IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Chatbot For College Enquiries Using Ai-Powered Information Assistance

D. Sreenivasulu, HOD, CSE Department, KLMCEW, Kadapa, Andhra Pradesh, India.

Syed Nafeesa Thehseen, Assistant Professor, *CSE Department*, KLMCEW, Kadapa, Andhra Pradesh, India

K. Umamaheshwari Devi, Assistant Professor, *CSE Department*, KLMCEW, Kadapa, Andhra Pradesh, India.

ABSTRACT-Chatbots are key tools for intelligent communication, offering real-time, automated responses across various domains. This paper introduces a Python-based, voice-enabled chatbot tailored for academic environments. It supports both speech and text input/output, using natural language processing (NLP), speech recognition, and text-to-speech (TTS) to handle user queries effectively. The system utilizes a structured JSON knowledge base to provide quick, context-aware responses on topics such as admissions, courses, and facilities. Built with libraries such as speech recognition, pyttsx3, and json, it ensures smooth, intuitive interaction without complex navigation. Notably, the chatbot operates offline, making it suitable for areas with low connectivity. Its modular design also supports future enhancements, including sentiment analysis, multilingual features, and cloud integration. This paper highlights how AI-driven voice interfaces can enhance human-computer interaction, streamline academic support, and improve access to institutional information.

KEY WORDS Natural Language Processing (NLP), Machine Learning (ML), emotional recognition, ELIZA, Neural Conversational Model, GPT-3, voice-enabled chatbot, Levenstein Distance algorithm, lemmatization, fuzzy matcher.

I. INTRODUCTION

In today's digital age, Artificial Intelligence (AI)—especially its subfields like Natural Language Processing (NLP) and Machine Learning (ML)—has transformed how humans interact with machines. One of the most practical outcomes of this transformation is the chatbot, which enables users to have real-time, conversational interactions with systems. This project presents a Python-based voice-enabled chatbot tailored for college enquiries, specifically designed to provide information to students, parents, and staff in an academic environment.

The chatbot accepts text and voice input, using the speech recognition library to convert speech to text and pyttsx3 for generating spoken responses. It uses a JSON-based knowledge base to map questions to answers, enabling fast and accurate responses on topics such as admissions, departments, facilities, courses, hostel info, and placements.

A key feature of this chatbot is its offline functionality, allowing it to operate without internet access—ideal for rural or limited-connectivity regions. Its modular architecture supports easy updates and future expansions, including multilingual support, emotional recognition, and deep learning-based NLP for advanced query understanding.

The chatbot was developed using a structured software engineering process, starting from system analysis and requirement gathering, through system design and module development, to thorough testing and validation. Emphasis was placed on usability, accessibility, and efficiency, ensuring that even non-technical users can interact naturally with the system.

This project showcases how AI can streamline institutional support by reducing the administrative burden, offering 24/7 automated assistance, and improving information access. It highlights the potential of voice-based AI systems to redefine digital engagement in the education sector and provides a foundation for future AI-powered academic tools.

II. LITERATURE SURVEY

The development of chatbots has rapidly progressed, moving from early **rule-based systems** to sophisticated **AI-driven conversational agents** using NLP and deep learning. This survey highlights key milestones and influential research that shaped modern chatbot technology:

- ✓ **ELIZA (1966, Weizenbaum)**: A pioneering rule-based chatbot using pattern matching. It laid the foundation for machine-human language interaction.
- ✓ **ALICE (2003, Wallace)**: Introduced AIML for scalable, template-based conversations, improving on ELIZA's limitations.
- ✓ **Neural Conversational Model (2015, Vinyals & Le)**: Shifted chatbot design towards sequence-to-sequence models using RNNs, enabling more natural, unscripted conversations.
- ✓ **XiaoIce (2020, Zhou et al.)**: Demonstrated emotionally intelligent chatbot design, focusing on empathy, personalization, and multi-turn dialogues.
- ✓ **GPT-3 (2020, OpenAI)**: Marked a leap in NLP with few-shot learning and large-scale transformer models capable of human-like, context-aware conversation.
- ✓ **Speech and Language Processing (2022, Jurafsky & Martin)**: A key academic resource explaining NLP fundamentals, intent recognition, and dialogue systems.
- ✓ **Next-Gen Voice Assistants (2018, Kepuska & Bohouta)**: Analyzed Siri, Alexa, Cortana, and others, emphasizing the integration of NLP, ASR, and TTS in real-time assistants.

The chatbot system in this project incorporates insights from these foundational works. Built in **Python**, it uses libraries like speech recognition, pyttsx3, and json to support **voice and text interactions**. While currently rule-based, its **modular design** prepares it for future upgrades using **deep learning and transformer models**, aligning it with modern AI chatbot trends. The system is designed to be **scalable, offline-capable**, and accessible for educational institutions.

III. EXISTING SYSTEM:

Traditional information systems in colleges rely on manual support, static FAQs, or basic web forms. These systems lack interactivity, real-time conversation, and personalization. They are difficult for non-technical users, don't support voice interaction, and depend on internet connectivity, making them less accessible in areas with low network connectivity.

Limitations of Existing Systems:

- ➢ No support for voice input/output
- ➢ Manual navigation required
- ➢ Lack of real-time or follow-up interaction
- ➢ Static, keyword-based responses
- ➢ Poor accessibility for visually impaired or non-digital users
- ➢ Dependence on the internet

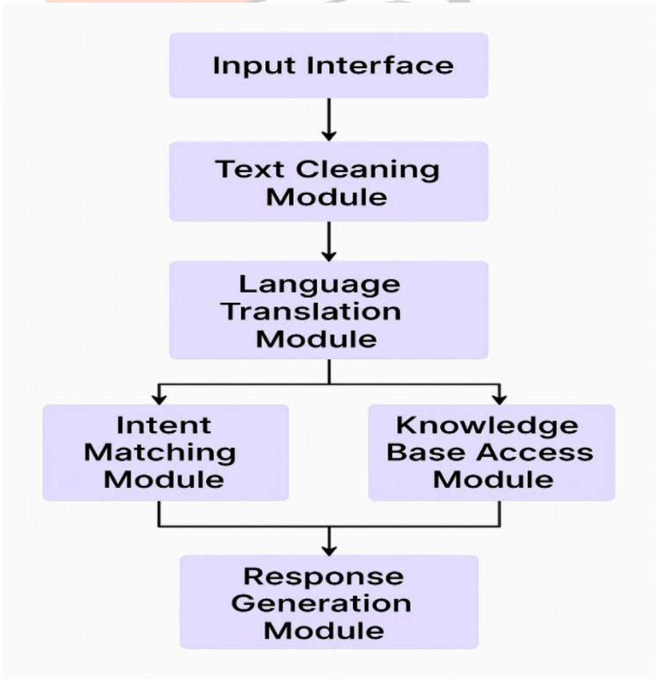
IV. PROPOSED SYSTEM:

The paper introduces an offline, voice-enabled chatbot using Python. It accepts spoken or typed questions, matches them to known intents from a JSON knowledge base, and replies via voice and text. It is customizable, modular, and designed for academic institutions.

Advantages:

- Voice input and spoken output
- Full offline functionality
- Lightweight and easy to scale or customize
- 24/7 availability for students, parents, and staff
- Reduces staff workload
- Can be integrated with websites, dashboards, or kiosks

Fig: Flow Chart for Proposed System


Input Interface
Text Cleaning Module
Language Translation Module
Intent Matching Module
Knowledge Base Access Module
Response Generation Module

V. SYSTEM ARCHITECHTURE

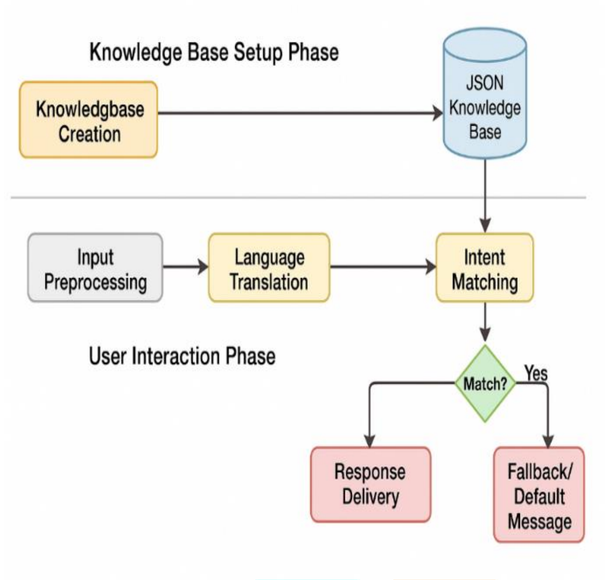


Fig: System Architecture

V. INPUT DESIGN:

The **input subsystem** is the first point of interaction for users and plays a key role in the system's usability and performance.

- **Supported Input Modes**:
  - **Text-based input** through a web interface (HTML5 form).
  - **Voice input** is considered for future versions using speech recognition.
  - **Multilingual input** is accepted and translated using deep translator.
- **Input Processing Workflow**:
  1. **Whitespace normalization**
  2. **Lowercasing** of all characters
  3. **Punctuation and special character removal**
  4. **Stopword elimination**
  5. **Tokenization** for keyword extraction

These steps ensure that the user’s query is clean, minimal, and suitable for intent recognition.

- **Design Features**:
  - Simple single-line input box
  - Submit via button or Enter key
  - User alerts for empty queries
  - Designed for **clarity, simplicity, and future extensibility**

VI. OUTPUT:

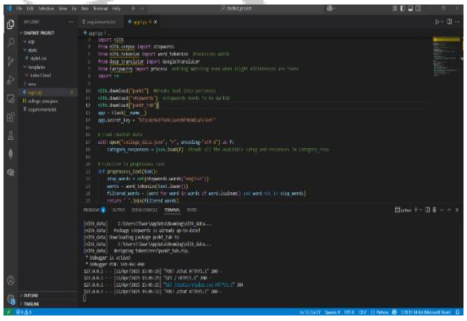
The **output design** ensures that system responses are clear, accurate, and appropriately delivered.

- **Types of Output**:
  - **Text output** on the web interface
  - **Spoken responses** using pyttsx3 (offline TTS engine)
  - **Multilingual response** using real-time translation
  - **Fallback messages** for unrecognized queries
- **UI and Accessibility**:
  - Conversational style message stacking
  - AJAX or Flask-powered dynamic message loading
  - High contrast, readable fonts for accessibility
  - Optional voice output for visually

Figure 1: Backend Console in Visual Studio Code
Displays the execution of app1.py, which runs the **Flask backend**, handles user requests, and processes input with **NLTK**.
The terminal logs confirm:

- Successful **POST requests**
- Active **route handling**
- Proper communication between the **client and server**

This screenshot verifies that the **backend is fully operational** and correctly integrated with the chatbot interface.



```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize  #tokenizes words
from deep_translator import GoogleTranslator
from fuzzywuzzy import process  #string matching even when slight differences are there
import re

nltk.download("punkt")  #breaks text into sentences
nltk.download("stopwords")  #stopwords needs to be [illegible]
nltk.download("punkt_tab")
app = Flask(__name__)
app.secret_key = "[illegible]"

# Load chatbot data
with open("college_data.json", "r", encoding="utf-8") as f:
    category_responses = json.load(f)  #loads all the available categ and responses in category_resp

# Function to preprocess text
def preprocess_text(text):
    stop_words = set(stopwords.words("english"))
    words = word_tokenize(text.lower())
    filtered_words = [word for word in words if word.isalnum() and word not in stop_words]
    return " ".join(filtered_words)
```

```
[nltk_data]     C:\Users\[illegible]\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\[illegible]\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
 * Debugger is active!
 * Debugger PIN: [illegible]
127.0.0.1 - - [12/Apr/2025 15:05:19] "POST /chat HTTP/1.1" 200 -
127.0.0.1 - - [12/Apr/2025 15:05:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Apr/2025 15:05:25] "GET /static/style1.css HTTP/1.1" 304 -
127.0.0.1 - - [12/Apr/2025 15:05:32] "POST /chat HTTP/1.1" 200 -
```

Code Execution

Fig: Initial Chatbot Interface

The image below shows the homepage of the chatbot when the server is initially launched via Flask at 127.0.0.1:5000. The interface is minimal and user-friendly, designed with accessibility in mind. Users are presented with a chat window, a language selection dropdown, a text input field, and buttons to either submit text or activate voice input. The clean layout allows users to begin interaction immediately without unnecessary distractions.

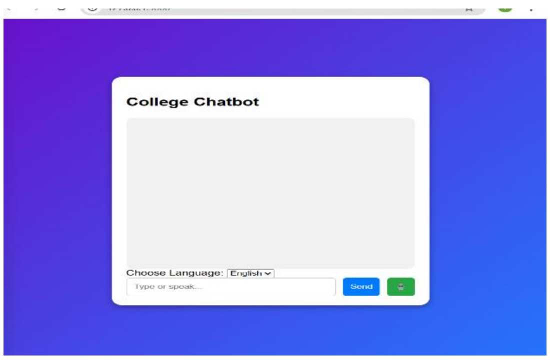
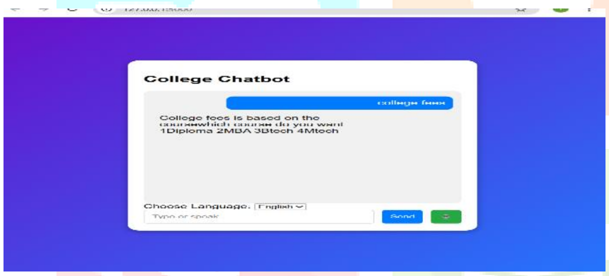


Fig Chatbot Interface

Chat Interaction with a Query on College Fees

This screenshot captures a live interaction where the user enters the phrase "college fees". The chatbot successfully identifies the intent and responds with a clarification, asking the user to specify the course. The response is rendered dynamically in the chat window, indicating successful backend processing, intent matching, and content retrieval from the knowledge base. This interaction demonstrates the primary functional behavior of the chatbot—accepting a natural language input and returning a context-aware response.



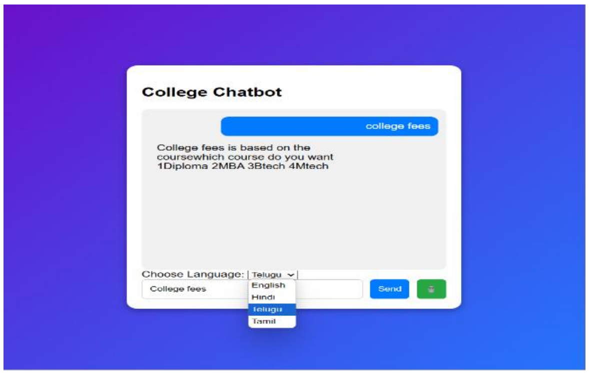
college fees

College fees is based on the coursewhich course do you want 1Diploma 2MBA 3Btech 4Mtech

Fig: Chatbot Interaction

Language Selection Functionality

This image highlights the chatbot's multilingual capability. The user is shown selecting Telugu from the language dropdown. This feature enables users to communicate in their native language. Internally, the input is translated into English for processing and the response is translated back into the selected language before being displayed. This two-way translation ensures broader accessibility and demonstrates the system's adaptability to regional contexts.



college fees

College fees is based on the coursewhich course do you want 1Diploma 2MBA 3Btech 4Mtech

College fees

Fig: Chatbot Language Selection

These output screenshots provide visual confirmation of the chatbot's operational flow, from user input to backend processing and the generation of dynamic responses. Each image supports the claim that the chatbot system is functionally complete, user-oriented, and capable of multilingual interaction across diverse contexts. The interface is responsive, the system logic is sound, and the backend processes user input with high accuracy and consistency.

VII. Conclusion

The Multilingual Intent-Based Chatbot successfully integrates **language translation**, **fuzzy logic intent recognition**, and **structured response generation** within a modular, user-friendly framework. Key strengths include:

- **Multilingual support** for inclusive communication
- **Robust intent matching** tolerant of phrasing and spelling errors
- **Simple UI** with voice/text interaction
- **Lightweight backend** built on Flask
- **Thorough testing** to ensure reliability

The project meets its goals and offers a scalable foundation for advanced features like **AI-based intent detection, cloud deployment**, and **context-aware interaction**. With continued refinement, this chatbot can evolve into a full-featured, intelligent virtual assistant for domains like education, customer service, and public support.

VIII. FUTURE ENHANCEMENTS

The current chatbot system provides solid functionality with multilingual support, fuzzy intent matching, and a static knowledge base. However, several future upgrades are proposed to improve performance, scalability, and user experience:

- **Advanced Intent Detection**: Replace fuzzy matching with deep learning models like BERT or RoBERTa for better semantic understanding and accuracy.
- **Dynamic Knowledge Base**: Migrate from static JSON to a real-time database (SQL/NoSQL) for easier updates and potential self-learning features.
- **Text-to-Speech (TTS)**: Enable voice-based responses for improved accessibility, especially for visually impaired users.
- **Enhanced UI**: Add features like typing animations, conversation history, responsive design, and avatar-based interactions to boost engagement.
- **Sentiment Analysis**: Incorporate emotion recognition to adjust tone and improve interaction in sensitive or personal contexts.

- **Admin Dashboard**: Build a web-based control panel for non-technical administrators to manage intents, track activity, and view analytics.
- **Cloud Deployment**: Migrate to platforms like AWS or Google Cloud for scalability, better performance, and multi-department access.
- **Multi-turn Conversations**: Introduce context retention to allow continuous, natural dialogue and follow-up queries.

IX. REFERENCE

1. Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, *"DeepFace: Closing the Gap to Human-Level Performance in Face Verification,"* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
2. Joy Buolamwini and Timnit Gebru, *"Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification,"* Proceedings of the 1st Conference on Fairness, Accountability, and Transparency, 2018.
3. Google Cloud, *"Cloud Translation API Documentation."* [Online]. Available: https://cloud.google.com/translate
4. Matthew Honnibal and Ines Montani, *"spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,"* To appear, 2017.
5. Jason Brownlee, *"Introduction to Text Cleaning with Python,"* Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/text-cleaning-for-machine-learning/
6. NLTK Project, *"Natural Language Toolkit Documentation,"* [Online]. Available: https://www.nltk.org/
7. Python Software Foundation, *"Python Language Reference, version 3.x,"* [Online]. Available: https://www.python.org/
8. Flask Documentation, *"Flask Web Framework,"* [Online]. Available: https://flask.palletsprojects.com/
9. RapidFuzz Library, *"RapidFuzz: Rapid String Matching in Python,"* [Online]. Available: https://maxbachmann.github.io/RapidFuzz/
10. Deep Translator Documentation, *"deep_translator — Python translation library,"* [Online]. Available: https://pypi.org/project/deep-translator/
11. Tesseract OCR, *"Tesseract Open Source OCR Engine,"* [Online]. Available: https://github.com/tesseract-ocr/
12. K. Kowsalya, *"Artificial Intelligence Chatbot System Using Natural Language Processing,"* International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 11, pp. 3204–3208, 2019.