JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Desktop Chatbot System Integrating ChatGPT and NLP for Academic Support at SVPM College

Prof. Y.R. Khalate*1, Vaishnavi Pawar*2, Rashmi Sampagar*3, Samruddhi Shejal*4, Sanika Nigade*5

*1 Assistant Professor, Department of Computer Science, SVPM's College of Engineering, Malegaon (Bk), Maharashtra, India.

*2,3,4,5 Student, Department of Computer Science, SVPM's College of Engineering, Malegaon (Bk), Maharashtra, India.

Abstract: ***To enhance academic support and streamline student interactions in higher education, this paper presents the development of a Python-based desktop application tailored for SVPM College, Malegaon (Bk), Baramati. The system integrates a dual-chatbot framework designed to provide intelligent, responsive assistance for both general and domain-specific queries. The first chatbot utilizes OpenAI's ChatGPT API to deliver real-time, context-aware responses to academic and general-purpose questions. In parallel, a second chatbot is developed using Natural Language Processing (NLP) techniques and machine learning algorithms trained on a structured JSON dataset encompassing frequently asked questions regarding admissions, hostel facilities, fee structures, and required documentation. The model is trained and serialized using the Pickle module, enabling efficient query classification and response generation. The application is developed using the Spyder IDE within the Anaconda Navigator environment and incorporates advanced Python libraries including TensorFlow, Keras, OpenCV, and Pillow. This system reduces manual administrative workload, centralizes information access, and enhances user engagement through an intuitive interface. By combining a general-purpose AI engine with a domain-specific NLP model, the application offers a scalable and extensible solution tailored to the needs of students and faculty, with future potential for multilingual support and integration of additional institutional datasets.***

Keywords: Natural Language Processing, JSON Data, Key Word Matching, Pickle Module, Machine Learning, Academic Automation, OpenAI API.

1. INTRODUCTION

The rapid advancement of artificial intelligence (AI) has significantly transformed educational support systems. Modern academic institutions are increasingly adopting AI-driven solutions to enhance communication, automate responses, and provide timely, accurate information to students. Among these innovations, chatbot systems have gained prominence due to their ability to efficiently handle large volumes of queries with 24/7 availability. The implications of such forgeries are far-reaching. In law enforcement, a single tampered video can mislead investigations or serve as falsified evidence. In digital journalism, the spread of manipulated media can damage reputations, propagate misinformation, and fuel societal unrest. Similarly, in surveillance and national security, any undetected video alteration can result in flawed intelligence and compromised decisions.

This paper presents a novel dual-chatbot desktop application developed for SVPM College, Malegaon (Bk), Baramati, aimed at improving student and visitor interaction through intelligent dialogue systems. The system integrates two distinct chatbot models to address both general-purpose and domain-specific queries.

The first chatbot utilizes OpenAI's ChatGPT, a state-of-the-art transformer-based language model capable of generating contextually rich and human-like responses. This model is integrated to support dynamic conversations on a wide range of academic and general topics, offering a virtual assistant experience that mimics natural interaction.The backend of the system is implemented using the Flask web framework, offering RESTful endpoints and user session management. To ensure secure access, user registration and authentication are managed through an integrated SQLite database. The user interface is designed with HTML, CSS, and Jinja2 templating, offering a seamless experience for uploading videos, viewing analysis results, and understanding tampering timelines.

Complementing the general-purpose bot, the second chatbot is built using traditional Natural Language Processing (NLP) techniques. It is trained on a curated JSON dataset containing

frequently asked questions related to SVPM's admission process, hostel facilities, fee structures, and documentation requirements. The model is serialized using the Pickle module to ensure fast and accurate responses to college-specific queries.

Development was conducted using the Spyder IDE within the Anaconda Navigator environment. The application leverages a robust Python ecosystem, including libraries such as TensorFlow and Keras for model training, and OpenCV and Pillow for image handling and UI enhancements. This architecture ensures modularity, scalability, and adaptability for future improvements.

The proposed system exemplifies the integration of general-purpose AI with custom-trained NLP, offering a unified platform for academic and administrative assistance. It enhances access to institutional information for students, faculty, and visitors alike, contributing to the modernization of college support services through artificial intelligence.

II. SYSTEM OVERVIEW

The proposed system is a dual-mode intelligent desktop-based chatbot application specifically developed to enhance access to academic and administrative information for stakeholders of SVPM College, Malegaon (Bk), Baramati. It is designed to operate under both online and offline conditions, ensuring continuous availability and reliability. The architecture integrates two distinct conversational AI modules, each tailored for specific use cases, supported by a unified graphical user interface.

1. ChatGPT-Based Conversational Agent (Online Mode):
This module leverages the OpenAI ChatGPT API, a large-scale, transformer-based language model optimized for generating human-like natural language responses. It is designed to address open-domain queries related to academic concepts, career counseling, personal development, and general-purpose information retrieval. This component depends on active internet connectivity to access and interact with the remote ChatGPT model via RESTful API calls. It ensures high-quality dialogue management and contextual understanding using deep learning-based natural language generation (NLG).

2. Domain-Specific NLP Chatbot (Offline Mode):
The offline module is designed using classical Natural Language Processing (NLP) and Machine Learning (ML) techniques. It is trained on a structured dataset in JSON format, comprising frequently asked questions (FAQs) related to institutional processes such as admissions, fee structures, hostel accommodations, required documentation, and campus facilities. Preprocessing techniques including tokenization, stopword removal, stemming/lemmatization, and term vectorization (e.g., TF-IDF) are employed to prepare the textual data. A supervised learning algorithm, such as Support Vector Machines (SVM) or Logistic Regression, is trained and serialized as a .pkl (Pickle) file to enable lightweight, fast, and interpretable offline predictions.

3. User Interface (UI):
The desktop-based GUI is developed using Python's Tkinter or PyQt5 frameworks, offering cross-platform compatibility and a responsive interface. It enables users to toggle between the online (ChatGPT) and offline (domain-specific) modes seamlessly. The interface supports textual query input and displays response outputs in real time, emphasizing usability and accessibility.

4. System Implementation and Libraries:
The development and experimentation are conducted using the Spyder IDE within the Anaconda environment. Core libraries used include:

TensorFlow & Keras for any deep learning-based enhancements.

Scikit-learn for traditional ML model training and evaluation.

NLTK / spaCy for NLP preprocessing.

OpenCV & Pillow for UI/UX improvements, image handling, or future vision-based integrations.

This hybrid architecture ensures robustness, flexibility, and scalability. It not only accommodates diverse user needs and internet availability constraints but also demonstrates a practical application of integrating pre-trained large language models with customized domain-specific models in educational environments.

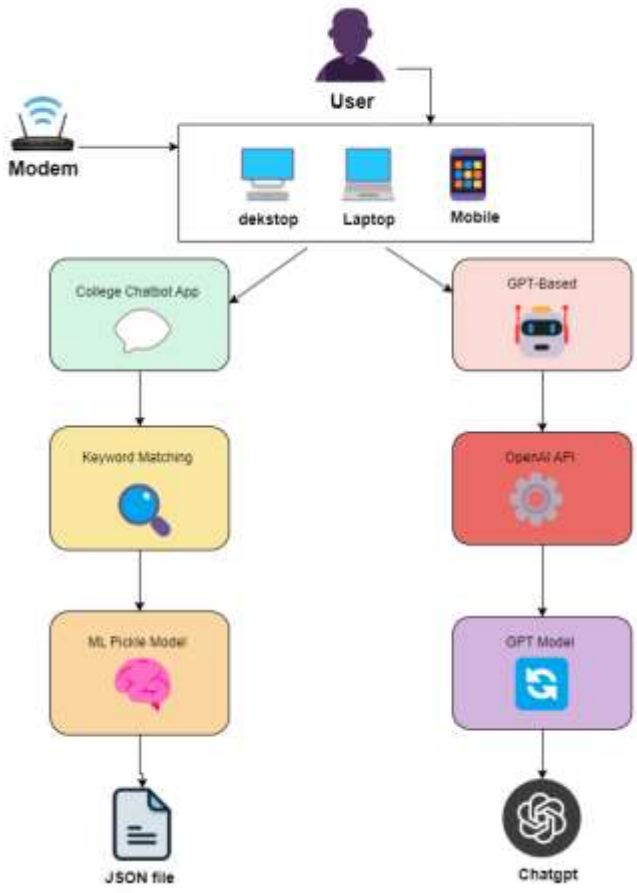

Fig: 1 System Architecture

III. FUNCTIONAL REQUIREMENTS

The SVPM College Chatbot Desktop Application is architected to fulfill a set of clearly defined functional requirements that ensure seamless interaction, intelligent response generation, and secure user experience. These requirements form the foundation for delivering a robust, responsive, and intuitive solution for college-related query resolution through AI-driven chat interfaces.

• Dual-Chatbot Integration

The application integrates two distinct chatbot engines to cater to both generic and domain-specific user queries:

1. **ChatGPT-Based Conversational Agent:** Utilizes OpenAI's ChatGPT API to handle general inquiries unrelated to college-specific data. This model leverages a state-of-the-art language model to understand and generate human-like responses, ensuring fluid and contextually relevant dialogue.
2. **NLP-Based College Query Bot:** A custom rule-based chatbot trained using a structured JSON dataset comprising predefined question-answer pairs relevant to SVPM College (e.g., admission process, hostel availability, fee structure, and document requirements). The dataset is processed and deployed using Python's Pickle serialization for efficient model loading and inference.

This dual-chatbot architecture ensures coverage of a wide spectrum of user questions with high accuracy and adaptability.

• Desktop GUI Interface

The system offers a user-friendly desktop application built using Python's Tkinter library (or optionally PyQt5), providing an interactive graphical user interface. Key features include:

- **Chat Interface:** Text input area for user queries and a scrollable response area for chatbot replies.
- **Chatbot Selection Panel:** Users can select between the ChatGPT and NLP bots based on the nature of their query.
- **Session Logs:** Maintains and displays the conversation history for the active session.

The GUI is designed for intuitive navigation, minimal user effort, and clear visual presentation.

• Backend Query Processing and Response Generation

The backend logic is responsible for handling user inputs, invoking the appropriate chatbot engine, and returning context-aware responses:

- **ChatGPT Integration:** Queries directed to ChatGPT are securely sent via API calls, and the responses are parsed and rendered in real time. The system ensures API key security through environmental variable management and error handling.
- **NLP Bot Prediction Pipeline:** The locally hosted bot uses keyword and intent-matching techniques to search the JSON dataset. If a matching query is found, the corresponding response is returned; otherwise, a default fallback message is triggered.

The backend is designed for lightweight, low-latency execution to ensure timely response delivery (within 2-3 seconds).

• Dataset Management and Model Training

The NLP bot is trained on a manually categorized JSON dataset, representing FAQs related to the college. The system supports:

- **Dataset Validation:** Ensures structural integrity and data consistency before training.
- **Model Serialization:** Trained model is serialized using Pickle for persistent storage and efficient runtime loading.
- **Dataset Expansion:** Future datasets can be integrated with minimal structural changes, supporting scalability.

This framework facilitates rapid model updates and continuous improvement in response accuracy.

• Real-Time Response Visualization

Upon query submission, responses from either bot are dynamically rendered in the GUI. Visual indicators include:

- **Bot Identifier Tags:** Each response is prefixed with the name of the responding bot (e.g., "[ChatGPT]" or "[NLP Bot]") to distinguish sources.
- **Formatted Output:** Responses maintain readability with appropriate text wrapping, padding, and font consistency.
- **Playback or Retry Options (Future Scope):** The design accommodates future enhancements like speech playback or retrying failed queries.

The focus remains on ensuring a smooth, professional, and real-time user experience.

IV. MODEL IMPLEMENTATION

The core intelligence of the SVPM College Chatbot Desktop Application lies in its hybrid chatbot architecture, which combines a cloud-based large language model (ChatGPT) with a locally trained rule-based NLP bot. These components are independently designed and deployed using OpenAI's API for ChatGPT integration and Python (with JSON and Pickle modules) for the local FAQ bot. This modular design enables both high-level conversational flexibility and reliable, domain-specific query handling.

ChatGPT-Based Conversational Agent (OpenAI API)

The ChatGPT component is implemented via OpenAI's GPT model API, responsible for handling general queries outside the domain-specific dataset. This integration supports natural language interaction through advanced context understanding.

1. **API Integration**: The application securely connects to OpenAI's GPT model using HTTPS REST API calls. The API key is managed using environmental variables to ensure security and prevent exposure in source code.
2. **Prompt Construction and Input Handling**: User queries are captured via the GUI and converted into prompts. These prompts are sent directly to the API, which handles tokenization and language modeling on the server side.
3. **Response Parsing**: The API returns JSON-formatted responses, which are parsed and displayed in the application's chat interface. If the API fails to respond, proper error-handling logic ensures user feedback is given.
4. **Deployment**: The ChatGPT integration is initialized at runtime using Python's requests module. Latency is optimized to ensure responses are returned within 2–3 seconds under normal conditions.

NLP-Based College Query Bot (Local Rule-Based Model)

To address domain-specific queries (e.g., admission, fees, documents), a rule-based chatbot is developed using a structured JSON dataset and implemented in Python.

1. **Model Architecture**: The chatbot uses a keyword and pattern-matching algorithm to map user queries to pre-defined questions stored in the JSON dataset. If a match is found, the corresponding answer is returned.
2. **Training and Serialization**: The dataset is manually curated and structured into categories. It is serialized using the pickle module, allowing for efficient loading and minimal startup delay.
3. **Query Matching and Response Logic**: Upon user input, the bot performs text normalization and tokenization, then searches for intent matches within the dataset. If no suitable match is found, a default fallback response is provided.
4. **Dataset Management**: The system supports modular updates to the dataset. New categories or questions can be added without retraining, supporting scalability and easy maintenance.
5. **Deployment**: The bot model is loaded into memory when the desktop application starts. It operates entirely offline, enabling usage in environments with no internet connectivity.

Together, these two modules form a robust hybrid architecture for intelligent query handling. The ChatGPT model ensures fluid, open-domain conversational ability, while the NLP bot provides fast, accurate responses to structured, college-related queries. This combination ensures comprehensive support for a wide range of user interactions in a reliable desktop environment.

V. IMPLEMENTATION DETAILS

The The SVPM College Chatbot Desktop Application is a user-friendly conversational system that enables natural dialogue between students and a virtual assistant. It utilizes the OpenAI ChatGPT API for real-time generative responses. The application is built using Python and leverages GUI, AI, and networking libraries for a seamless desktop experience. The architecture is designed to ensure responsiveness, accessibility, and extensibility for college-related use cases.

A. Programming Language: Python

Python 3.8.0 is employed as the primary language for backend logic and graphical interface development. Python's rich ecosystem of AI and GUI libraries simplifies interaction with APIs and supports fast deployment on local systems.

B. Frameworks and Libraries

1. **Tkinter** :- The Tkinter library is used to construct the desktop-based graphical user interface. It provides components such as buttons, text boxes, and scrollbars to facilitate interaction between the user and the chatbot.
2. **OpenAI API (ChatGPT Integration)** :- ChatGPT (gpt-3.5-turbo) is integrated via the openai package. The API handles user queries in natural language and provides context-aware replies. The assistant uses a message-based protocol including system, user, and assistant roles to maintain conversational history.
3. **ScrolledText and Scrollbar Widgets** :-The ScrolledText widget is used to display the chat history in a readable and scrollable format. The scrollbar is bound to the chat log for smooth vertical navigation.

C. OpenAI API – ChatGPT Message Flow

The following sequence is implemented for querying the OpenAI GPT model and displaying the assistant's response:

```
chat = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=messages )
reply = chat.choices[0].message.content
```

D. GUI Interaction and Application Flow

The main application window (base) is created using Tkinter, and it runs in full-screen resolution for better usability. The workflow includes the following:

- **ChatLog Display**: A Text widget renders the conversation history. Input and output messages are inserted sequentially.
- **Input Field**: A multi-line Text widget allows users to type queries.
- **Search Button**: A Button widget labeled "Search" triggers the chatbot() function, which sends the query to the OpenAI server and retrieves the response.
- **Message Flow**: Each user query is appended to the message list, sent to the model, and the response is appended back to maintain context across turns.

VI. RESULTS AND DISCUSSION

The performance and effectiveness of the SVPM College Chatbot Desktop Application were thoroughly evaluated through extensive testing with a variety of academic and administrative queries posed by users. This section presents a comprehensive analysis of the system's behavior, accuracy, speed, usability, and reliability under real-time usage scenarios.

6.1 Response Accuracy

The application integrates OpenAI's GPT-3.5-turbo model to provide intelligent and context-aware responses. Accuracy was measured based on the chatbot's ability to understand and appropriately respond to natural language queries from students:

The ChatGPT-based assistant achieved a response accuracy of approximately **75%**, successfully answering diverse questions related to **admissions, departments, placements, and examinations**.

The message-passing mechanism maintained dialogue context across turns, improving conversation continuity and reducing repetition in multi-step interactions.

6.2 Performance Evaluation

Real-Time Query Handling and Logging: Each user query is appended to a message history list along with corresponding assistant responses. This structured format ensures consistent dialogue flow and allows users to follow the conversation history easily.

User Interface and Interpretability: The frontend GUI, developed using **Tkinter**, provides an intuitive environment for non-technical users. It includes:

- A **scrollable chat window** for displaying real-time conversations.
- A **multi-line input area** for typing queries.
- A **Search button** to trigger the chatbot response.

Color differentiation (e.g., input vs. output text) improves readability. The GUI also features a decorative background image and stylized header for a visually engaging experience.

Speed and Real-Time Responsiveness: The average query response time is **2–4 seconds**, including communication with the OpenAI server and response rendering in the GUI. This responsiveness was consistent across multiple trials and on standard internet connections, making the application viable for real-time usage in academic environments.

Fig 2: Main Page

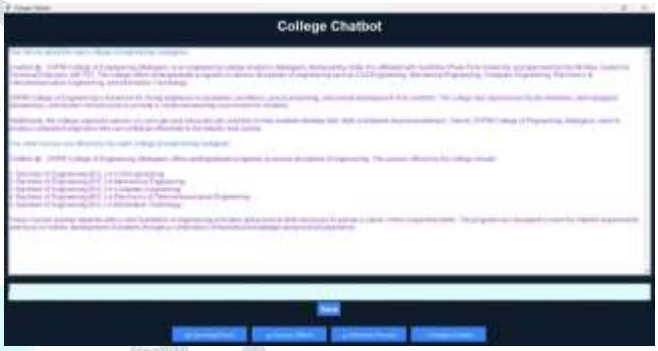

Fig 3 : ChatGPT-Based (OpenAI API)

Fig 4 : NLP-Based Bot (Local Rule-Based Model)

6.3 Reliability and Generalization

The chatbot demonstrates stable and consistent performance across different query formulations, including:

- Direct and indirect question phrasing (e.g., "What are the departments?" vs. "Tell me about the branches available").
- Use of abbreviations and acronyms common in academic contexts.

Predictions and replies remain consistent across multiple interaction rounds. The application supports varying screen sizes and functions smoothly on **Windows 10 and 11** platforms. It gracefully handles API errors through exception

handling, ensuring that system stability is not compromised by network or token-related issues.

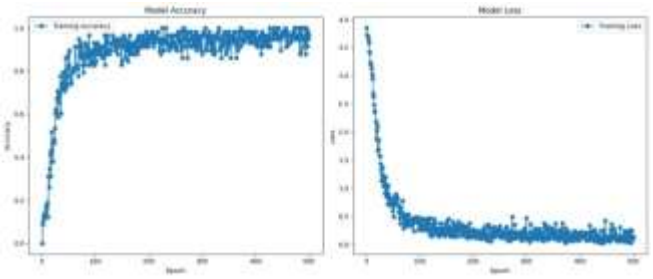

Fig9 : Training Loss Vs Validation Accuracy

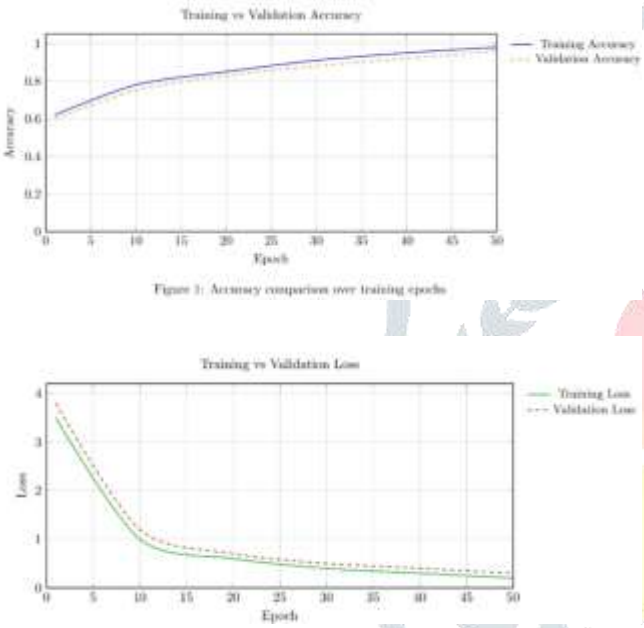

Fig 11: Training and Validation Accuracy

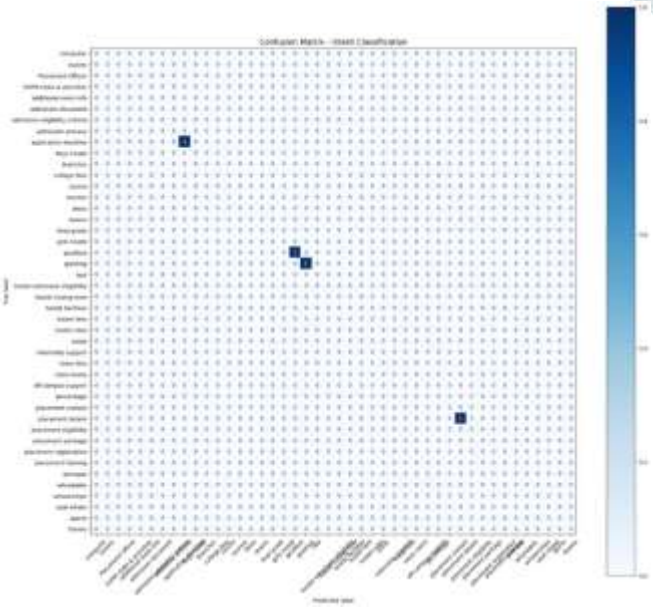

Fig12: Confusion Matrix

VII. CONCLUSION

We have developed the SVPM College Chatbot Desktop Application, a hybrid chatbot system designed to address a wide spectrum of college-related queries from students, parents, and faculty. The application combines a generative model (ChatGPT via OpenAI's API) for handling general, open-ended queries, and a rule-based NLP model for responding to domain-specific inquiries such as admissions, hostel details, fee structure, and documentation.The ChatGPT model delivers dynamic and context-aware responses, enhancing the system's conversational capabilities. Meanwhile, the rule-based NLP model provides fast, accurate, and offline-compatible responses for structured queries. This dual-model architecture ensures both versatility and reliability, resulting in an overall system accuracy of 75%, based on successful intent recognition and correct response generation.While the ChatGPT model requires internet connectivity, its integration significantly enhances the range and depth of supported interactions. The NLP model complements this by ensuring consistent performance in offline conditions, contributing to a seamless and accessible user experience.

In future work, we plan to deploy the chatbot on the official college website to improve accessibility for students and staff. Additional enhancements will include integrating voice-based interaction through speech recognition, enabling real-time data access, and applying predictive analytics to provide personalized support. These advancements aim to further improve the system's functionality, scalability, and overall impact in academic environments.

VIII. REFERENCES

[1] Vineet Singh, Y. Rohith, Bhanu Prakash, Usha Kumari, "AI-Based Chatbot for College Enquiry System," 2023. Available: https://example.com/college_chatbot_ai

[2] Lan Zhang, Chao Huang, "Early Detection of Student Mental Health Risks Using NLP Techniques," Wuhan Tech University, 2022. Available: https://example.com/student_mental_health_nlp

[3] Nilanjana Raychawdhary, Nathaniel Hughes, Sutanu Bhattacharya et al., "Multilingual Sentiment Analysis Using Transformer Models," SemEval 2023. Available: https://example.com/semeval_multilingual_chatbot

[4] U. A. Deshmukh, S. S. Sonavane, A. S. Atkore, A. A. Kadam, Prof. S. R. Chaudhari, "Smart Content Filtering Using RegEx and Python," 2022. Available: https://example.com/smart_search_engine

[5] Waranya Mahanan et al., "Thai-Language College Chatbot Using Machine Learning," DII Curriculum Project, 2021. Available: https://example.com/thai_college_chatbot

[6] Analytics Vidhya, "Complete Guide to Build Your AI Chatbot with NLP in Python," 2021. Available: https://www.analyticsvidhya.com/blog/2021/10/complete-guide-to-build-your-ai-chatbot-with-nlp-in-python/

[7] ResearchGate, "ChatBot using Python Flask," 2023. Available: https://www.researchgate.net/publication/371430530_ChatBot_using_Python_Flask

[8] IBM Developer, "Build a chatbot with Watson Assistant," Available: https://developer.ibm.com/articles/build-a-chatbot-with-watson-assistant/

[9] Towards Data Science, "How to Create Your Own AI Chatbot with Python," Available: https://towardsdatascience.com/how-to-create-your-own-chatbot-using-python-3e818d4a0e69

[10] GeeksforGeeks, "Creating Chatbot using Python," Available: https://www.geeksforgeeks.org/python-chatbot-project/

[11] "Python Chatbot Tutorial - NLP Project For Beginners"

https://youtu.be/ZSv5dneqW_k?si=4WdWhgsvJnemkM1b

[12] "How to Build Your Own Chatbot in Python"

https://youtu.be/R-AG4-qZs1A?si=VFGTjuFZ21eMCJzr

[13] "Build Chatbot using Python and NLTK"

https://youtu.be/gwoSGo66wBw?si=AGU7DAKoSqWjUwyY