect to the business in their home dialect or first language. This allows the consumers to connect with business more easily. They will be much more inclined to contact the customer service department. This simplicity of interaction boosts your capacity to keep consumers. The company's client retention will improve with multilingual chatbots. As a result, business should ensure that language does not hinder the progress company want to achieve.
* **ML personalization** :In recent years, there has been a tremendous growth in the usage of digital technology to provide individualized experiences. It is now feasible to provide unique material suited to each customer's interests and demands, making suggestions simple and straightforward. A component of machine learning is the creation of algorithms that can provide content and relevance recommendations to clients, much as human speech / language specialists do with actual people. Machines, unlike real speech / language specialists, lack emotions, stereotyping, and biases. Machine learning entails developing recommendation engines that can access a broad range of data resources, including private data from clients, external corporate data, and websites. Recommendation engines enable businesses to create a relevant, tailored experience for every consumer, regardless of their previous purchase patterns or preferences. Retail giants are increasingly using recommendation systems to offer tailored shopping experiences for specific consumers. Customers generally welcome these suggestions, but they are particularly successful with millennial consumers, who are more inclined to buy things they see promoted on media or in magazines. Machine learning is being used by the organizations involved to create a recommender system that combines the most of all previously acquired external data, integrates it with analytical and NLP techniques, and improves the resultant suggestions.

**CHAPTER-2**

**LITERATURE SURVEY**

There are two main approaches to generate a chatbot response. If the chatbot uses a traditional approach then it is hard-coded rule-based templates and rules that process for responses. However, nowadays many new and interesting approaches allow deep learning approaches to emerging. Neural network models are trained with several data to study the process of generating responses that are grammatically relevant and in terms of user speech intent [6]. These are some previous works related to chatting systems or chatbot for universities. Online Chatting System for college inquiries using a knowledgeable database by Bathe, Malusare, and Kolpe is doing pattern matching to perform information retrieval on chatbots. It has not already used the deep learning approach, so that is still too rigid by rules for users to make inquiries. But, detailed working steps in this research are very good, there such a UML, and various kinds of process diagrams [7]. Erasmus the AI chatbot, Erasmus is a chatbot on Facebook which is used to answer the queries related to the college information. Designing end-to-end systems using cloud services, starting from api.ai (Dialogflow), Mlab (MongoDB cloud), IBM Bluemix (webhook API), scrapper import.io to minimize the scripting code process, but what's happening here, it is all because there are too many diverse cloud services, there is quite a long latency between the services [8]. Then, Eaglebot a Chatbot Based Multi Tier Question Answering System for Retrieving Answers from Heterogeneous Sources Using BERT. It is a scalable chatbot system using 3 route selection methods with the main framework using Dialogflow, then several completion methods from document retrieval and document reader [9]. Eaglebot which serves to answer various questions commonly asked by students in the university domain. However, the Eaglebot application still has some limitations by Dialogflow chatbot framework which has some request limits, if it wants to unlimit requests you need to subscribe for the member privileged. Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network [10] by Anran Jiao was comparing performance between RASA NLU stack with Neural Network Classifier and Entity Extractor from scratch. This research was told that the RASA NLU method still superior to extract all the entities and classify user speech intent. It’s also quite comprehensive research but this research only depends on a free API for response completion and not using its own database. Compared to all those previous works, this research will be more reliable by using deep learning for chatbot natural language understanding and combining free API with it’s own database for response completion. So it won’t need for pattern matching, not using cloud provider service-based, and not only depends on a free API service so that customization will be easier, scalable and no limit request for user interaction. Facebook users in Indonesia August 2020. There were 166.500.000 Facebook users in Indonesia in August 2020, which accounted for 60.8% of its entire population. The majority of them were men - 54.6%. People aged 25 to 34 were the largest user group 59.000.000. The highest difference between men and women occurs within people aged 25 to 34, where men lead by 9.000.000. So that this Chatbot uses the Facebook platform as the user interaction medium.

Existing Methods

* Rule-based Systems
* Supervised Machine Learning Models
* Natural Language Processing (NLP) Models
* Deep Learning Models

**Chatbots’ customer-related functions and their influence on service quality** ,recognize in chatbots an emerging digital marketing strategy, which companies are increasingly implementing in order to adapt to the growing digitally oriented service world. Focusing on the marketing sector, five chatbots’ central functions were identified, i.e. interaction, entertainment, trendiness, customization and problem solving, which can also be interpreted as chatbots’ customer-related functions. Based on the these, a study was conducted, revealing a positive correlation between them and the chatbot’s communication accuracy and credibility, as between the latter and customer satisfaction. Similarly, other studies have likewise approached customer satisfaction as the subject of their research, however in relation to service quality. Based on the results of their investigations, a positive relationship between the two variables could be established here as well [10,21,22]. Based on these studies, this work will consider the five above-mentioned chatbots’ functions from a customer care perspective, elaborate on them and enlighten how they affect service quality. As a result of an in-depth literature review, so far undetected commonalities between the five customer-related functions were uncovered by the authors of this work and for this reason subsequently divided into two categories: “improvement of service performance” and “fulfillment of customer’s expectations”. These are intended to represent the chatbots’ objectives, which in turn serve to achieve the chatbots’ final functional goal of enhancing service quality. An overview of the determination of the chatbots’ objectives’ categories based on scientific literature can be found.

**Fulfillment of customer’s expectations** : The remaining two chatbots’ customer-related functions, trendiness and customization, were clustered together due to their ability to meet customer expectations. The term trendiness is understood as a part of people’s lifestyle as well as the extent to which they see themselves being involved in the latest trends, in order to pursue the goal of strengthening their social identity [31]. Especially today’s era of social media allows consumers to have a wider access to a wealth of information and a greater control over content consumption than ever before. Thus, meanwhile, the extensive range of social media competes with the traditional advertising, being considered by consumers as a more trustworthy source for making purchasing decisions. Indeed, in relation to aspects such as customer’s opinions, attitudes, purchasing behaviour and information gathering, they seem to be a major influencing factor [32]. According to Muntinga et al. [33], information provides an important motivation for people to consume brand-related content, as it covers four sub- motivations: Surveillance, knowledge, pre-purchase information, and inspiration. Firstly, surveillance involves observing one’s own social environment and examining what is currently popular, trendy or in vogue. Secondly, knowledge implies benefiting from others’ expertise by retrieving and consuming brand or product-related information. Third, pre-purchase information helps consumers in making a well thought purchase decision by analysing product reviews. Fourth and finally, inspiration refers to the brand-related information that people consume as a source of inspiration. This means that consumers choose to get inspired and influenced by products that other people wear or use, in order to get a feeling of what is currently in and thus obtain social recognition. For these reasons, chatbots rely on technologies which allow them to provide the user, during or beyond the conversation, with high quality information about newly designed products or emerging trends. In this way, the new customer requirements may be met and the service quality enhanced. From a customer care perspective, care should be taken not to ignore this social phenomena and to design chatbots in a way that allows them to best meet the newly developed trendiness expectations.

**The primary objectives of this study are to**:

* Evaluate the efficacy of ML-driven chatbots and virtual assistants in customer support roles.
* Compare the performance of these advanced technologies with traditional customer support methods.
* Assess customer satisfaction and response efficiency when interacting with ML-driven systems.
* Identify the benefits and limitations of using machine learning in customer support.

**2.1. Advantages of ML-powered Customer Support Chatbots:**

* **Improved Response Times:** Chatbots can respond instantly to customer inquiries, reducing waiting times and improving the customer experience.
* **24/7 Availability:** Chatbots are available round-the-clock, allowing customers to get support whenever they need it, even outside of regular business hours.
* **Handling High Volume:** ML-powered chatbots can manage a large number of interactions simultaneously, addressing a significant limitation of traditional customer support.
* **Cost Savings:** By automating routine tasks, chatbots can help reduce the cost of customer support, enabling businesses to focus human resources on more complex or valuable tasks.
* **Scalability:** As businesses grow, chatbots can scale up to handle more interactions without requiring significant additional investment.
* **Continuous Improvement:** Through machine learning, these chatbots can continually improve over time. They learn from customer interactions, allowing them to refine their responses and better handle diverse queries.

**2.2.Challenges in Implementing ML-powered Chatbots for Customer Support**

* While ML-powered chatbots offer many benefits, there are also challenges to consider:
* **Data Requirements:** High-quality training data is essential for effective chatbot performance. Insufficient or poor-quality data can lead to inaccurate responses and a subpar user experience.
* **Complex Queries:** While AI chatbots are proficient with common questions, they may struggle with highly complex, ambiguous, or multi-part queries, which may still require human intervention.
* **User Acceptance:** Customers may prefer human interaction for certain types of queries, especially if they perceive chatbots as impersonal or lacking empathy.
* **Security and Privacy:** Chatbots need to handle customer data securely and comply with privacy regulations, as they often collect sensitive information

**CHAPTER-3**

**PROPOSED METHODOLOGY**

**3.1.1 Software Requirements:**

The software stack for building, training, and deploying a customer support chatbot includes a variety of tools, frameworks, and libraries for data processing, machine learning, and deployment.

**3.1.2. Operating System:**

* **Linux (Ubuntu):** Many ML and AI frameworks perform better on Linux, and it is commonly used in production environments.
* **Windows and macOS:** Suitable for development; many tools support cross-platform development, though Linux compatibility is often preferred for deployment.

**3.1.3 Programming Languages:**

* + **JavaScript:** Often used for front-end integration and building chatbot interfaces on web applications.
  + **R:** Although less common for chatbot development, R is sometimes used for data analysis and preprocessing.
  + **Python:** The primary language for machine learning and NLP tasks due to its extensive libraries and frameworks.

**3.1.4. Machine Learning and NLP Frameworks:**

* + **TensorFlow:** Widely used for building and training machine learning models, especially deep learning models. TensorFlow Hub offers pre-trained models, and TensorFlow Serving can deploy models in production.
  + **PyTorch:** Another popular deep learning framework, known for its flexibility and ease of use. PyTorch is particularly useful for research and prototyping.
  + **Hugging Face Transformers:** This library provides access to state-of-the-art transformer models like BERT, GPT-2, and T5. It includes pre-trained models and tools for fine-tuning on specific tasks.
  + **spaCy:** Useful for natural language processing tasks, such as tokenization, named entity recognition, and intent classification.
  + **NLTK and TextBlob:** Useful for text preprocessing tasks like tokenization, lemmatization, and sentiment analysis.

**3.1.5. Data Processing and Management Tools:**

* + - **Pandas and NumPy:** Essential Python libraries for data manipulation, cleaning, and preprocessing.
    - **SQL:** Many projects use SQL databases to store and retrieve structured data related to chatbot interactions, such as customer queries and responses.
    - **MongoDB or Firebase:** NoSQL databases are often used for unstructured data, enabling scalable and flexible storage solutions for chatbot data

**3.1.6. Model Training and Experimentation Tools:**

* **Jupyter Notebook:** An interactive environment for developing and testing machine learning models, often used during the experimentation phase.
* **Google Colab:** Provides free access to GPUs and TPUs for training deep learning models and testing code.
* **Weights & Biases or TensorBoard:** Tools for tracking and visualizing model training performance, useful for hyperparameter tuning and monitoring experiments.

**3.1.7. Chatbot Development Platforms and APIs:**

* **Dialogflow (Google):** Provides pre-built NLP tools for intent recognition and entity extraction. Also includes integrations with various messaging platforms.
* **Microsoft Bot Framework:** An SDK and set of tools for building and deploying chatbots across multiple channels, including web, mobile, and social media.
* **Rasa:** An open-source framework for building custom, conversational AI chatbots. Rasa offers both NLU and dialogue management capabilities, ideal for more complex use cases.
* **Twilio or Facebook Messenger API:** APIs for integrating the chatbot with SMS, social media, and other messaging platforms.

**3.1.8.Data Collection and Preparation**

* Data Gathering: Collect a dataset of customer queries and responses. This can be from existing customer support logs, FAQs, or similar sources. You may also include labeled data for intent classification and sample dialogues for training.
* Data Cleaning: Remove any irrelevant, duplicate, or outdated data. This process may include text normalization, lowercasing, removing stop words, and handling misspellings.
* Data Labeling: Label the dataset for intent classification, entity recognition, and any other specific tasks the chatbot needs to perform.
* Data Splitting: Split the data into training, validation, and test sets (e.g., 70% training, 15% validation, and 15% testing) to ensure the model is trained and evaluated properly.

**CHAPTER-4**

**OBJECTIVES**

**4.1.1 .The primary objectives**

1) Explore the evolution of customer support chatbots

2) Investigate machine learning techniques in chatbot development

3) Identify challenges and ethical considerations

**Objective: Improve Natural Language Understanding (NLU) Capabilities**

* **Description:** Develop a customer support chatbot that can better understand and interpret a wider range of customer queries, including complex or multi-part questions, by leveraging advanced NLP models (e.g., transformers like BERT or GPT). This objective aims to address the common limitation of chatbots struggling with nuanced language, sarcasm, or complex inquiries.
* **Research Gap:** Many existing chatbots lack the ability to understand intricate or ambiguous language structures, which can lead to misinterpretation and customer frustration.

**Objective: Enhance Context Retention and Multi-turn Conversation Capabilities**

* **Description:** Create a chatbot that can maintain context across multiple turns in a conversation, allowing it to handle follow-up questions and provide more coherent, personalized responses. This will improve the chatbot’s ability to interact in a more human-like manner, enhancing the overall customer experience.
* **Research Gap:** Many current chatbot solutions struggle with maintaining context over extended interactions, leading to disjointed conversations that can confuse users and decrease user satisfaction.

**Objective: Integrate Sentiment Analysis for Better Customer Engagement**

* **Description:** Integrate sentiment analysis into the chatbot’s architecture to detect the emotional tone of customer messages and adjust responses accordingly. This can help the chatbot identify frustrated or unhappy customers and escalate those interactions to human agents when needed, ensuring more empathetic and timely responses.
* **Research Gap:** While sentiment analysis is increasingly common in other applications, many customer support chatbots lack this feature. This gap can result in missed opportunities to de-escalate situations and provide more empathetic service.

**Objective: Improve Transfer Learning and Domain Adaptation Techniques for Industry-Specific Chatbots**

* **Description:** Investigate and implement transfer learning techniques to make chatbots more adaptable to specific industries (e.g., healthcare, finance, retail) with minimal fine-tuning. This will allow for the rapid deployment of chatbots that are tailored to the specific needs and vocabulary of different domains.
* **Research Gap:** Many existing customer support chatbots are either too generic or require extensive retraining to be effective in specific industries, limiting their scalability and effectiveness in niche sectors.

**4.1.2.Project Objectives and Requirements:**

* **Objective:** Identify the primary goals of the chatbot, such as answering FAQs, handling customer inquiries, or escalating complex queries to human agents.
* Scope: Decide on the scope of the chatbot, including the types of queries it should handle, expected response accuracy, and user experience requirements.
* Target Audience: Understand the target users of the chatbot, which can inform tone, language style, and the kinds of queries it needs to handle.
* Platform: Decide where the chatbot will be deployed (e.g., website, mobile app, social media, messaging platforms).

**4.1.3.Design the Chatbot Architecture:**

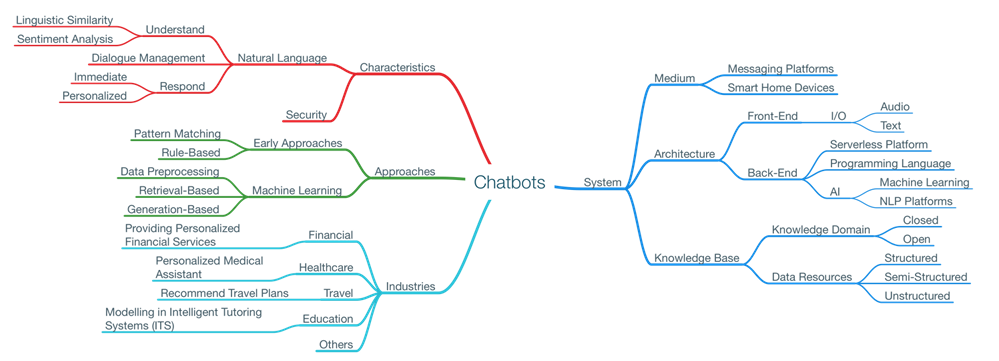
* NLU and NLP Components:
  + Use NLP techniques to process and understand user queries. Select or develop components for tasks like intent classification, entity recognition, and sentiment analysis.
  + Decide whether to use pre-trained models (e.g., BERT, GPT-3) or custom-trained models for NLP tasks. Libraries like Hugging Face Transformers or spaCy are commonly used here.
* Dialogue Management:
  + Choose a framework or algorithm for dialogue management. Options include rule-based systems, reinforcement learning, or pre-built frameworks (like Rasa) that offer dialogue management features.
  + Define the conversation flow, including decision points, fallback mechanisms, and handling of multiple intents or follow-up questions.
* Response Generation:
  + Design the response generation component. Simple chatbots use predefined responses based on intents, while advanced chatbots use models (e.g., GPT-3) to generate dynamic, contextually relevant replies.
  + Ensure the responses align with the chatbot’s tone and style, and prepare fallback responses for cases where the bot does not understand the query

**4.1.4. Model Training and Testing**

* Model Selection: Based on the complexity of the task, select the models for specific NLP tasks:
  + Use deep learning models like BERT or LSTM for intent classification and entity recognition.
  + For complex, open-ended responses, you may use a transformer model like GPT or T5 for text generation.
* Training: Train the models on the prepared dataset. Use appropriate machine learning libraries like TensorFlow or PyTorch and experiment with hyperparameters (e.g., learning rate, batch size).
* Validation and Hyperparameter Tuning: Use the validation set to tune hyperparameters for better performance. Track model performance metrics (accuracy, F1-score) to evaluate effectiveness.
* Testing: Evaluate the final model on the test set to ensure it meets performance requirements. Test with real user queries or historical data to assess accuracy and response quality.

**CHAPTER-5**

**System Components and Block Diagram:**

****

**Figure 1.1**

**5.1.1 Components:**

**Research Design** :The research adopts a mixed-methods approach, combining both qualitative and quantitative methods to comprehensively evaluate the efficacy of machine learning-driven chatbots and virtual assistants in customer support. This approach allows for a thorough analysis of both numerical data and subjective experiences, providing a holistic understanding of the performance and impact of these technologies .

**Data Collection**: Sources of data included surveys, interviews, and historical data. Surveys were conducted to gather quantitative data from customers who have interacted with chatbots and virtual assistants. These surveys included questions on user satisfaction, response time, and perceived accuracy of the responses . In-depth interviews were conducted with customer support managers and agents to collect qualitative data. These interviews focused on the experiences, challenges, and benefits of integrating ML-driven chatbots into their support systems (Jain et al., 2020). Historical customer support data, including records of previous interactions handled by both human agents and chatbots, were collected from participating companies. This data provided a basis for comparing the performance of traditional and ML driven support systems Tools and techniques for data collection included online survey platforms like SurveyMonkey and Google Forms to distribute and collect survey responses (Luo et al., 2019), interview recording devices such as digital recorders and transcription software to accurately capture and document interviews (Jain et al., 2020), and data management systems like SQL and CRM software to extract customer support records from company databases.

**Machine Learning Models and Algorithms** :Specific ML models used included NLP techniques, deep learning models, and reinforcement learning algorithms. NLP techniques were employed to enable chatbots to understand and generate human language, with models such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) used for their advanced language processing capabilities (Devlin et al., 2018; Radford et al., 2019). Deep learning models, including Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, were used to handle sequential data and improve the chatbots' ability to understand context and maintain conversation coherence (Hochreiter & Schmidhuber, 1997). Chatbots were further trained using reinforcement learning algorithms to optimize their responses based on user feedback and interactions (Sutton & Barto, 2018).

**Evaluation Metrics** : Criteria for assessing chatbot and virtual assistant performance included response time, accuracy, user satisfaction, and resolution rate. Response time was measured as the average time taken by the chatbot to respond to customer queries (Luo et al., 2019). Accuracy was determined by the percentage of correct responses provided by the chatbot, compared to a set of predefined correct responses (Shum et al., 2018). User satisfaction was measured through survey responses, rating the overall satisfaction of users with the chatbot interactions (Jain et al., 2020). The resolution rate was the percentage of customer queries successfully resolved by the chatbot without requiring human intervention.

**Data Analysis Methods :** Statistical and analytical techniques used included descriptive statistics, inferential statistics, sentiment analysis, and regression analysis. Descriptive statistics were used to summarize and describe the main features of the collected data, including mean, median, and standard deviation of response times and satisfaction ratings (Field, 2013). Inferential statistics techniques such as t-tests and ANOVA were used to compare the performance metrics of chatbots and human agents, determining if observed differences were statistically significant (Field, 2013). Sentiment analysis was applied to qualitative data from interviews to analyze the sentiment and opinions of customer support managers and agents regarding the use of chatbots (Liu, 2012). Regression analysis was employed to identify factors that significantly impact user satisfaction and resolution rates, helping to understand the relationships between various performance metrics and customer outcomes.

**5.1.2 Chatbot Characteristics**

* Understanding: Chatbots utilize natural language processing (NLP) techniques, such as linguistic similarity and sentiment analysis, to understand user inputs and adapt responses accordingly.
* Response Features: Key features include immediate and personalized replies, facilitated by dialogue management systems.
* Security: Ensuring robust data protection measures, such as encryption and secure data handling, is critical for user trust and system integrity.

**Approaches to Chatbot Development**

* Early Approaches:
  + Pattern Matching: Rule-based systems that rely on predefined patterns to generate responses.
  + Data Preprocessing: Preparing data for efficient processing in chatbot algorithms.

**Machine Learning-Based:**

* + Retrieval-Based Models: Selecting appropriate responses from a predefined database.
  + Generation-Based Models: Dynamically generating responses using AI algorithms.

**Applications Across Industries**

* Healthcare: Chatbots assist with personalized medical recommendations and patient support.
* Financial Services: Offer services like financial planning and customer account assistance.
* Travel: Provide personalized travel plans and itinerary suggestions.
* Education: Support intelligent tutoring systems (ITS) by modeling learner behavior and needs.

**System Architecture**

* Medium: Chatbots operate across platforms, including messaging apps and smart home devices.
* Components:
  + Front-End: Manages user interaction through audio or text interfaces.
  + Back-End: Utilizes serverless platforms, programming languages, and AI techniques for processing and logic.
* AI Integration: Employs natural language processing (NLP) and machine learning for intelligent interaction.

**Knowledge Base**

* Types of Knowledge Domains:
  + Closed Systems: Limited to predefined scopes of knowledge.
  + Open Systems: Designed for broader and evolving data sources.
* Data Resources:
  + Structured (e.g., databases).
  + Semi-structured (e.g., XML, JSON).
  + Unstructured (e.g., plain text, images).

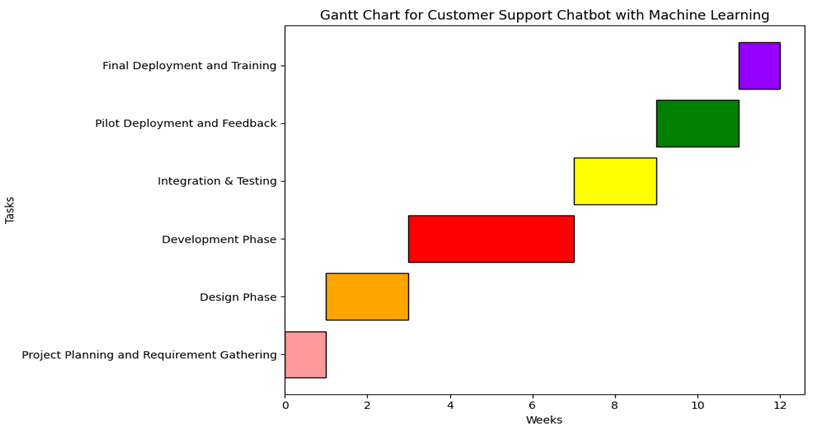
**5.1.3 Future Potential**

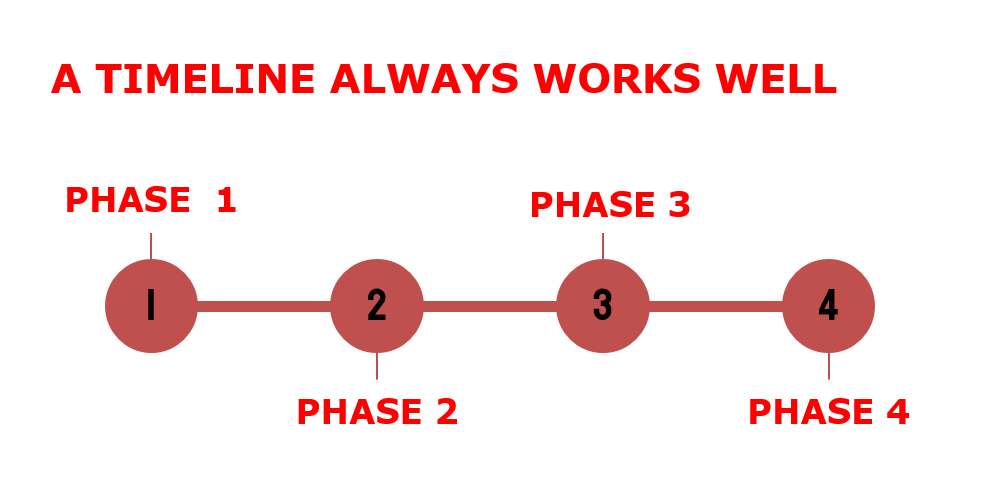
* Chatbots can expand to include more sophisticated machine learning algorithms, multimodal interfaces (combining audio, video, and text), and enhanced capabilities in real-time decision-making across diverse domains.

**CHAPTER-6**

**TIMELINE FOR EXECUTION OF PROJECT**

**(GANTT CHART)**

****



**CHAPTER-7**

**OUTCOMES**

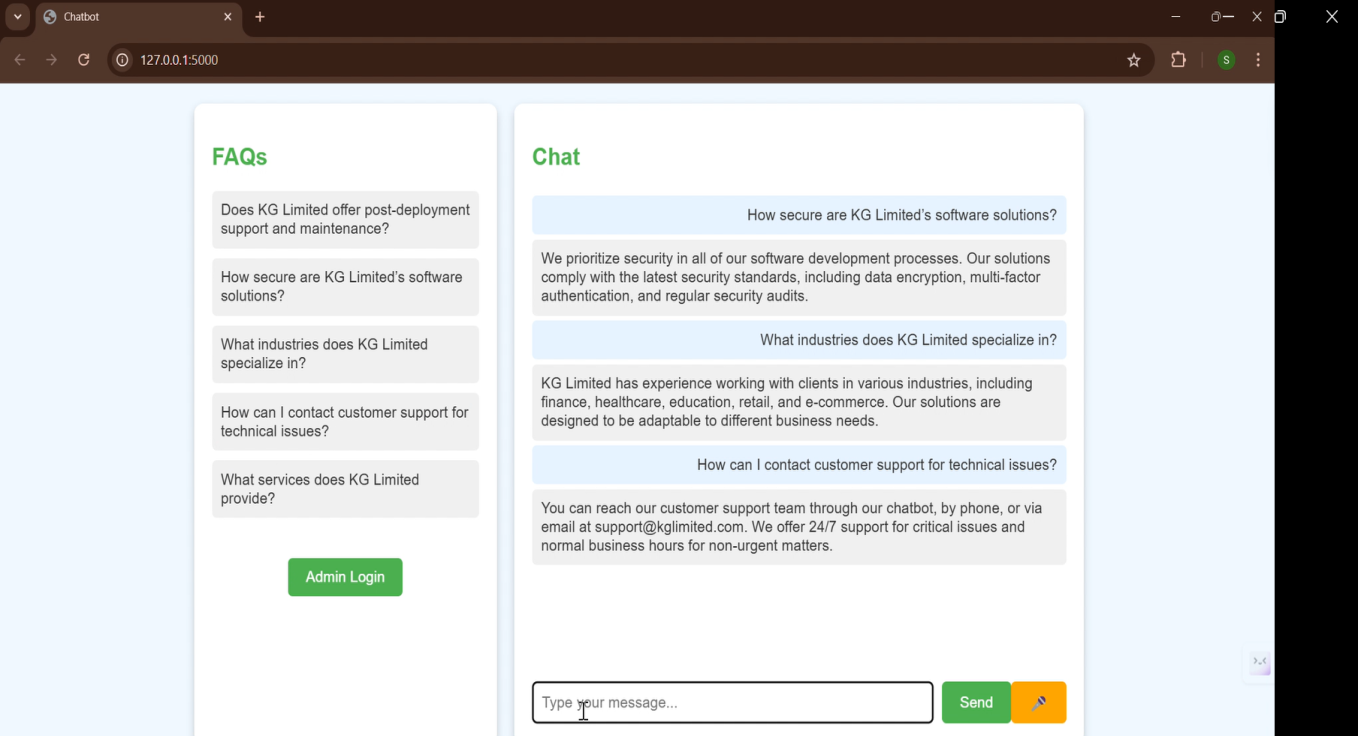
The project reveals several significant trends and insights. First, ML-driven chatbots substantially reduce response times, leading to quicker resolutions and higher customer satisfaction. Second, the accuracy and reliability of chatbots are higher than those of human agents, likely due to the advanced language processing and learning capabilities of the ML models used. Third, the overall efficiency of customer support operations improves with the integration of chatbots, as they handle a large volume of queries simultaneously without fatigue or errors. Lastly, customers express a clear preference for the convenience and effectiveness of chatbot interactions over traditional support methods. These findings suggest that integrating ML-driven chatbots into customer support systems can lead to enhanced performance, greater customer satisfaction, and more efficient operations.

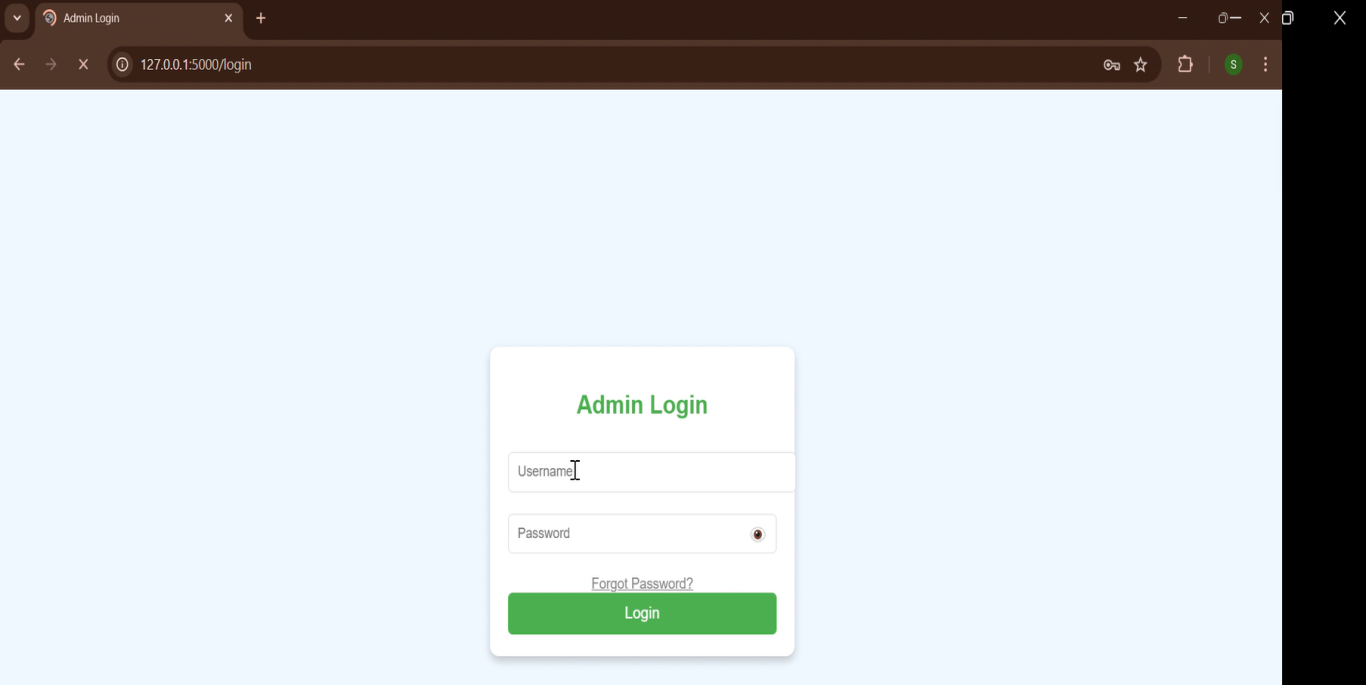
* Improved Customer Service Efficiency
* Enhanced Customer Experience and Satisfaction
* Scalability and Flexibility
* Data-Driven Insights

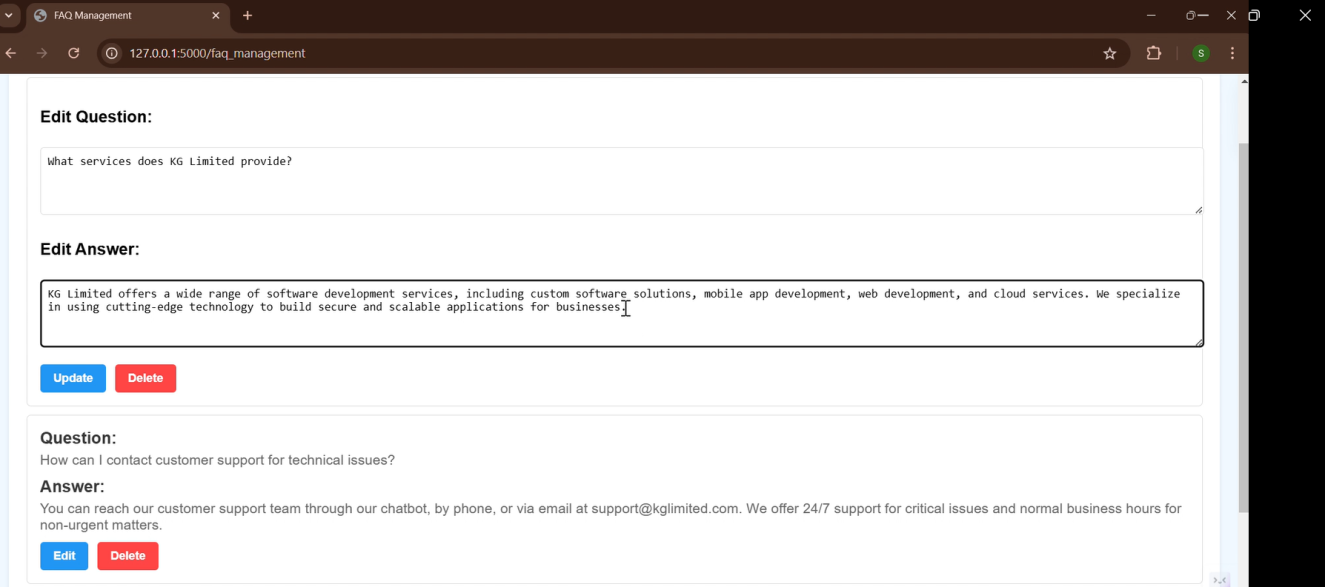
**CHAPTER-8**

**RESULTS AND DISCUSSIONS**

**Fig 1.2**

****

****

****

**User Interface Design:**

* + The chatbot's interface includes a well-organized FAQ panel on the left, providing quick access to common questions. This design ensures ease of use and reduces the need for typing.
  + The chat window on the right offers real-time responses to user queries, enhancing interactivity and user engagement.

**Functionality:**

* + The FAQ section covers essential topics such as security measures, industry specialization, and customer support, demonstrating the chatbot's focus on addressing user concerns comprehensively.
  + The chatbot provides direct answers with clarity, such as details about 24/7 customer support availability and the security protocols followed by KG Limited.

**Security and Trust:**

* + Highlighted features like data encryption, multi-factor authentication, and regular audits emphasize the importance of secure software solutions, which could increase user trust in the system.

**Applicability Across Industries:**

* + The chatbot mentions KG Limited's expertise across diverse industries, including healthcare, education, and e-commerce. This versatility indicates that the chatbot can cater to a wide range of customer needs.

**Suggestions for Improvement:**

* + While the chatbot interface is user-friendly, it could benefit from additional features like language support or a feedback mechanism for user input.
  + Adding more visual cues, such as icons for FAQ categories, might improve the user experience further.

**CHAPTER-9**

**CONCLUSION**

In conclusion, implementing a customer support chatbot powered by machine learning offers significant advantages for both businesses and customers. By leveraging natural language processing and advanced AI models, such chatbots can understand and respond to customer inquiries in real time, providing quick, accurate, and consistent service. This capability not only improves customer satisfaction but also enhances operational efficiency by handling a large volume of repetitive inquiries autonomously.Additionally, a well-designed chatbot can be available 24/7, providing support outside traditional business hours, which reduces wait times and ensures customers always have access to assistance. With features like intent recognition, entity extraction, and sentiment analysis, these chatbots can provide more personalized and empathetic interactions, seamlessly escalating more complex issues to human agents when necessary.This project has explored the efficacy of machine learning-driven chatbots and virtual assistants in automating customer support, revealing significant advancements in operational efficiency and customer satisfaction metrics. Key findings highlight the ability of AI-powered systems to handle diverse customer queries effectively, outperforming traditional methods in terms of response time and accuracy. The deployment of machine learning technologies has shown transformative potential in customer service operations, demonstrating scalability and adaptability across various industries. The integration of AI-driven chatbots and virtual assistants not only streamlines support processes but also reduces operational costs significantly, marking a paradigm shift in service delivery. Implications for Businesses and Technology Providers For businesses, integrating machine learning-driven solutions offers practical benefits such as cost savings, enhanced service delivery, and improved customer retention rates. Technology providers can capitalize on these findings by developing advanced AI algorithms tailored to specific customer support needs, thereby catering to a growing demand for efficient and scalable support solutions. businesses are encouraged to invest in AI technologies that enhance the capabilities of customer support operations. Implementing robust machine learning models trained on extensive datasets enables organizations to provide personalized customer interactions and real-time problem resolution. Moreover, continuous updates and refinement of AI systems are essential to maintaining relevance and addressing evolving customer needs.Looking ahead, the trajectory of customer support automation is poised for further advancements in AI capabilities and integration with emerging technologies like augmented reality and natural language processing. Future research should continue exploring avenues for enhancing AI-driven interactions, ensuring ethical considerations and customer-centricity remain central to technological advancements in customer support.

**11.1.REFERENCES**

**[1]** "Customer Support Chatbot Using Machine Learning," *ResearchGate*, [Online].

**[2]** IRJET, "A Comprehensive Study on Decision-Tree-Based Customer Support Chatbot Systems," *International Research Journal of Engineering and Technology*, vol. 10, no. 12, 2023. [Online]. Available:<https://www.irjet.net/archives/V10/i12/IRJET-V10I12101.pdf>.

**[3]** IIIS, "Enhancing Multi-Language Chatbots for Global Customer Support," 2024.

**[4]** SSRN, "Continuous Learning and Reinforcement Models in Customer Service Chatbots," 2023. [Online]. Available:<https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4649539>. .

**[5]** IJMERR, "Clustering Algorithms for Optimizing Chatbot FAQ Modules," *International Journal of Mechanical Engineering and Robotics Research*, vol. 9, no. 3, 2020. [Online]. Available:<https://www.ijmerr.com/uploadfile/2020/0312/20200312023706525.pdf>.

**[6]** ResearchBerg, "Data Preprocessing for NLP in Customer Support Chatbots," *Engineering Quarterly Management and Economics Review*, vol. 4, no. 2, 2021. [Online]. Available:<https://www.researchberg.com/index.php/eqme/article/view/46>.

**[7]** DLABI, "Integration of Speech-to-Text in Chatbots Using Google APIs," *DLABI Journal of Artificial Intelligence and Automation*, vol. 7, no. 1, 2021. [Online]. Available:<https://dlabi.org/index.php/journal/article/view/106>.

**[8]** IRE Journals, "Performance Evaluation Metrics for Chatbots Post-Deployment," *International Research Explorer Journals*, vol. 15, no. 8, 2021. [Online]. Available:<https://www.irejournals.com/formatedpaper/17048601.pdf>.

**11.1.1.** APPENDIX**-A**

**PSUEDOCODE**

**app.py**

from flask import Flask, request, jsonify, send\_from\_directory, redirect, url\_for, render\_template, session

from flask\_sqlalchemy import SQLAlchemy

import openai

import os

app = Flask(\_\_name\_\_, static\_folder='static')

# Configure the database URI

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'mysql+pymysql://root:root%4012@localhost/chatbot\_db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

app.secret\_key = 'your\_secret\_key' # Required for session handling

db = SQLAlchemy(app)

# Set up OpenAI API key securely

openai.api\_key = os.getenv("OPENAI\_API\_KEY") # Set your key as an environment variable for security

# Define admin credentials

ADMIN\_USERNAME = "Sunil@ele"

ADMIN\_PASSWORD = "Sunil12"

class Query(db.Model):

id = db.Column(db.Integer, primary\_key=True)

question = db.Column(db.String(500), nullable=False)

answer = db.Column(db.String(1000), nullable=False)

# Serve the main page

@app.route('/')

def index():

return send\_from\_directory(app.static\_folder, 'index.html')

# Chat endpoint for handling user messages

@app.route('/api/chat', methods=['POST'])

def chat():

data = request.json

user\_message = data.get('message', '')

if not user\_message:

return jsonify({"response": "Message cannot be empty"}), 400

# Check if a similar question exists in the database

similar\_query = Query.query.filter(Query.question.like(f"%{user\_message}%")).first()

if similar\_query:

return jsonify({"response": similar\_query.answer, "source": "database"})

else:

ai\_response = get\_ai\_response(user\_message)

# Store the new query in the database

new\_query = Query(question=user\_message, answer=ai\_response)

try:

db.session.add(new\_query)

db.session.commit()

except Exception as e:

db.session.rollback()

return jsonify({"error": "Failed to save to database", "details": str(e)}), 500

return jsonify({"response": ai\_response, "source": "AI"})

def get\_ai\_response(user\_message):

"""

Interact with OpenAI to generate a response for a user message.

"""

try:

response = openai.Completion.create(

engine="text-davinci-003",

prompt=f"Answer this question briefly: {user\_message}",

max\_tokens=50

)

return response.choices[0].text.strip() or "I'm sorry, I don't have an answer for that."

except Exception as e:

return "There was an error generating a response. Please try again later."

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

# Validate credentials

if username == ADMIN\_USERNAME and password == ADMIN\_PASSWORD:

session['logged\_in'] = True # Store login state in the session

return redirect(url\_for('faq\_management'))

else:

return render\_template('login.html', error="Invalid credentials. Please try again.")

return render\_template('login.html')

@app.route('/faq\_management')

def faq\_management():

if not session.get('logged\_in'):

return redirect(url\_for('login')) # Redirect to login if not authenticated

return render\_template('faq\_management.html') # Render FAQ management page

@app.route('/logout')

def logout():

session.pop('logged\_in', None) # Clear the login state from the session

return redirect(url\_for('index'))

@app.route('/api/queries', methods=['GET'])

def get\_queries():

queries = Query.query.all()

return jsonify([{"id": q.id, "question": q.question, "answer": q.answer} for q in queries])

@app.route('/api/faqs', methods=['GET'])

def get\_faqs():

try:

faqs = Query.query.order\_by(Query.id.desc()).limit(7).all()

return jsonify([{"question": faq.question, "answer": faq.answer} for faq in faqs])

except Exception as e:

return jsonify({"status": "error", "message": "Failed to retrieve FAQs", "details": str(e)}), 500

@app.route('/api/update\_faq', methods=['PUT'])

def update\_faq():

if not session.get('logged\_in'):

return jsonify({"error": "Unauthorized"}), 401

try:

data = request.get\_json()

if not data:

return jsonify({"error": "No data provided"}), 400

faq\_id = data.get('id')

question = data.get('question', '').strip()

answer = data.get('answer', '').strip()

# Validate input

if not all([faq\_id, question, answer]):

return jsonify({"error": "Question and answer cannot be empty"}), 400

# Get the FAQ entry

faq = Query.query.get(faq\_id)

if not faq:

return jsonify({"error": "FAQ not found"}), 404

# Update the FAQ

faq.question = question

faq.answer = answer

# Commit the changes

db.session.commit()

return jsonify({

"status": "success",

"message": "FAQ updated successfully",

"data": {

"id": faq.id,

"question": faq.question,

"answer": faq.answer

}

})

except Exception as e:

db.session.rollback()

return jsonify({

"error": "Failed to update FAQ",

"details": str(e)

}), 500

@app.route('/api/delete\_faq/<int:faq\_id>', methods=['DELETE'])

def delete\_faq(faq\_id):

if not session.get('logged\_in'):

return jsonify({"error": "Unauthorized"}), 401

try:

faq = Query.query.get(faq\_id)

if not faq:

return jsonify({"error": "FAQ not found"}), 404

db.session.delete(faq)

db.session.commit()

return jsonify({

"status": "success",

"message": "FAQ deleted successfully",

"data": {"id": faq\_id}

})

except Exception as e:

db.session.rollback()

return jsonify({

"error": "Failed to delete FAQ",

"details": str(e)

}), 500

@app.route('/api/faq/<int:faq\_id>', methods=['GET'])

def get\_faq(faq\_id):

if not session.get('logged\_in'):

return jsonify({"error": "Unauthorized"}), 401

try:

faq = Query.query.get(faq\_id)

if not faq:

return jsonify({"error": "FAQ not found"}), 404

return jsonify({

"status": "success",

"data": {

"id": faq.id,

"question": faq.question,

"answer": faq.answer

}

})

except Exception as e:

return jsonify({

"error": "Failed to retrieve FAQ",

"details": str(e)

}), 500

# Catch-all route for undefined paths

@app.route('/<path:path>')

def catch\_all(path):

return jsonify({"error": "Endpoint not found", "path": path}), 404

if \_\_name\_\_ == '\_\_main\_\_':

with app.app\_context():

db.create\_all()

app.run(debug=True)

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Chatbot</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f0f8ff;

color: #333;

display: flex;

justify-content: center;

}

.container {

display: flex;

width: 80%;

max-width: 1000px;

margin-top: 20px;

gap: 20px;

}

.sidebar {

flex: 1;

background-color: #ffffff;

border-radius: 10px;

padding: 20px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

max-height: 600px;

overflow-y: auto;

}

.chat-container {

flex: 2;

background-color: #ffffff;

border-radius: 10px;

padding: 20px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

height: 600px;

display: flex;

flex-direction: column;

}

.faq-container h2, .chat-container h2 {

color: #4CAF50;

}

.faq-container .faq-question {

padding: 10px;

cursor: pointer;

margin-bottom: 10px;

border-radius: 5px;

background-color: #f0f0f0;

transition: background-color 0.3s;

}

.faq-container .faq-question:hover {

background-color: #e0e0e0;

}

.chat-messages {

flex-grow: 1;

overflow-y: auto;

margin-bottom: 10px;

}

.message {

padding: 10px;

border-radius: 5px;

margin: 5px 0;

}

.user {

background-color: #e6f3ff;

text-align: right;

}

.bot {

background-color: #f0f0f0;

}

.input-container {

display: flex;

}

#user-input {

flex-grow: 1;

padding: 10px;

font-size: 16px;

border: 1px solid #ddd;

border-radius: 5px;

margin-right: 10px;

}

#send-button, #voice-button {

padding: 10px 20px;

font-size: 16px;

border: none;

cursor: pointer;

color: white;

border-radius: 5px;

}

#send-button {

background-color: #4CAF50;

}

#voice-button {

background-color: #FFA500;

position: relative;

}

/\* Mic animation \*/

#voice-button.listening::after {

content: '';

position: absolute;

width: 20px;

height: 20px;

top: -5px;

right: -5px;

background: #FF4500;

border-radius: 50%;

animation: pulse 1s infinite;

}

@keyframes pulse {

0% {

transform: scale(1);

opacity: 1;

}

50% {

transform: scale(1.5);

opacity: 0.7;

}

100% {

transform: scale(2);

opacity: 0;

}

}

/\* Scrollbars \*/

.sidebar::-webkit-scrollbar, .chat-messages::-webkit-scrollbar {

width: 8px;

}

.sidebar::-webkit-scrollbar-thumb, .chat-messages::-webkit-scrollbar-thumb {

background-color: #4CAF50;

border-radius: 10px;

}

.login-footer {

text-align: center;

margin-top: 20px;

padding: 20px;

}

#login-button {

background-color: #4CAF50;

color: #fff;

border: none;

padding: 10px 20px;

font-size: 16px;

border-radius: 5px;

cursor: pointer;

}

#login-button:hover {

background-color: #45a049;

}

.language-selector {

position: absolute;

top: 10px;

right: 10px;

}

.language-selector select {

padding: 5px;

font-size: 14px;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="language-selector">

<select id="language-dropdown">

<option value="en">English</option>

<option value="hi">हिन्दी</option>

<option value="kn">ಕನ್ನಡ</option>

<option value="te">తెలుగు</option>

</select>

</div>

<div class="container">

<div class="sidebar faq-container">

<h2 id="faq-title">FAQs</h2>

<div id="faq-list"></div>

<div class="login-footer">

<button id="login-button" onclick="location.href='/login'">Admin Login</button>

</div>

</div>

<div class="chat-container">

<h2 id="chat-title">Chat</h2>

<div id="chat-messages" class="chat-messages"></div>

<div class="input-container">

<input type="text" id="user-input" placeholder="Type your message...">

<button id="send-button">Send</button>

<button id="voice-button">🎤</button>

</div>

</div>

</div>

<script>

const OPENAI\_API\_KEY = "-------------------------------------------------------------------------------------------"; // Replace with your OpenAI API key

// Translations for static UI elements

const translations = {

en: { faqTitle: "FAQs", chatTitle: "Chat", loginButton: "Admin Login", sendButton: "Send", placeholder: "Type your message..." },

hi: { faqTitle: "प्रश्नोत्तर", chatTitle: "चैट", loginButton: "प्रशासकलॉगिन", sendButton: "भेजें", placeholder: "अपनासंदेशलिखें..." },

kn: { faqTitle: "ಪ್ರಶ್ನೋತ್ತರಗಳು", chatTitle: "ಚಾಟ್", loginButton: "ನಿರ್ವಹಣೆಲಾಗಿನ್", sendButton: "ಕಳುಹಿಸಿ", placeholder: "ನಿಮ್ಮಸಂದೇಶವನ್ನುಬರೆಯಿರಿ..." },

te: { faqTitle: "ప్రశ్నలు", chatTitle: "చాట్", loginButton: "అడ్మిన్లాగిన్", sendButton: "పంపండి", placeholder: "మీసందేశాన్నిటైప్చేయండి..." }

};

constfaqList = document.getElementById('faq-list');

constchatMessages = document.getElementById('chat-messages');

constuserInput = document.getElementById('user-input');

constsendButton = document.getElementById('send-button');

constlanguageDropdown = document.getElementById('language-dropdown');

constvoiceButton = document.getElementById('voice-button'); // Make sure this matches your HTML

consttranslationCache = {}; // Cache to store translations

let selectedLanguage = 'en'; // Default language

// Function to send batch texts to OpenAI API for translation

async function translateBatch(texts, targetLanguage) {

const response = await fetch('https://api.openai.com/v1/chat/completions', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${OPENAI\_API\_KEY}`,

},

body: JSON.stringify({

model: "gpt-3.5-turbo",

messages: [

{ role: "system", content: `Translate the following texts to ${targetLanguage} without changing their order:` },

{ role: "user", content: texts.join("\n") },

],

}),

});

if (!response.ok) throw new Error("Translation API call failed");

const data = await response.json();

return data.choices[0].message.content.split("\n").map(text =>text.trim());

}

// Function to translate individual text

async function translateText(text, targetLanguage) {

if (targetLanguage === 'en') return text; // No translation needed for English

if (translationCache[`${text}\_${targetLanguage}`]) {

return translationCache[`${text}\_${targetLanguage}`];

}

consttranslatedTexts = await translateBatch([text], targetLanguage);

translationCache[`${text}\_${targetLanguage}`] = translatedTexts[0]; // Cache result

return translatedTexts[0];

}

// Function to load FAQs from the server

async function loadFAQs() {

try {

const response = await fetch('/api/faqs');

if (!response.ok) throw new Error("Error fetching FAQs");

constfaqs = await response.json();

faqList.innerHTML = ''; // Clear previous FAQ list

faqs.forEach(faq => {

constquestionElement = document.createElement('div');

questionElement.classList.add('faq-question');

questionElement.textContent = faq.question; // Display original text initially

questionElement.dataset.originalText = faq.question; // Store original text for translation

questionElement.addEventListener('click', () => {

userInput.value = faq.question;

sendMessage();

});

faqList.appendChild(questionElement);

});

} catch (error) {

console.error('Error loading FAQs:', error);

}

}

// Function to translate FAQ section

async function translateFAQs(language) {

try {

constfaqElements = faqList.querySelectorAll('.faq-question');

constoriginalTexts = Array.from(faqElements).map(faq =>faq.dataset.originalText);

const translations = await translateBatch(originalTexts, language);

faqElements.forEach((faqElement, index) => {

faqElement.textContent = translations[index];

});

} catch (error) {

console.error('Error translating FAQs:', error);

}

}

// Function to update the UI language

async function updateLanguage(language) {

selectedLanguage = language;

const translation = translations[language];

document.getElementById('faq-title').textContent = translation.faqTitle;

document.getElementById('chat-title').textContent = translation.chatTitle;

document.getElementById('login-button').textContent = translation.loginButton;

document.getElementById('send-button').textContent = translation.sendButton;

userInput.placeholder = translation.placeholder;

await translateFAQs(language);

}

// Function to append a chat message

function appendMessage(text, sender) {

constmessageElement = document.createElement('div');

messageElement.classList.add('message', sender);

messageElement.textContent = text;

chatMessages.appendChild(messageElement);

chatMessages.scrollTop = chatMessages.scrollHeight;

}

// Function to send a chat message

async function sendMessage() {

const message = userInput.value.trim();

if (message === '') return;

appendMessage(message, 'user');

userInput.value = '';

try {

consttranslatedMessage = await translateText(message, 'en'); // Translate to English before sending to backend

const response = await fetch('/api/chat', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ message: translatedMessage }),

});

if (!response.ok) throw new Error("Error communicating with bot");

const data = await response.json();

consttranslatedResponse = await translateText(data.response, selectedLanguage); // Translate bot response to selected language

appendMessage(translatedResponse, 'bot');

} catch (error) {

console.error('Error:', error);

}

}

// Function to handle speech recognition

function initializeSpeechRecognition() {

// Check browser support for speech recognition

if (!('SpeechRecognition' in window) && !('webkitSpeechRecognition' in window)) {

alert('Speech recognition is not supported in this browser.');

voiceButton.style.display = 'none';

return;

}

constSpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;

const recognition = new SpeechRecognition();

// Configure recognition

recognition.continuous = false;

recognition.interimResults = false;

recognition.lang = getLanguageCode(selectedLanguage);

voiceButton.addEventListener('click', () => {

try {

recognition.start();

voiceButton.classList.add('listening');

} catch (error) {

console.error('Speech recognition error:', error);

alert('Could not start speech recognition. Please try again.');

}

});

recognition.onresult = async (event) => {

constvoiceInput = event.results[0][0].transcript;

try {

consttranslatedInput = await translateText(voiceInput, selectedLanguage);

userInput.value = translatedInput;

sendMessage();

} catch (error) {

console.error('Translation error:', error);

alert('Error processing voice input. Please try again.');

}

voiceButton.classList.remove('listening');

};

recognition.onerror = (event) => {

console.error('Speech recognition error:', event.error);

voiceButton.classList.remove('listening');

switch(event.error) {

case 'no-speech':

alert('No speech was detected. Please try again.');

break;

case 'audio-capture':

alert('Audio capture failed. Check your microphone settings.');

break;

case 'not-allowed':

alert('Speech recognition is not allowed. Please check browser permissions.');

break;

default:

alert('Speech recognition failed. Please try again.');

}

};

recognition.onend = () => {

voiceButton.classList.remove('listening');

};

}

// Helper function to get language code for speech recognition

function getLanguageCode(language) {

constlanguageCodes = {

'en': 'en-US',

'hi': 'hi-IN',

'kn': 'kn-IN',

'te': 'te-IN'

};

return languageCodes[language] || 'en-US';

}

// Event listeners

sendButton.addEventListener('click', sendMessage);

userInput.addEventListener('keypress', (e) => {

if (e.key === 'Enter') sendMessage();

});

languageDropdown.addEventListener('change', (e) => {

updateLanguage(e.target.value);

});

// Initialize the app

document.addEventListener('DOMContentLoaded', async () => {

await loadFAQs(); // Load FAQ data from the server

await updateLanguage('en'); // Set default language

initializeSpeechRecognition(); // Initialize speech recognition

});

</script>

</body>

</html>

**login.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Admin Login</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

font-family: Arial, sans-serif;

background-color: #f0f8ff;

margin: 0;

}

.login-container {

background-color: #ffffff;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

padding: 20px;

width: 300px;

text-align: center;

}

.login-container h2 {

color: #4CAF50;

margin-bottom: 20px;

}

.login-container input[type="text"],

.login-container input[type="password"] {

width: 100%;

padding: 10px;

margin: 10px 0;

border-radius: 5px;

border: 1px solid #ddd;

}

.password-container {

position: relative;

display: flex;

align-items: center;

}

.password-container input {

width: calc(100% - 40px);

}

.toggle-password {

position: absolute;

right: 10px;

cursor: pointer;

}

.forgot-password {

display: block;

margin-top: 10px;

color: #888;

font-size: 14px;

}

.login-button {

width: 100%;

padding: 10px;

background-color: #4CAF50;

border: none;

color: #fff;

font-size: 16px;

cursor: pointer;

border-radius: 5px;

}

.login-button:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<div class="login-container">

<h2>Admin Login</h2>

<form action="/login" method="post">

<input type="text" name="username" placeholder="Username" required>

<div class="password-container">

<input type="password" id="password" name="password" placeholder="Password" required>

<span class="toggle-password" onclick="togglePassword()">👁️</span>

</div>

<a href="#" class="forgot-password">Forgot Password?</a>

<button type="submit" class="login-button">Login</button>

{% if error %}

<p style="color: red;">{{ error }}</p>

{% endif %}

</form>

</div>

<script>

function togglePassword() {

constpasswordField = document.getElementById('password');

if (passwordField.type === 'password') {

passwordField.type = 'text';

} else {

passwordField.type = 'password';

}

}

</script>

</body>

</html>

**faq\_management.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>FAQ Management</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

background-color: #f0f8ff;

}

.header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 20px;

padding: 10px;

background-color: white;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.header h1 {

color: #4CAF50;

margin: 0;

}

.logout-btn {

padding: 10px 20px;

background-color: #ff4444;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

text-decoration: none;

}

.faq-container {

background-color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.faq-item {

border: 1px solid #ddd;

margin-bottom: 10px;

padding: 15px;

border-radius: 5px;

}

.faq-content {

margin-bottom: 10px;

}

.faq-content h3 {

margin: 0 0 5px 0;

color: #333;

}

.faq-content p {

margin: 0 0 10px 0;

color: #666;

}

.edit-form {

display: none;

margin-top: 10px;

}

.edit-form textarea {

width: 100%;

padding: 8px;

margin: 5px 0;

border: 1px solid #ddd;

border-radius: 4px;

resize: vertical;

min-height: 60px;

}

.button-group {

display: flex;

gap: 10px;

margin-top: 10px;

}

.save-btn, .edit-btn, .delete-btn, .cancel-btn {

padding: 8px 15px;

border: none;

border-radius: 4px;

cursor: pointer;

font-weight: bold;

}

.save-btn {

background-color: #4CAF50;

color: white;

}

.edit-btn {

background-color: #2196F3;

color: white;

}

.cancel-btn {

background-color: #999;

color: white;

}

.delete-btn {

background-color: #ff4444;

color: white;

}

.error-message {

color: #ff4444;

margin-top: 5px;

}

.status-popup {

display: none;

position: fixed;

top: 20px;

right: 20px;

padding: 15px 25px;

border-radius: 5px;

color: white;

font-weight: bold;

box-shadow: 0 2px 10px rgba(0,0,0,0.2);

z-index: 1000;

animation: slideIn 0.3s ease-out;

}

.status-popup.success {

background-color: #4CAF50;

}

.status-popup.error {

background-color: #ff4444;

}

@keyframes slideIn {

from {

transform: translateX(100%);

opacity: 0;

}

to {

transform: translateX(0);

opacity: 1;

}

}

</style>

</head>

<body>

<div class="header">

<h1>FAQ Management</h1>

<a href="/logout" class="logout-btn">Logout</a>

</div>

<div class="faq-container">

<div id="faq-list">

<!-- FAQs will be dynamically loaded here -->

</div>

</div>

<div id="statusPopup" class="status-popup"></div>

<script>

// Function to fetch and display FAQs

async function loadFAQs() {

try {

const response = await fetch('/api/queries');

if (!response.ok) {

throw new Error('Failed to fetch FAQs');

}

constfaqs = await response.json();

constfaqList = document.getElementById('faq-list');

faqList.innerHTML = '';

faqs.forEach(faq => {

constfaqElement = createFAQElement(faq);

faqList.appendChild(faqElement);

});

} catch (error) {

console.error('Error loading FAQs:', error);

showStatusMessage('Failed to load FAQs. Please try again later.', false);

}

}

// Function to create an FAQ element

function createFAQElement(faq) {

const div = document.createElement('div');

div.className = 'faq-item';

div.setAttribute('data-faq-id', faq.id);

div.innerHTML = `

<div class="faq-content">

<h3>Question:</h3>

<p class="question-text">${faq.question}</p>

<h3>Answer:</h3>

<p class="answer-text">${faq.answer}</p>

</div>

<div class="edit-form">

<h3>Edit Question:</h3>

<textarea class="edit-question">${faq.question}</textarea>

<h3>Edit Answer:</h3>

<textarea class="edit-answer">${faq.answer}</textarea>

</div>

<div class="button-group">

<button onclick="startEdit(${faq.id})" class="edit-btn">Edit</button>

<button onclick="deleteFAQ(${faq.id})" class="delete-btn">Delete</button>

</div>

`;

return div;

}

// Function to toggle between editing and updating

async function startEdit(id) {

constfaqItem = document.querySelector(`[data-faq-id="${id}"]`);

consteditButton = faqItem.querySelector('.edit-btn');

if (editButton.textContent.trim() === 'Update') {

// Update FAQ

const question = faqItem.querySelector('.edit-question').value;

const answer = faqItem.querySelector('.edit-answer').value;

try {

const response = await fetch('/api/update\_faq', {

method: 'PUT',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ id, question, answer })

});

if (!response.ok) {

throw new Error('Failed to update FAQ');

}

// Update UI

faqItem.querySelector('.question-text').textContent = question;

faqItem.querySelector('.answer-text').textContent = answer;

faqItem.querySelector('.edit-form').style.display = 'none';

faqItem.querySelector('.faq-content').style.display = 'block';

editButton.textContent = 'Edit';

showStatusMessage('FAQ updated successfully!', true);

} catch (error) {

console.error('Error:', error);

showStatusMessage('Failed to update FAQ.', false);

}

} else {

// Enable editing

faqItem.querySelector('.faq-content').style.display = 'none';

faqItem.querySelector('.edit-form').style.display = 'block';

editButton.textContent = 'Update';

}

}

// Function to delete FAQ

async function deleteFAQ(id) {

if (confirm('Are you sure you want to delete this FAQ?')) {

try {

const response = await fetch(`/api/delete\_faq/${id}`, {

method: 'DELETE'

});

if (!response.ok) {

throw new Error('Failed to delete FAQ');

}

// Remove the FAQ from the DOM

document.querySelector(`[data-faq-id="${id}"]`).remove();

showStatusMessage('FAQ deleted successfully!', true);

} catch (error) {

console.error('Error:', error);

showStatusMessage('Failed to delete FAQ.', false);

}

}

}

// Function to show status messages

function showStatusMessage(message, isSuccess) {

const popup = document.getElementById('statusPopup');

popup.textContent = message;

popup.className = `status-popup ${isSuccess ? 'success' : 'error'}`;

popup.style.display = 'block';

setTimeout(() => {

popup.style.display = 'none';

}, 3000);

}

// Load FAQs on page load

document.addEventListener('DOMContentLoaded', loadFAQs);

</script>

</body>

</html>