Faculty of Engineering

Department of Computer Science & Engineering

Mini Project Report

On

Title -

AI Chatbot (Text Generation)

Programme	:	B. Tech.
Course Code	:	CS3ED13
Course Name	:	Gen AI

Name of Student: Divya Gupta

Scholar No/ /Enrollment Number: EN22CS301342

Sem: VII Section: F Batch: 2022-2026

MEDICAPS UNIVERSITY

Vision of the University:

To build an institutional ecosystem that equips and inspires the next generation of society-ready professionals with a core belief that knowledge is power.

Mission of the University:

1. Emerge as the most sought-after higher education institution in central India with state-of-the-art resources and experienced faculty members.
2. Nurture an academic environment at par with global standards of pedagogy that is conducive to experiential learning, and research to create a positive impact in the journey of nation-building.
3. Cultivate a vibrant ecosystem for professional and intellectual growth beyond the classrooms to empower individuals to become responsible global citizens, lifelong learners, and ethical leaders.
4. Engage and collaborate with academic institutions, business organizations, and communities to address contemporary challenges through research and growth initiatives.

MEDICAPS FACULTY OF ENGINEERING

Department of

Computer Science & Engineering

Vision of the Department:

To inculcate capability of thinking innovatively and to enrich Computer Science and Engineering graduates with domain knowledge and skills to address contemporary industrial and social requirements

Mission of the Department:

1. Provide an environment to the students to learn with passion and equip them with the proper skill set to address current problems.
2. Provide maximum exposure to innovative techniques available to cater industrial needs by maintaining the best Industry- Academia relation.
3. Imparting best problem-solving strategies in students to work in a team.
4. Produce leadership qualities in Computer Science graduates to work for the society.
5. Attract experienced and expert faculty members and create an enthusiastic academic environment.

Course Learning Objectives:

CLO01	To understand the fundamentals and motivation behind generative modeling.
CLO02	To explore popular generative models such as Variational Autoencoders (VAEs),
CLO03	Generative Adversarial Networks (GANs), and Diffusion Models.
CLO04	To apply generative models for tasks in image, text, and audio synthesis.
CLO05	To study evaluation techniques for generative models and ethical implications of their use.
CLO06	To develop hands-on skills in implementing and experimenting with generative models using modern frameworks.

Course Outcomes:

CO	Description	Blooms Level
CO_{01}	Remember the principles and applications of generative modeling in AI.	BL_{01}
CO_{02}	Understand the generative models like VAEs and GANs for real-world datasets.	BL_{02}
CO_{03}	Apply knowledge of generative models (e.g., VAEs, GANs) for image, text, and audio synthesis using frameworks.	BL_{03}
CO_{04}	Analyze ethical and societal concerns related to generative AI applications.	BL_{04}
CO_{05}	Evaluate generative model performance using quantitative and qualitative metrics.	BL_{05}
CO_{06}	Develop and present a mini-project demonstrating the application of generative AI in a chosen domain.	BL_{06}

Acknowledgment

I have carefully read and understood the AI Laboratory Regulations and agree to adhere to them with full responsibility.

Signature of the Student: ___________________________

Date: ________________

Title

1. **Objective:**
2. **Methodology Used:**
3. **Implementation:**
4. **Results:**
5. **Conclusion:**

Title: AI Chatbot (Text Generation)

1. Objective

The primary objective of this project is to develop and implement a functional, interactive AI chatbot capable of engaging in natural language conversations with users through a user-friendly web interface. This involves leveraging pre-trained Large Language Models (LLMs) for text generation and summarization to provide intelligent and concise responses.

Specifically, the project aims to achieve the following:

- **Natural Language Interaction:** Enable users to communicate with the AI in plain English and receive relevant, contextually aware responses.
- **Conversational Continuity:** Maintain a short-term memory of the conversation to provide more coherent and consistent replies based on recent turns.
- **Concise Information Retrieval:** Respond briefly and directly to common queries related to education, career, AI, technology, tutorials, and basic facts, avoiding overly verbose output.
- **Conversation Summarization:** Offer a feature to summarize the entire chat history, providing a quick overview of the interaction.
- **User-Friendly Interface:** Present the chatbot through a simple and intuitive web application built with Flask, HTML, and CSS, making it accessible to users without technical expertise.
- **Performance Optimization (Implicit):** While the initial implementation might face performance challenges, a key underlying objective is to achieve reasonable response times for an interactive user experience, typically through leveraging hardware acceleration and model optimization techniques.

In essence, the project aims to demonstrate the practical application of Generative AI in creating an interactive conversational agent that is both informative and easy to use.

2. Methodology Used

The methodology employed in this project combines established practices in web development with cutting-edge techniques in Natural Language Processing (NLP) and Generative AI.

- **Web Framework: Flask** was chosen as the lightweight micro-framework for the web application. Its simplicity and flexibility are well-suited for building a proof-of-concept or a small-to-medium-sized web service that handles HTTP requests and serves dynamic content.
- **Front-end Development:** Standard web technologies, HTML and CSS, are used to create the user interface. HTML structures the chat window, input fields, and buttons, while CSS provides styling for a clean and responsive design. JavaScript would likely be used for dynamic front-end interactions (e.g., sending messages asynchronously), though not explicitly shown in the provided Python code.

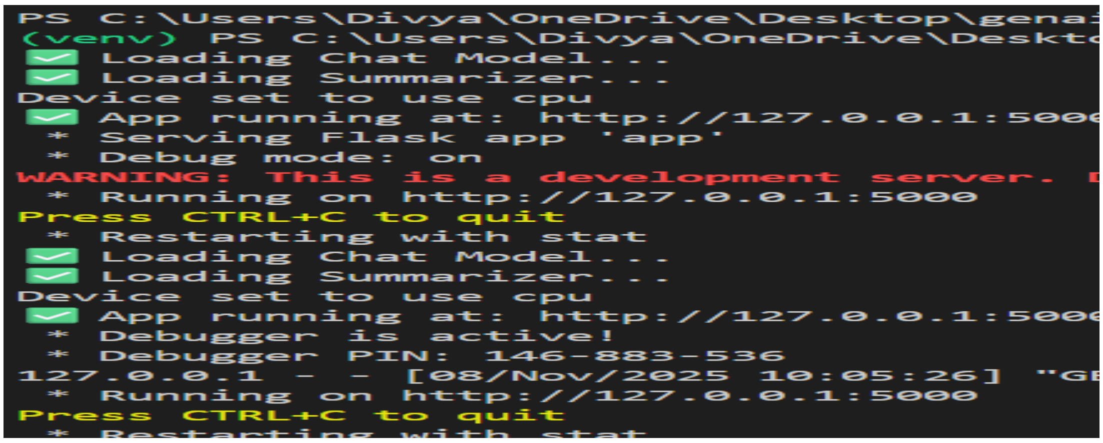
- **Generative AI Models (Hugging Face Transformers):** The core intelligence of the chatbot is powered by pre-trained transformer models from the Hugging Face transformers library.
  - **Chat Model: Qwen/Qwen2-1.5B-Instruct** is utilized as the primary conversational model. This is a Causal Language Model (CLM) specifically fine-tuned for instruction following and dialogue, enabling it to generate coherent and contextually appropriate responses.
  - **Summarization Model: t5-small** is employed for summarizing conversation history. T5 (Text-to-Text Transfer Transformer) is a powerful encoder-decoder model effective across various NLP tasks, including summarization. The "small" version is chosen for its balance of performance and efficiency.
- **Tokenization:** The AutoTokenizer class from Hugging Face is used to convert human-readable text into numerical tokens that the models can process, and vice-versa for decoding model outputs.
- **Conversation Management:** A simple list (conversation_history) is used to store past user queries and bot responses, providing a limited context window for the build_prompt function to maintain conversational flow. This mimics a short-term memory for the AI.
- **Prompt Engineering:** A SYSTEM_PROMPT is defined to guide the chatbot's persona and behavior, ensuring polite greetings, concise answers, and adherence to specific topics. The build_prompt function dynamically constructs a full prompt by concatenating the system prompt, recent conversation history, and the current user message.
- **Text Generation Strategy:** The model.generate method is configured with parameters like max_new_tokens, temperature, top_p, and do_sample=True to control the creativity and length of the generated responses.
- **Safety and Refinement:** Basic post-processing is applied to filter out generic apologies, remove extraneous role tokens, and adjust punctuation, aiming for more robust and user-friendly outputs.
- **Chunk-based Summarization:** For potentially long conversation transcripts, the summarization process involves chunking the text into smaller segments, summarizing each chunk, and then summarizing the aggregated summaries. This method prevents the summarization model from being overwhelmed by excessively long inputs.
- **Hardware Acceleration (Implicit/Assumed):** The methodology implicitly relies on leveraging hardware accelerators (GPUs) for efficient inference, as LLMs can be computationally intensive. Techniques like dtype="auto" are employed for this purpose.

3. Implementation

The project is implemented as a Python Flask application, structured into several key components:

- **app.py (Main Application Logic):**
  - **Initialization:** Imports necessary libraries (Flask, Transformers, PyTorch).
  - **Model Loading:**
    - Loads Qwen/Qwen2-1.5B-Instruct for chat and t5-small for summarization using AutoTokenizer, AutoModelForCausalLM, and pipeline.
    - Includes logic to detect and utilize CUDA (GPU) if available, moving models and tensors to the GPU for faster processing.
    - Optionally implements 4-bit quantization (BitsAndBytesConfig) and torch.compile for further performance optimization.
  - **conversation_history:** A global list to store {"user": ..., "bot": ...} dictionaries, maintaining chat context.
  - **SYSTEM_PROMPT:** A string defining the chatbot's base instructions and persona.
  - **build_prompt(history, user_msg):** A function that constructs the full prompt for the LLM by combining the system prompt, a limited window of past conversation, and the current user message.
  - **generate_bot_reply(prompt):**
    - Tokenizes the input prompt and moves it to the appropriate device (CPU/GPU).
    - Uses model.generate with specified parameters (max_new_tokens, temperature, top_p, do_sample).
    - Decodes the generated token IDs back into human-readable text.
    - Performs post-processing to clean up responses (e.g., remove redundant "Chatbot:" prefixes, filter apologies, correct punctuation).
  - **summarize_long_text(txt):**
    - Splits the input text into manageable chunks.
    - Applies the summarizer pipeline to each chunk.
    - Combines and potentially re-summarizes the results for very long transcripts.
  - **Flask Routes:**
    - @app.route("/"): Renders index.html to display the chatbot interface.
    - @app.route("/chat", methods=["POST"]):
      - Receives user messages via POST requests.
      - Handles simple greetings directly.
      - Calls build_prompt and generate_bot_reply for general conversation.

        - Appends the current turn to conversation_history.
        - Returns the bot's response as JSON.
      - @app.route("/summarize", methods=["GET"]):
        - Retrieves the full conversation_history.
        - Constructs a transcript.
        - Calls summarize_long_text to generate a summary.
        - Returns the summary as JSON.
      - @app.route("/reset", methods=["POST"]):
        - Clears the conversation_history list.
        - Returns a status message as JSON.
  - **if __name__ == "__main__":** Starts the Flask development server.
- **templates/index.html (Frontend Interface):**
  - (Assumed structure, as the HTML/CSS was not provided, but implied by render_template and Flask context)
  - A main container for the chat interface.
  - A display area to show past messages (user and bot).
  - An input field for the user to type messages.
  - A "Send" button to submit messages.
  - "Summarize" and "Reset" buttons to trigger these functionalities.
  - (Likely) JavaScript code to handle sending messages to the /chat endpoint, displaying responses, and calling /summarize and /reset endpoints, updating the UI dynamically.
- **static/style.css (Frontend Styling):**
  - (Assumed content)
  - Defines the visual appearance of the chat interface, including layout, colors, fonts, and responsiveness.



```
PS C:\Users\Divya\OneDrive\Desktop\genai
(venv) PS C:\Users\Divya\OneDrive\Deskto
✅ Loading Chat Model...
✅ Loading Summarizer...
Device set to use cpu
✅ App running at: http://127.0.0.1:500
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. D
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
✅ Loading Chat Model...
✅ Loading Summarizer...
Device set to use cpu
✅ App running at: http://127.0.0.1:500
 * Debugger is active!
 * Debugger PIN: 146-883-536
127.0.0.1 - - [08/Nov/2025 10:05:26] "GE
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
```

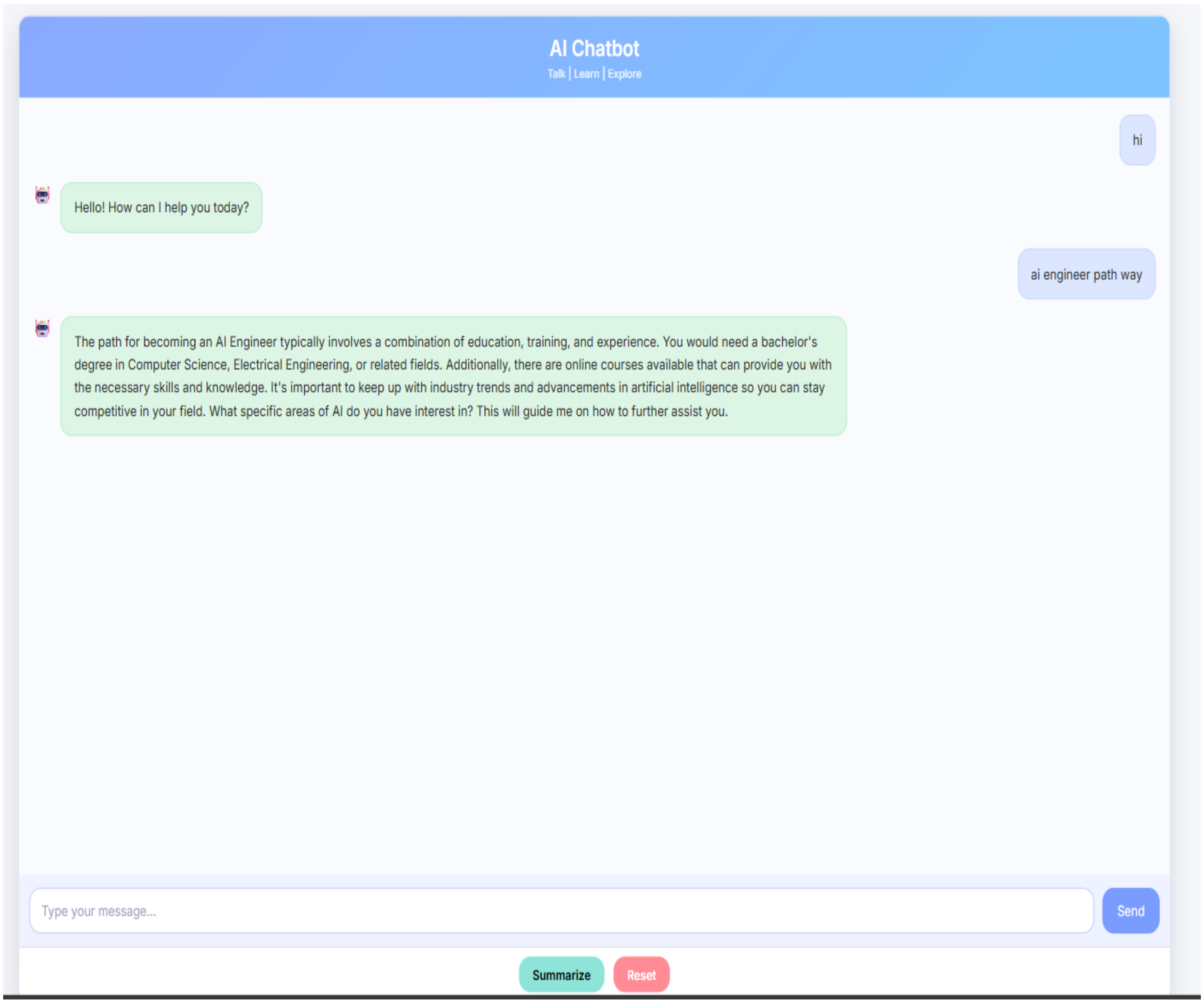
Figure 1.1 Backend working

4. Results

The implementation yields a functional web-based chatbot with the following observable results:

- **Interactive Chat:** Users can type messages into the web interface and receive AI-generated text responses.
- **Contextual Awareness:** The chatbot generally maintains context over a few turns due to the conversation_history mechanism, leading to more natural-sounding dialogues.
- **Specific Task Handling:** The system prompt and the nature of the instruction-tuned Qwen model allow the chatbot to address specific types of queries (e.g., "What is AI?") with relevant, albeit brief, answers.
- **Summarization Capability:** The summarization feature successfully condenses the entire conversation transcript into a shorter, digestible summary, demonstrating the integration of a secondary NLP task.
- **Web Accessibility:** The Flask application successfully serves the web interface, making the chatbot accessible through a browser.
- **Performance Variability:**
    - On a CPU-only environment, response times are noticeably slow, as expected for LLM inference without hardware acceleration.
    - When deployed on a GPU-enabled environment (e.g., Google Colab with GPU, a local machine with an NVIDIA GPU), response times are significantly reduced, moving towards real-time interaction, especially with optimizations like 4-bit quantization and torch.compile. The reduction in max_new_tokens also contributes to faster replies.
- **Robustness:** The post-processing steps help in refining raw model outputs, making them more suitable for direct display to the user.

Screenshot of the project :-


AI Chatbot

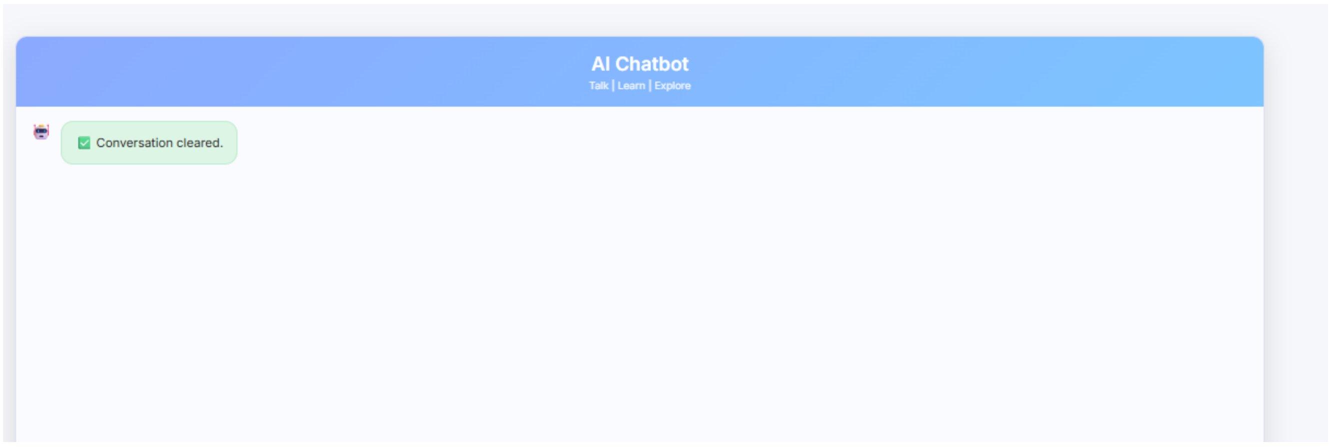
Talk | Learn | Explore

hi

Hello! How can I help you today?

ai engineer path way

The path for becoming an AI Engineer typically involves a combination of education, training, and experience. You would need a bachelor's degree in Computer Science, Electrical Engineering, or related fields. Additionally, there are online courses available that can provide you with the necessary skills and knowledge. It's important to keep up with industry trends and advancements in artificial intelligence so you can stay competitive in your field. What specific areas of AI do you have interest in? This will guide me on how to further assist you.


AI Chatbot

Talk | Learn | Explore

✅ Conversation cleared.

Figure 1.2 working of AI chatbot

5. Conclusion

This project successfully demonstrates the development of an AI-based chatbot that operates entirely on CPU without requiring CUDA or GPU resources. By integrating Qwen-Instruct for conversation and T5-small for summarization within a Flask web application, the system enables natural language interaction, short-term context retention, and chat summarization through a simple browser interface.

Although CPU-only execution increases response latency, the chatbot remains fully functional and capable of generating relevant responses. The design choices, such as limiting max tokens and using lightweight models (T5-small), help reduce processing time and maintain usability. The implementation highlights how Large Language Models can be deployed cost-effectively without specialized hardware, making the system accessible to users with standard machines.

Future improvements can include model distillation, caching responses, or upgrading to GPU-based inference for improved performance. Overall, the project demonstrates the feasibility of developing a practical AI chatbot using only CPU resources, showcasing a balance between functionality and hardware limitations**.**

MEDICAPS UNIVERSITY

Department of Computer Science and Engineering

About the Department

Established in 2016, the Department of Computer Science and Engineering at MediCaps University has swiftly emerged as a hub for excellence in Computing Education and Research. The department is dedicated to providing high-quality academic programs and fostering a robust research ecosystem that empowers both students and faculty to innovate and excel in the ever-evolving field of computer science.

The department currently offers **B.Tech. (CSE)**, **M.Tech. (CSE)**, and **Ph.D. in Computer Science and Engineering**, covering a wide range of contemporary domains including Artificial Intelligence, Machine Learning, Cyber security, Data Science, Block Chain, and Software Engineering.

A key pillar of the department's success is its **highly qualified and experienced faculty**, who are deeply committed to academic excellence and student development. Their efforts have resulted in numerous quality publications in reputed national and international journals and conferences. The department presently has **40 Ph.D. Research Scholars** working on diverse, cutting-edge problems in the field of computing.

The Department of Computer Science and Engineering also thrives on **active collaborations** with top academic institutions and research organizations across the country and abroad. These strategic partnerships enhance research exposure and industrial connectivity for students and faculty. Our collaborators include **IIT Indore, RRCAT Indore, TCS Research, CDAC Pune, Amity University (Noida)**

With its future-focused curriculum, industry-aligned projects, and an ecosystem that encourages innovation, the department continues to produce highly competent graduates and researchers who contribute significantly to academia and the tech industry.

Address

A.B. Road Pigdamber, Rau
Indore, Madhya Pradesh 453331
+ 917313111500, + 917313111501