Intelligent Enterprise Assistant: Enhancing Organizational Efficiency through AI-driven Chatbot Integration

A MINI PROJECT REPORT

Submitted by

ANUSHRI T(917722H008)
SRI DARSHINI M(917722H056)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND BUSINESS SYSTEMS

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI – 15
(A Government Aided Autonomous Institution Affiliated to Anna University)

ANNA UNIVERSITY: CHENNAI 600 025

NOV 2024

THIAGARAJAR COLLEGE OF ENGINEERING
MADURAI-15

(A Government Aided Autonomous Institution Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report **"Intelligent Enterprise Assistant: Enhancing Organizational Efficiency through AI-driven Chatbot Integration"** is the bonafide work of **"ANUSHRI T (917722H008) , SRI DARSHINI M (917722H056)"** who carried out the Mini project work under my supervision during the Academic Year 2024-2025.

SIGNATURE
Dr.P.Chitra, M.E., Ph.D.,
FACULTY INCHARGE &
HEAD OF THE DEPARTMENT
COMPUTER SCIENCE AND BUSINESS SYSTEMS
THIAGARAJAR COLLEGE OF ENGG.
MADURAI – 625 015

Submitted for the VIVA VOCE Examination held at Thiagarajar College of Engineering on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Acknowledgement

I wish to express my deep sense of gratitude to **Dr.L.Ashok Kumar,** Principal of Thiagarajar College of Engineering for his support and encouragement throughout this project work.

I wish to express my sincere thanks to **Dr.P.Chitra**, Head of the Department of Computer Science and Business Systems for her support and ardent guidance.

I owe my special thanks and gratitude to the faculty incharge Dr.P.Chitra Professor, Department of Computer Science and Business Systems for her guidance and support throughout our project.

I am also indebted to all the teaching and non-teaching staff members of our college for helping us directly or indirectly by all means throughout the course of our study and project work.

I extremely thank my parents, family members and friends for their moral support and encouragement for my project.

ABSTRACT

In the modern digital landscape, the use of chatbots has revolutionized the way businesses and organizations interact with their customers. This project introduces a secure and intelligent chatbot platform designed for IT support and customer service environments. The chatbot is built using Hugging Face's large language models (LLMs) such as BERT and GEMMA , providing high-quality, context-aware responses to user inquiries. Key features of the platform include a user management system that supports registration, login, and email-based two-factor authentication (2FA) to safeguard user data. Additionally, the chatbot interface, built with Streamlit, is intuitive and easy to use, allowing users to ask questions and receive instant, AI-generated responses. Chat histories are stored in a SQLite database, offering the potential for future analysis and improvements based on user interactions.

One of the primary challenges addressed by this project is ensuring that the chatbot platform is both secure and scalable, without compromising on ease of use. By implementing OTP-based 2FA, the system protects against unauthorized access, making it suitable for use in environments where sensitive data may be involved. The Hugging Face Models provides the backend for the chatbot, enabling it to handle a wide range of queries effectively, while SQLite ensures that all user data, including chat history, is stored in a lightweight yet efficient manner.

Table of Contents

List of Figures

List of Abbreviations

OTP	**One Time Password**
2FA	**Two Factor Authentication**
LLM	**Large Language Model**
API	**Application Programming Interface**
SMTP	**Simple Mail Transfer Protocol**
SQUAD	**Stanford Question Answering Dataset**
BERT	**Bidirectional Encoder Representations**

INTRODUCTION

With the growing need for personalized and responsive user interfaces, chatbots have become a pivotal tool in customer engagement and support systems. This project presents a comprehensive chatbot platform that offers seamless interaction combined with a strong layer of security through two-factor authentication (2FA) and loads a SQuAD-like dataset and tokenizes it for training. Then, Tokenizes the dataset for input to the model. The chatbot, powered by Hugging Face's pre-trained models, can assist users by generating natural language responses to a variety of queries. The project integrates essential components such as user registration, login, and OTP verification, ensuring a secure and user-friendly experience. By utilizing SQLite as the backend database, this system not only stores user credentials but also maintains a history of all chatbot interactions, providing valuable insights into user behavior. The entire platform is built using Streamlit, a powerful Python-based framework that allows developers to create interactive and intuitive web applications with minimal effort.

In addition to security and user management, the chatbot platform leverages the power of Hugging Face’s Gemma-2b-it model, Gradio hosted model to deliver highly contextual and coherent responses. By using advanced natural language processing (NLP) techniques, the system can understand and respond to user queries in real time, making it an ideal solution for customer service, IT support, and other interaction-driven environments. The chatbot's ability to generate dynamic responses based on user input enhances user satisfaction by providing timely and relevant information. Furthermore, the integration of a chat history feature allows users to revisit past interactions,

while administrators can analyze chat logs for trends and improvements. This combination of personalized responses and robust data handling positions the chatbot as a scalable solution that can adapt to various business needs.

To further enhance the platform's usability, Streamlit is employed to provide an intuitive and responsive user interface. Streamlit simplifies the process of building web applications, enabling real-time updates and smooth interactions with minimal coding effort. The platform allows users to easily register, log in, and engage with the chatbot, ensuring a seamless flow from authentication to conversation. With its clean and interactive design, Streamlit ensures that even users with limited technical expertise can navigate the platform effortlessly. This approach makes the chatbot highly accessible, whether it's for internal use within organizations or as a customer-facing service, offering a balance between functionality, security, and ease of use.

LITERATURE SURVEY

Chatbots have become increasingly prominent across various domains due to their ability to enhance user interaction, improve service efficiency, and reduce response times. Several studies have focused on improving chatbot security, effectiveness, and sentiment analysis by leveraging advanced algorithms and datasets.

In the paper *"Comprehensive framework for implementing blockchain-enabled federated learning and full homomorphic encryption for chatbot security system"* (Springer, 2024), the authors propose a framework that enhances chatbot security using **Full Homomorphic Encryption (FHE)** and **blockchain-enabled federated learning**. FHE enables chatbots to process encrypted data, ensuring sensitive information remains secure during inference tasks. Tested using the **SentiWordNet** and **MNIST** datasets, this approach prevents data breaches without compromising functionality. Decentralized model training via federated learning safeguards user privacy, making this framework ideal for industries like healthcare and banking.

The paper *"Intelligent chatbot interaction system capable for sentimental analysis using hybrid machine learning algorithms"* (Information Processing and Management, 2023) introduces a hybrid approach to sentiment analysis, combining **BRNN-LSTM**, **Naïve Bayes Classifier (NBC)**, and **Fuzzy Naïve Bayesian Classifier (FNB)**. Tested on a **Reddit conversations dataset**, the **BRNN-FNB** model achieved **93% accuracy** using the Seq-to-Seq technique, showcasing its effectiveness in understanding user emotions and enhancing chatbot performance. This hybrid model is crucial for improving customer service interactions and social media management.

In *"How to Make chatbots productive – A user-oriented implementation framework"* (International Journal of Human-Computer Studies, 2023), the focus is on creating chatbots that align with user needs. By employing **Convolutional Neural Networks (CNN)** for text classification, the study emphasizes the importance of user acceptance and practicality in chatbot design. Using datasets from industry dialogues, the proposed framework integrates design principles to enhance user experience, advocating for large-scale studies to test its efficiency.

The three papers address different aspects of chatbot development. The first focuses on **security and privacy**, offering solutions for data-sensitive industries using FHE and blockchain. The second highlights **sentiment analysis** with hybrid models for better emotional understanding, while the third emphasizes **user-centric design**, advocating for practical and widely accepted chatbot systems. Integrating these approaches can lead to secure, effective, and user-friendly chatbot solutions that meet modern enterprise demands.

PROBLEM DEFINITION AND BACKGROUND

In today's digital age, user engagement and interaction have become vital aspects for businesses, especially in customer service and support systems. The growing popularity of Artificial Intelligence (AI) and Machine Learning (ML) technologies has led to the widespread adoption of **chatbots** across various sectors, enabling organizations to offer real-time support, reduce operational costs, and enhance the overall user experience. Chatbots can handle a variety of tasks, from answering common questions to assisting in complex problem-solving, creating an efficient communication channel.

However, as the demand for personalized and real-time responses grows, security concerns have also escalated. Chatbots often handle sensitive user data, such as personal information and authentication credentials, which makes them an attractive target for cyberattacks. **Ensuring secure access** to chatbot systems, along with user authentication, has become a pressing challenge. Unauthorized access to chatbot systems can result in data breaches, identity theft, and financial loss.

At the same time, there is a need for a lightweight, scalable, and easy-to-use system that can be deployed by developers without requiring extensive infrastructure. Combining an intuitive interface with strong security measures like **Two-Factor Authentication (2FA)** can significantly improve the reliability and safety of chatbot platforms.

Problem Statement

There is a need for a secure, user-friendly chatbot platform that offers seamless, real-time interaction while ensuring that user data and

authentication processes are well protected. The platform should integrate **Two-Factor Authentication (2FA)** for secure login, **session management** for persistent chat history, and should be easily deployable with minimal infrastructure requirements. Additionally, the solution should be lightweight and intuitive, offering a smooth user experience for both technical and non-technical users.

Current chatbot solutions often lack the combination of **robust security**, **easy-to-use interface**, and **efficient chat management** that this project aims to address. By leveraging **Streamlit** for simplicity, **SQLite** for secure data storage, and **Hugging Face models** for intelligent responses, this project seeks to fill these gaps, providing a scalable solution that can be adopted across industries without compromising on user security or experience.

Existing Approach

Several existing chatbot platforms focus on addressing the interaction problem with the use of **pre-trained AI models** (e.g., GPT, BERT, or T5). These models are highly effective at understanding and generating human-like responses to user queries. However, most of these systems overlook essential features such as user authentication and secure storage of data.

1. **Basic User Authentication**:
    - Some chatbot systems implement simple username and password authentication. While this provides basic access control, it is vulnerable to attacks like phishing, brute-force attempts, and password leaks.
    - Additionally, in cases where passwords are stored as plain text, there is a significant risk of data breaches.

2. **Lack of Advanced Security Measures**:
    - Many chatbots do not employ advanced security mechanisms like Two-Factor Authentication (2FA), which provides a critical extra layer of protection. In the absence of 2FA, if a user's password is compromised, attackers can easily gain unauthorized access to the chatbot.
3. **Limited Data Storage**:
    - While chatbots handle real-time interactions, there are limited systems that provide comprehensive **storage of chat history** linked to users for later retrieval. The lack of this feature affects user experience, as conversations cannot be resumed from where they were left off, and important information exchanged in the chat is lost.
4. **Heavy, Complex Systems**:
    - Many existing chatbot platforms are built on complex architectures that require significant resources, making them less suitable for local or smaller-scale deployments where simplicity and portability are prioritized.

REQUIREMENTS

1. Python Libraries and Modules

a. **sqlite3**: For database operations.
b. **hashlib**: For password hashing.
c. **smtplib**: To send OTP emails.
d. **random**: To generate random OTPs.
e. **email.message**: For formatting and sending emails.
f. **os**: To access environment variables (e.g., Hugging Face token).
g. **torch**: For running machine learning models.
h. **streamlit**: For the web interface.
i. **huggingface_hub**: To interact with Hugging Face models
j. **BeautifulSoup:** Parses and extracts content from HTML/XML documents.
k. selenium: Imports modules for using Selenium WebDriver to automate web interactions.

2. Hugging Face Token

Obtain an API key from Hugging Face to access the model. Ensure the API token is set in your environment variables.

3. Email Service Setup

a. Configure an SMTP server to send OTPs, using Gmail's SMTP:
b. Set up a Gmail account and enable "Less secure apps" or create an app-specific password.
c. Update the email configuration in the code

4. Database (SQLite)

The project uses an SQLite database to manage user data and chat history. Ensure the database has proper write and read permissions on your system.

5. Two-Factor Authentication (2FA)

OTP generation and verification are implemented using the `random` module, and OTPs are sent via email using smptlib. The OTP is stored in the session state for verification.

6. Gemma Chatbot Integration

The chatbot integrates with a model hosted on Hugging Face, responding to user queries using either the InferenceClient or Gradio Client. Ensure access to models like Gemma 2.2B or Llama 3.2B.

7. Bert Model Integration

The BERT model in this code is used for fine-tuning and answering questions based on context, **DistilBERT** for question-answering tasks.The BERT framework was pretrained using text from Wikipedia and can be fine-tuned with question-and-answer data sets.

8. Gradio Client API

The **Gradio Client API** is a Python library provided by Gradio that allows developers to interact programmatically with machine learning models hosted on Gradio interfaces. It enables sending input data to a Gradio application and receiving the model's output seamlessly.

9. Web Interface (Streamlit)

The project features an interactive interface built with Streamlit. Maintain the required folder structure for server deployment.

10. Security Considerations

Store environment variables securely. Passwords are hashed with SHA-256 for security.

11. Deployment Requirements

a. **Local Deployment**: Run the project locally with the necessary libraries.
b. **Cloud Deployment**: Consider platforms like Streamlit Cloud, Heroku, or AWS for hosting. Securely set environment variables for API tokens and email credentials.

CONTROL FLOW GRAPH

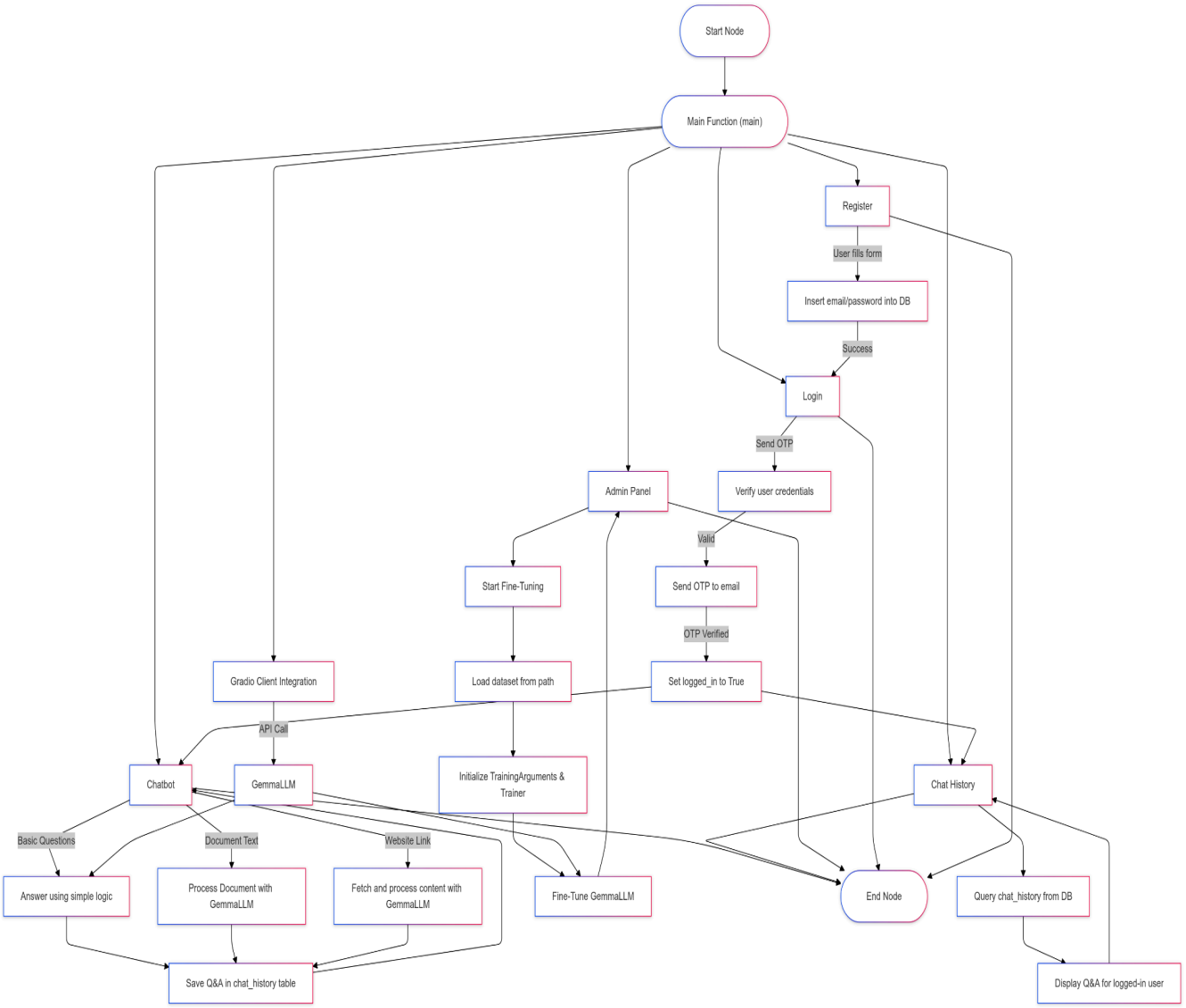

FIG 1. CONTROL FLOW GRAPH

PROPOSED METHODOLOGY

1. Database Connection Module Using SQLite

The database connection module is responsible for managing the storage of user data (such as email and password) and the chat history. SQLite is used for this purpose due to its simplicity, portability, and serverless architecture, making it ideal for local applications.

Steps in Database Setup and Operations:

- ❖ Database Initialization (Creating Tables):
  - ➢ The SQLite database is initialized with two tables:
    - users table: Stores user information, including the email and hashed password.
    - chat_history table: Stores the chat messages exchanged between the user and the system. It links messages to users using user_id as a foreign key.
  - ➢ The users table uses an auto-incremented `id` as the primary key.
  - ➢ The chat_history table stores each message, along with the user's user_id and the role (either user or bot).
- ❖ User Registration:
  - ➢ During registration, the system checks if the email already exists in the database. If the email is new, the password is hashed using SHA-256 before storing it in the database.
    - Each registered email must be unique.
    - Passwords are stored in their hashed form, not as plain text, ensuring secure storage.

- ❖ User Login:
  - ➢ The login function retrieves the hashed password from the database and compares it with the hashed version of the entered password.
  - ➢ A successful login returns the user's ID, which is used for session management and storing chat history.
- ❖ Chat History Management:
  - ➢ After login, all chat messages between the user and the chatbot are stored in the chat_history table. Each message is linked to the user's user_id.
    - ■ Chat history can be retrieved later by querying the chat_history table using the user_id.
    - ■ This provides a persistent record of all conversations.

2. Two-Factor Authentication (2FA)

Objective:
To implement an additional layer of security for user authentication by using a combination of:

1. User-provided email and password.
2. One-Time Password (OTP) sent to the user's email.

This two-step process ensures that both the login credentials and the OTP are verified before granting access to the system.

Steps in 2FA Implementation:

1. User Login :
   - ➢ Users enter their email and password on the login screen.
   - ➢ The system checks whether the provided email exists in the database by querying the users table.

- ➢ For security, the password is hashed using SHA-256 before being compared to the hashed password stored in the database. This ensures that plain-text passwords are never exposed or stored.
- ➢ Passwords are securely hashed using SHA-256.
- ➢ If the email and password match a record in the database, the user proceeds to the next step; otherwise, the login fails.

2. Generate OTP :
   - ➢ Once the user's email and password are validated, a 6-digit OTP is generated using the random module.
   - ➢ This OTP is unique to the user session and is stored in memory (session state) until the user completes the verification process.
   - ➢ The OTP is randomly generated and consists of 6 digits, adding an extra layer of security.
   - ➢ The OTP is stored temporarily in the session state for further comparison.
3. Send OTP via Email:
   - ➢ The generated OTP is sent to the user's registered email using the smtplib library. This ensures that the OTP reaches the correct user, preventing unauthorized access.
   - ➢ An SMTP connection is established with Gmail's server, and the OTP is delivered in the email's body.
   - ➢ The system uses an email server (SMTP) to send the OTP.
   - ➢ Only the user with access to the email can retrieve the OTP.
4. OTP Verification:
   - ➢ After receiving the OTP, the user enters it into the application.
   - ➢ The entered OTP is compared to the OTP stored in the session state. If they match, the login is successful, and the user is granted access to the application. Otherwise, an error message is displayed.
   - ➢ The OTP is validated against the stored OTP.

- ➢ Upon successful verification, the user is authenticated and allowed to proceed.

3. Chatbot interaction module:

The Chatbot Interaction Module is central to the user experience, enabling seamless communication between users and the chatbot powered by Hugging Face's pre-trained models. This module processes the input from users, interacts with the pre-trained model, and provides relevant responses based on the queries.

Key Features of the Chatbot Interaction Module:

- ❖ User Input Handling:
  - ➢ User inputs are taken via the Streamlit interface, which provides a text box for the user to type their query or message.
  - ➢ The input is sanitized to prevent the injection of malicious scripts or commands.

    user_message = st.text_input("Enter your message here:")
- ❖ Interaction with Hugging Face Models:
  - ➢ The system uses a Hugging Face pre-trained model for generating responses.
  - ➢ When the user submits a message, the chatbot sends the query to the NLP model, which generates a relevant response based on the context.

    from transformers import pipeline

    chatbot = pipeline("conversational")

    response = chatbot(user_message)
- ❖ Interaction with Gradio Client:

- ➢ The system uses a Gradio-hosted pre-trained model for generating responses.
- ➢ When the user submits a message, the chatbot sends the query to the model through the Gradio Client API. The model processes the input and generates a relevant response based on the context.
- ➢ The Gradio Client allows seamless integration with Hugging Face-hosted models by specifying the model's URL and endpoint parameters.

❖ Response Generation:

- ➢ The response from the model is displayed in the chatbot window within the Streamlit interface.
- ➢ All responses are linked to the user's session for storing in the chat_history table, preserving the conversation context for future retrieval.

  st.write("Chatbot:", response)

❖ Session Management:

- ➢ Every interaction is stored with the user's session ID, and upon successful login, the session ID is generated. This helps the system in identifying and maintaining continuity in user conversations.

❖ Context-Aware Conversations:

- ➢ The chatbot supports context-aware conversations by maintaining the dialogue history. Each query-response pair is processed in the context of previous interactions, allowing for a coherent and natural conversational experience.

❖ Supervised fine-tuning:

- ➢ This code uses **supervised fine-tuning** on a SQuAD-like dataset, where the model is trained to predict the start and end positions of answers within a given context for a question-answering task. The fine-tuning process adapts the pretrained BERT model to the

specific structure and domain of the input dataset. The role of fine-tuning here is to enhance the model's performance in extracting accurate answers from custom datasets.

- ❖ BERT Model Integration:
  - ➢ The BERT model in this code is used for fine-tuning and answering questions based on context, leveraging **DistilBERT** for question-answering tasks. It processes input questions and context to predict start and end positions of the answer within the text.

RESULTS

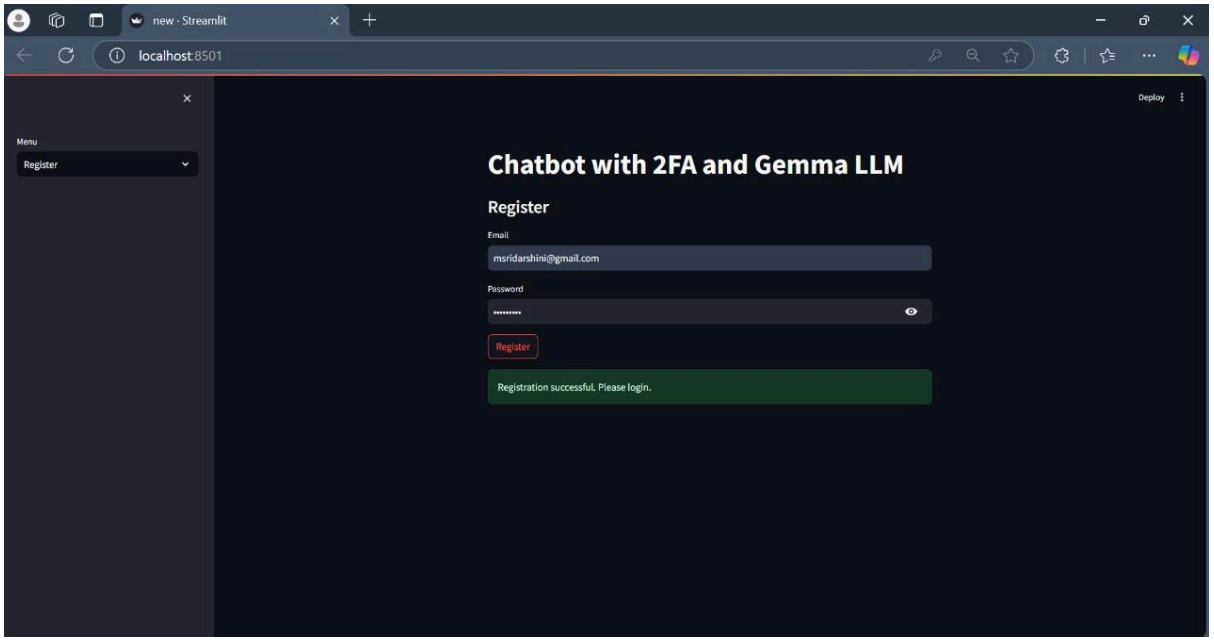

FIG 2. REGISTRATION PAGE

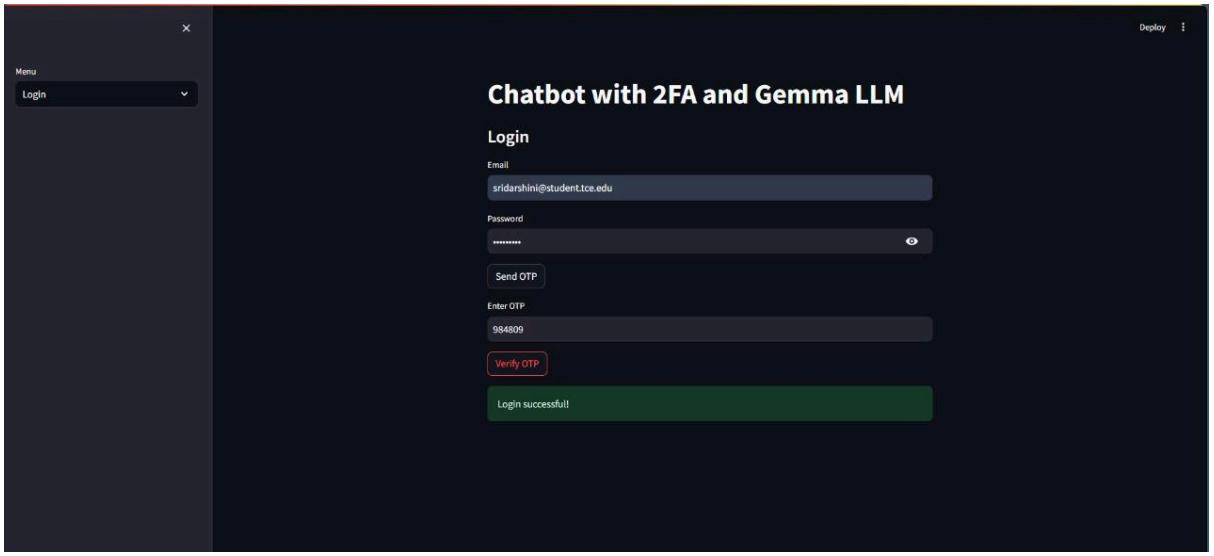

FIG 3. LOGIN PAGE

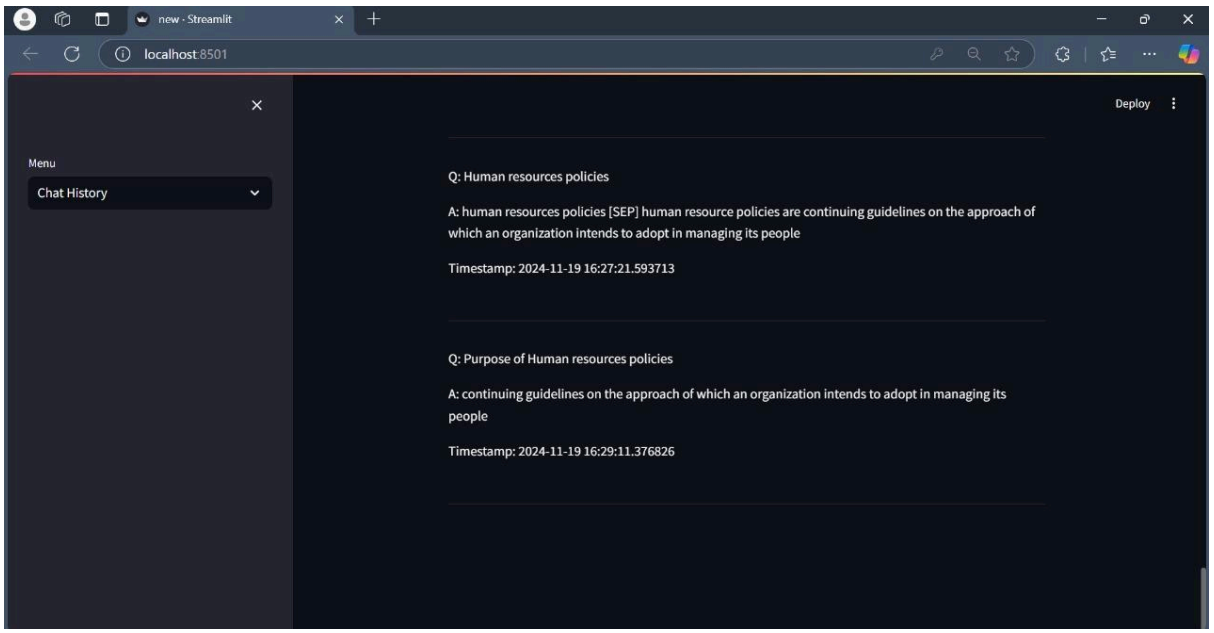

FIG 4. CHAT HISTORY

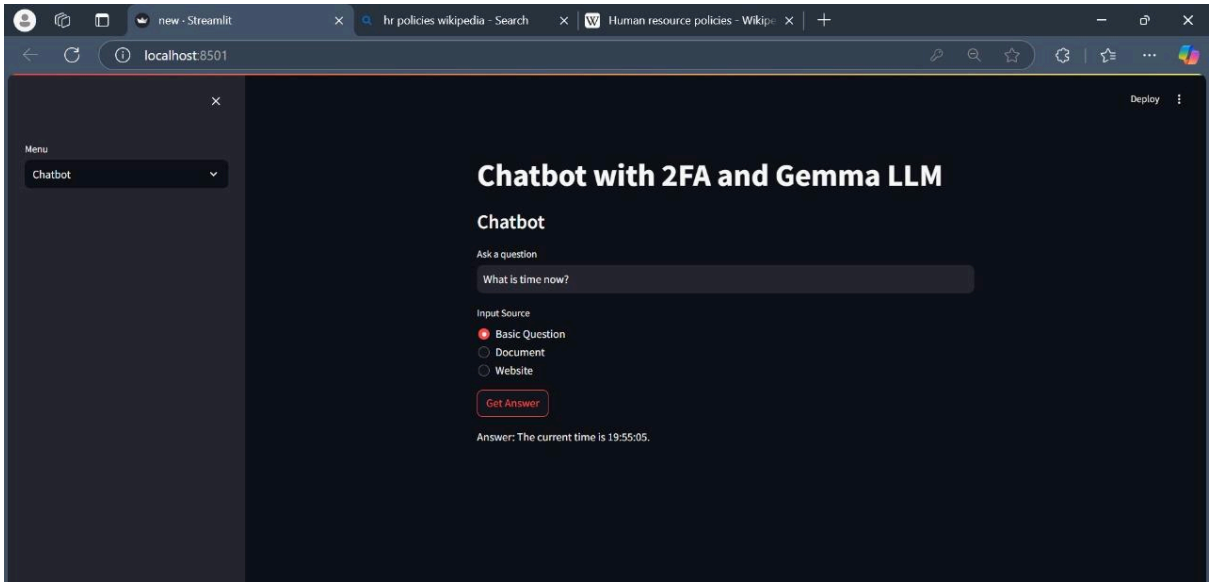

FIG 5. BASIC QUERIES

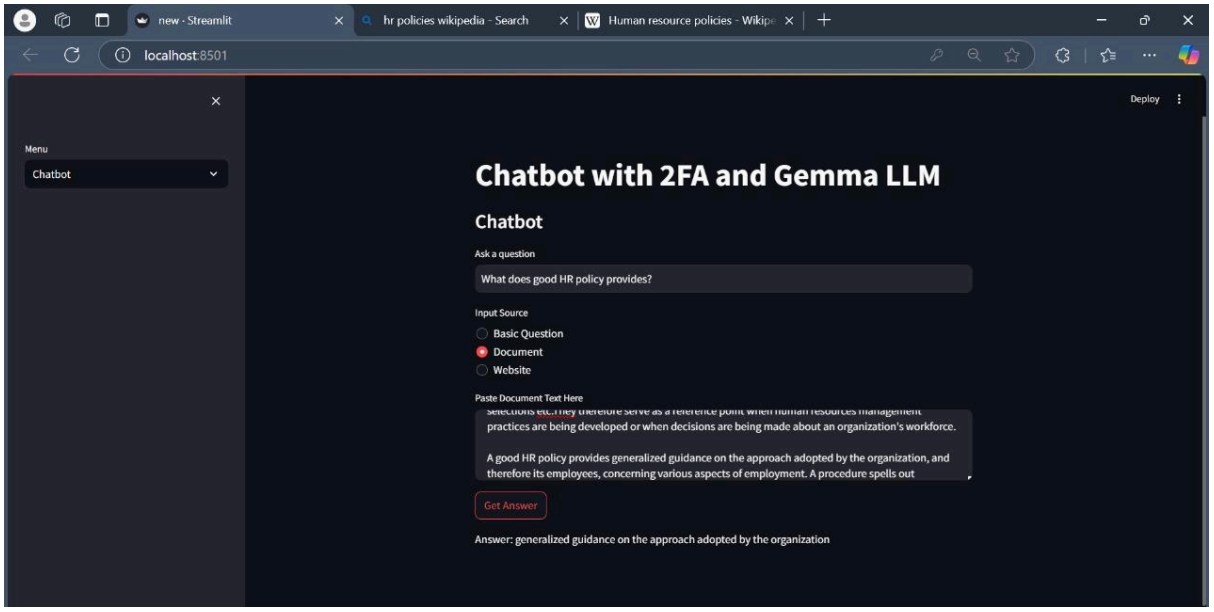

FIG 6. TEXT BASED QUERIES

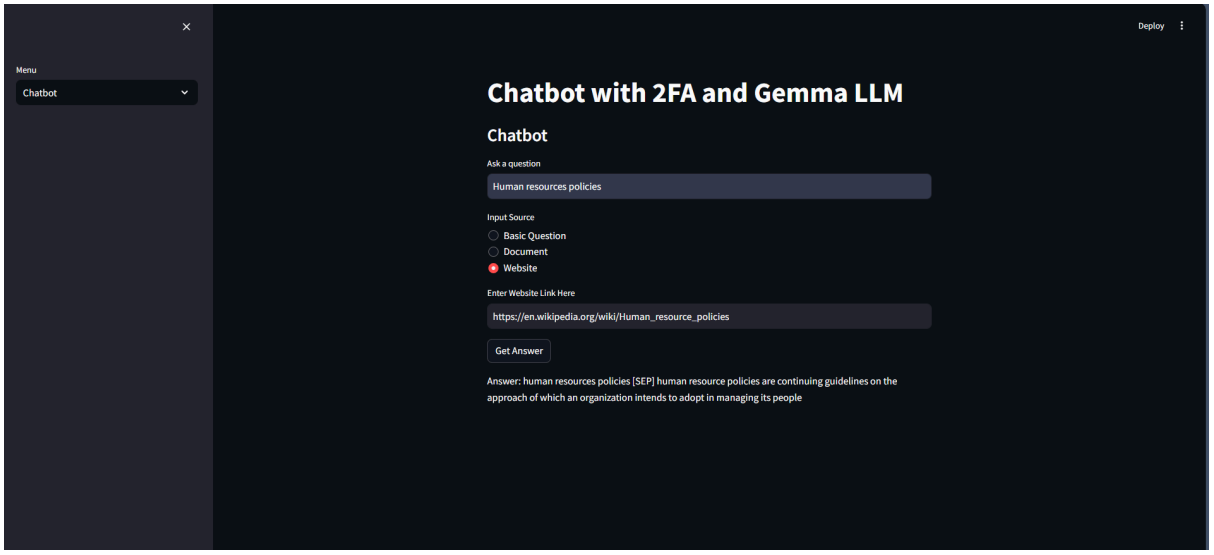

FIG 7. WEBSITE BASED QUERIES

FIG 8. ANSWERS FROM GRADIO-CLIENT

CONCLUSION

The development of the IT support chatbot solution with integrated user authentication and two-factor authentication (2FA) provides a highly secure and user-friendly platform for delivering AI-driven support services. By incorporating a robust database system for managing user credentials and chat history, the platform ensures that both personal information and conversational data are stored securely. The use of hashing for password storage and OTP-based 2FA strengthens the security framework. The chatbot, powered by advanced models such as Gemma 2B, provides accurate and timely responses to user queries, enhancing overall user satisfaction. The interface, built using Streamlit, is simple and intuitive, allowing users to easily register, log in, and interact with the chatbot.

One of the key strengths of the project lies in its adaptability. By using Hugging Face's API, BERT, GEMMA, Supervised Fine-tuning and Gradio, the chatbot is capable of delivering contextually relevant responses. The integration of a scalable and flexible database allows for future expansion, including the the enhancement of the chatbot's conversational abilities. Furthermore, the system's real-time feedback mechanism allows the chatbot to improve based on user interactions, making it a continuously evolving tool.

However, there are areas that could be improved. While the chatbot provides solid responses for common IT support queries, it is still limited by the dataset and model used. Furthermore, the reliance on email-based OTP for 2FA, while secure, might present delays or accessibility issues for some users. Despite these limitations, the overall system represents a significant step toward automating and securing IT support services. The project demonstrates how modern AI technologies can be leveraged to create innovative, practical, and secure solutions to common enterprise problems.

FUTURE WORKS

1. **Model Enhancement and Fine-tuning**: Future improvements would include fine-tuning the chatbot's AI model with more specific IT support datasets. Training it with real-time user data would enhance its accuracy and ability to handle complex queries.
2. **Voice Interaction and Multimodal Capabilities**: Integrating voice interaction and multimodal support (images/videos) would improve the user experience, enabling hands-free use and allowing visual troubleshooting through uploaded content.
3. **Integration with External IT Systems**: Integrating with platforms like Jira or ServiceNow would streamline communication between the chatbot and IT departments, enabling automatic ticket creation and real-time updates.
4. **Mobile App Development**: Developing a mobile app would enable users to access IT support on the go, with features like push notifications for OTP and a seamless, 24/7 support experience.
5. **Internationalization and Multilingual Support**: Adding multilingual support and localizing the platform for different regions would make the system more inclusive and globally applicable to a diverse user base.
6. **Self-Learning and Continuous Improvement**: Implementing a self-learning mechanism with reinforcement learning would enable the chatbot to evolve continuously by analyzing past interactions and improving its responses over time.
7. **Scalability and Cloud Integration**: Cloud solutions like AWS or Azure would ensure scalability, allowing the platform to handle growing user demands while maintaining reliability and responsiveness.