Project Documentation: TECH Chatbot

Second Milestone:

Integration of Frontend with Backend through Ollama and DeepSeek

BY MONA RACHEL D.C

1. Aim

The aim of this milestone is to extend the chatbot by integrating both a **locally hosted AI backend (Ollama)** and a **cloud-based AI backend (DeepSeek)** with a **Gradio-powered frontend interface**. The system is designed to allow users to interact with the chatbot seamlessly while choosing the backend that best fits their needs. By combining the speed and privacy of a local lightweight model (gemma3:1b) with the advanced capabilities of DeepSeek's cloud API, this milestone demonstrates the feasibility of building a scalable, flexible, and secure conversational assistant.

2. Description

The TECH Chatbot in its enhanced version is a conversational assistant that provides intelligent responses to user queries by integrating **two AI engines**: Ollama and DeepSeek. The user interacts with the chatbot through a **web interface built using Gradio**, which presents a chat window, a saved chat history box, and an option to clear or restart conversations.

When a user submits a query, the chatbot forwards the input to the selected backend. If the user selects **Ollama**, the request is sent to a locally running instance of the gemma3:1b model on port 11434. If the user selects **DeepSeek**, the request is transmitted to the DeepSeek API endpoint using a valid API key for authentication. Both backends process the input and generate a natural language response, which is then displayed in the Gradio chat window.

The implementation also incorporates **user authentication** to ensure that only authorized users can access the chatbot. This makes the system not just a demonstration prototype but a deployable application that combines AI-driven responses with usability, scalability, and security.

3. Tools Used and Their Contribution

The chatbot has been built using a combination of technologies, each of which plays a specific role:

Tool / Technology	Contribution / Role
Python	Core programming language for backend integration and UI logic.
Gradio	Provides a web-based interface for chat interaction.
Ollama	Local backend running the **gemma3:1b** model for lightweight AI responses.
DeepSeek API	Cloud-based advanced LLM service for generating responses.
Requests Library	Handles HTTP POST requests to both Ollama and DeepSeek APIs.
gemma3:1b Model	Lightweight AI model ensuring efficient local response generation.
Authentication	Username/password login for securing chatbot access.

4. Procedure

The development of this milestone followed a systematic procedure:

Step 1: Install Prerequisites

Python and the required libraries (requests, gradio) were installed. Additionally, Ollama was installed locally on the system to enable the use of the gemma3:1b model.

Step 2: Configure Ollama Model

The gemma3:1b model was pulled from the Ollama repository using the command:

ollama pull gemma3:1b

Step 3: Run Ollama Server

The Ollama server was started using:

ollama serve --port 11434

This ensured that Ollama was available locally at http://localhost:11434.

Step 4: Integrate DeepSeek API

The API key for DeepSeek was obtained from the official platform. The DeepSeek endpoint and authorization header were configured in the Python code to enable secure communication.

Step 5: Implement Chatbot Logic

Python functions were created to query both Ollama and DeepSeek. The chatbot_reply function was implemented to determine which backend to use based on the user's selection.

Step 6: Develop Gradio Frontend

The user interface was built with Gradio Blocks. It included a chat window, a saved chats box, a clear button, and a dropdown menu that allowed users to select between Ollama and DeepSeek.

Step 7: Add Authentication

Login functionality was integrated using Gradio's built-in authentication feature. A username and password are required before accessing the chatbot interface.

Step 8: Launch Chatbot
Finally, the chatbot was launched on a local server:http://127.0.0.1:7860

5. Source Code (Core Example)

```
import requests
import gradio as gr

# ---------------------------
# API Configuration
# ---------------------------
OLLAMA_URL = "http://localhost:11434/api/generate"
```

```
OLLAMA_MODEL = "gemma3:1b"

DEEPSEEK_URL = "https://api.deepseek.com/v1/chat/completions"
DEEPSEEK_API_KEY = "your_api_key_here"

# ---------------------------
# Backend Query Functions
# ---------------------------
def query_ollama(message):
    try:
        response = requests.post(
       OLLAMA_URL,
            json={"model": OLLAMA_MODEL, "prompt": message, "stream": False}
        )
        return response.json().get("response", "⚠️ No response from Ollama.")
    except Exception as e:
        return f"❌ Ollama Error: {str(e)}"

def query_deepseek(message):
    try:
        headers = {"Authorization": f"Bearer {DEEPSEEK_API_KEY}"}
        payload = {
            "model": "deepseek-chat",
            "messages": [{"role": "user", "content": message}]
        }
        response = requests.post(DEEPSEEK_URL, headers=headers, json=payload)
        return response.json()["choices"][0]["message"]["content"]
    except Exception as e:
        return f"❌ DeepSeek Error: {str(e)}"

# ---------------------------
# Unified Chatbot Function
# ---------------------------
def chatbot_reply(message, history, backend):
    if backend == "Ollama":
        return query_ollama(message)
    else:
        return query_deepseek(message)
```

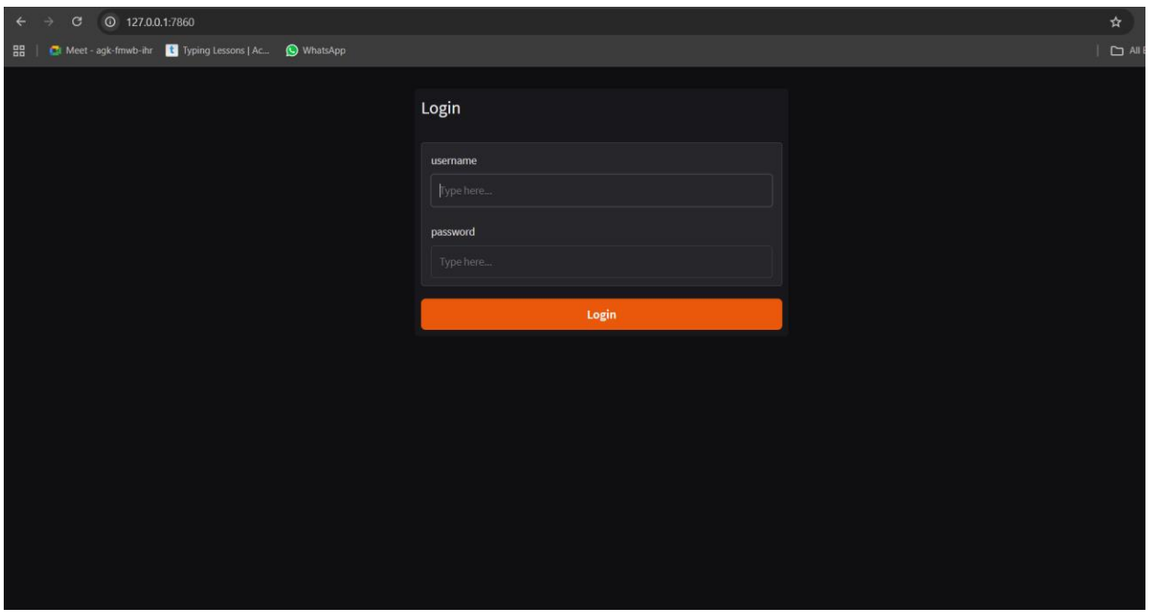
6. Frontend Features

The Gradio frontend provides:

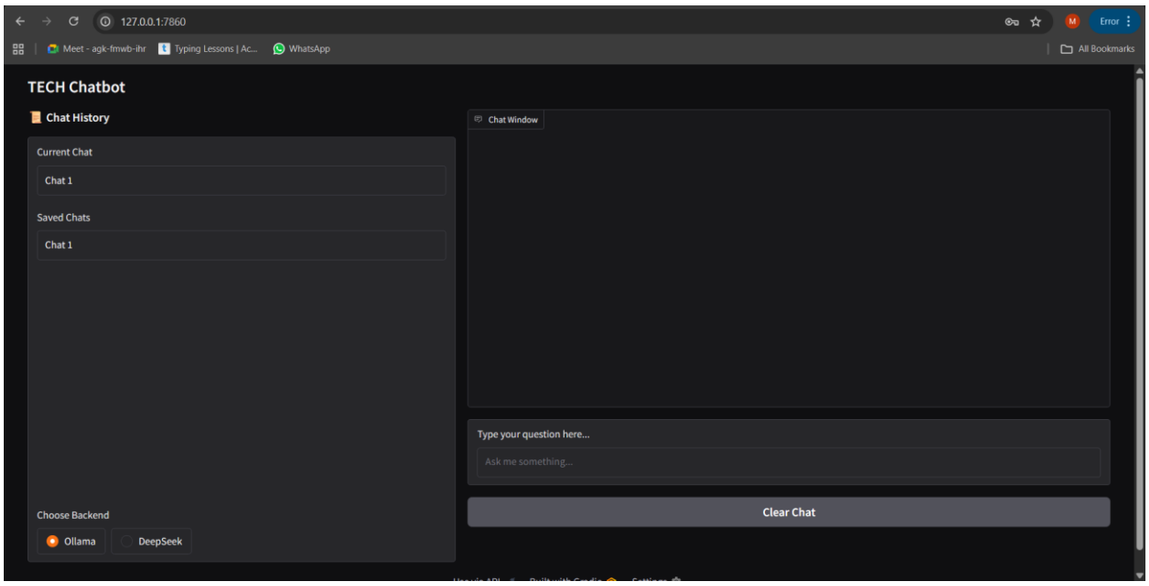
- A **chat window** that displays messages between the user and the AI assistant.
- A **saved chat history** box that records past conversations.
- A **backend selection dropdown** where the user can choose between Ollama (local) and DeepSeek (cloud).
- A **login screen** that prompts the user for credentials before accessing the chatbot.

7.Screenshots

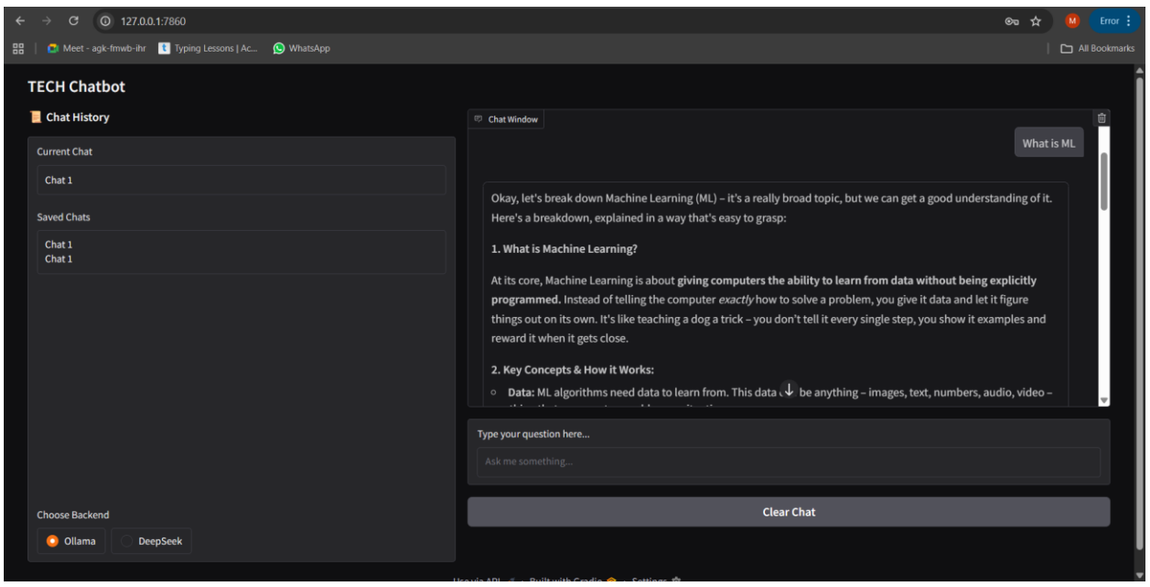
Login page



Chatbot Frontend



Chatbot Reply (using Ollama and Deepseek)



8. Conclusion

This milestone successfully demonstrates the integration of a **Gradio-based frontend** with **two separate AI backends, Ollama and DeepSeek**. By combining these technologies, the chatbot achieves both **local autonomy** and **cloud scalability**. Users can enjoy quick, resource-friendly responses via Ollama or leverage the advanced capabilities of DeepSeek for more complex interactions.

The project highlights the potential of **multi-backend AI systems**, showing that a chatbot can be made more versatile and reliable by allowing multiple backends to coexist under a unified interface. The integration is secure, interactive, and user-friendly, laying the groundwork for further enhancements such as **database-driven storage**, **multi-user sessions**, and **hybrid response generation** in future milestones.