

Project Title:

AI Chatbot using Large Language Models

Submitted by:

Subhranshu das

Technology stack:

Python, FastAPI, Streamlit, Gemini LLM, Github

Abstract

This project showcases the design and implementation of a full-stack AI chatbot using modern Large Language Models. The system architecture enables the user to interact with the bot using a web page created with the help of streamlit. The backend uses python and FastAPI in order to communicate with the Gemini LLM API to generate responses.

1.Introduction

With the rise of Artificial Intelligence the usage of such conversational AI systems such as ChatGPT and Gemini have been used in various fields to answer queries of people and also support them with any information required.

The objective of this project is to replicate such chatbots including the implementation of streamlit for web interface and FastAPI for the backend to make API calls.

2.Objectives

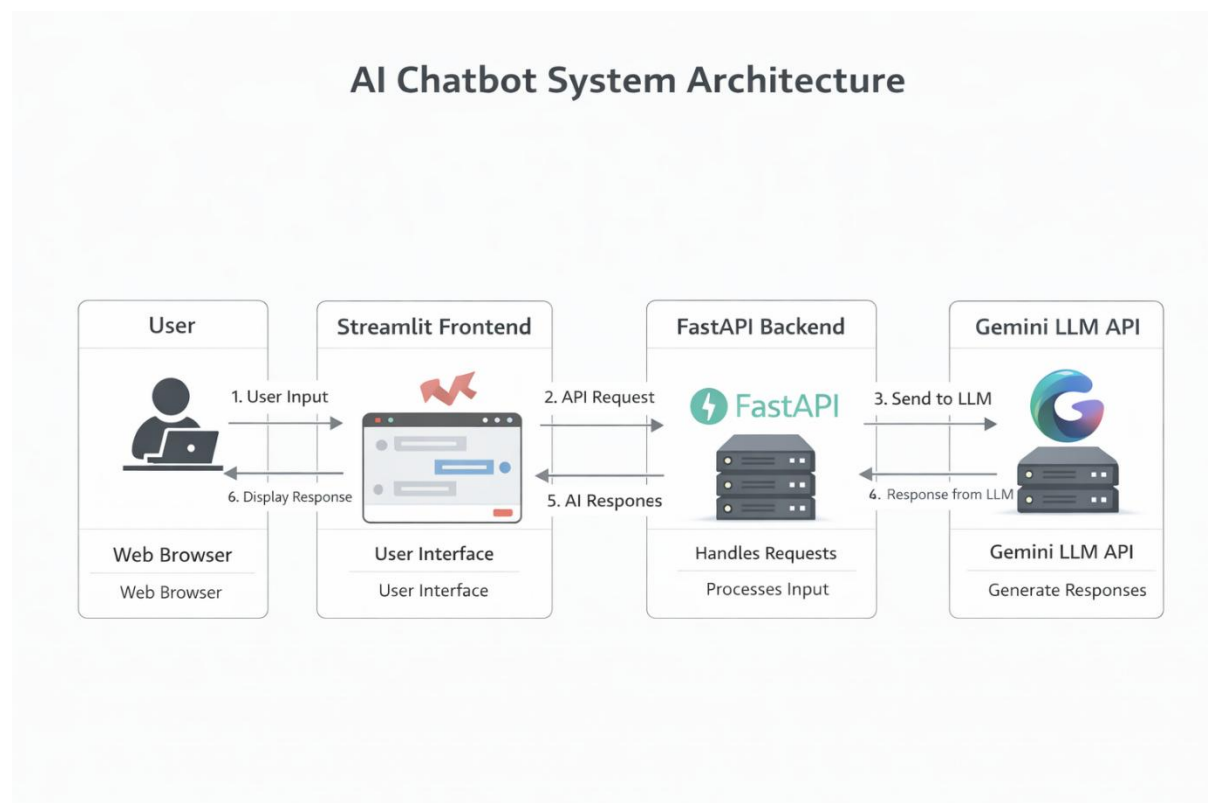
- Design a chatbot using a large language model
- Build a backend API using FastAPI
- Create an interactive frontend using Streamlit
- Implement conversation memory
- Deploy code using Github for version control.

3.System Architecture

Architecture Overview

The system follows a client-server architecture:

1. User interacts with the chatbot UI (Streamlit)
2. Streamlit send user input to FastAPI via HTTP POST
3. FastAPI processes the input and sends it to Gemini LLM API
4. The LLM generates a response
5. FastAPI returns the response to Streamlit
6. Streamlit displays the response to the user



4.Technology Stack

Component

Programming Language

Backend Framework

Frontend Framework

LLM API

Technology

Python

FastAPI

Streamlit

Google Gemini

5.Backend Design (FastAPI)

The backend is responsible for:

- Handling API requests
- Managing conversation context
- Communicating with the LLM app

Key features include:

- RESTful /chat endpoint
- Conversation memory
- Error handling
- Secure API key usage via environment variables

FastAPI was chosen for its speed, simplicity, and automatic API documentation support.

6.Frontend Design (Streamlit)

The frontend provides:

- Chat-based UI
- Real-time message updates
- Session based message storage

Streamlit was chosen to inculcate rapid UI development with minimal code.

Features include:

- User name input
- Chat message display
- Session state for conversation history

7.LLM Integration

The chatbot uses Google Gemini LLM to generate responses.

The backend sends the full conversation context to the model to maintain coherence across multiple turns.

Why Gemini?

- High-quality natural language generation
- Fast response times

- Easy python integration

8. Challenges faced and solutions

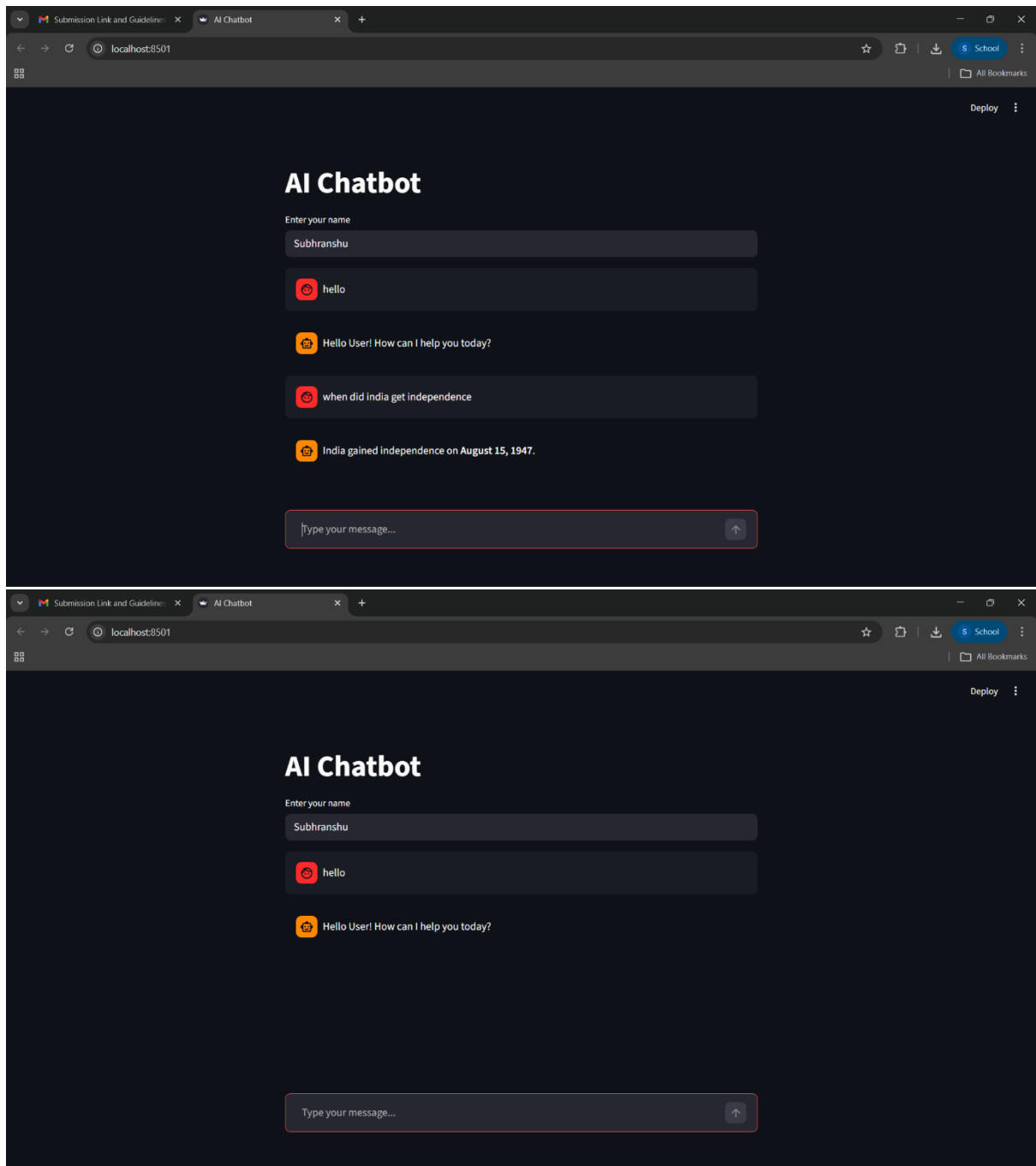
Challenge1:

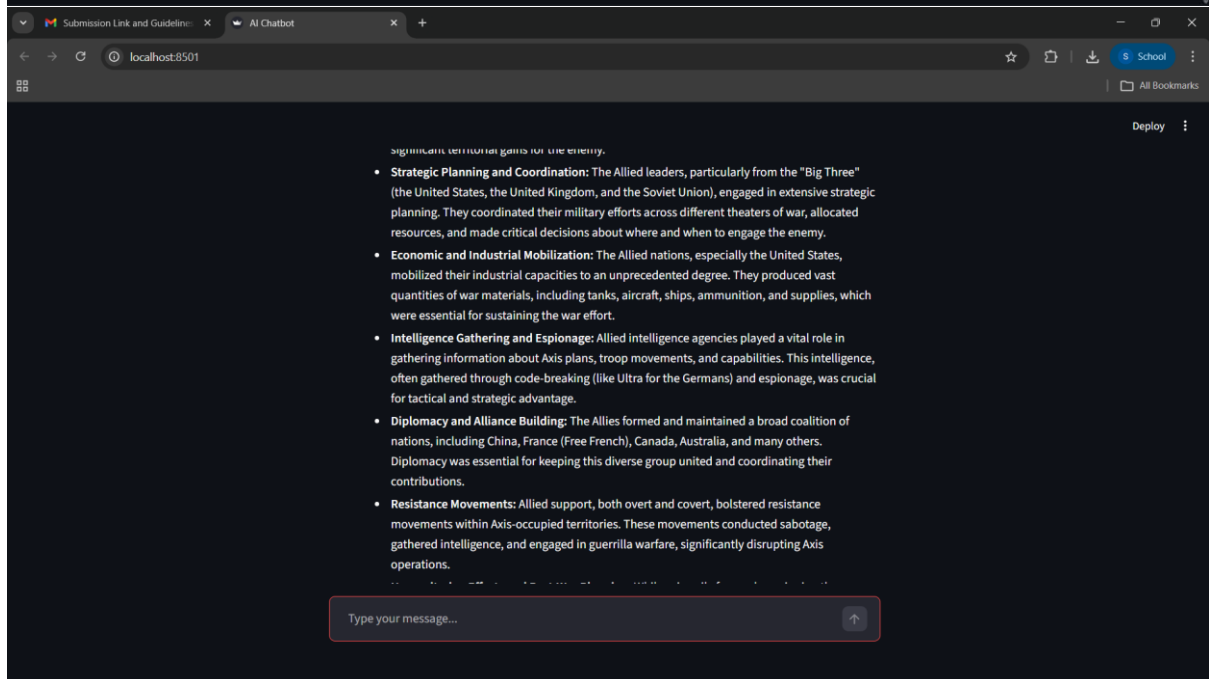
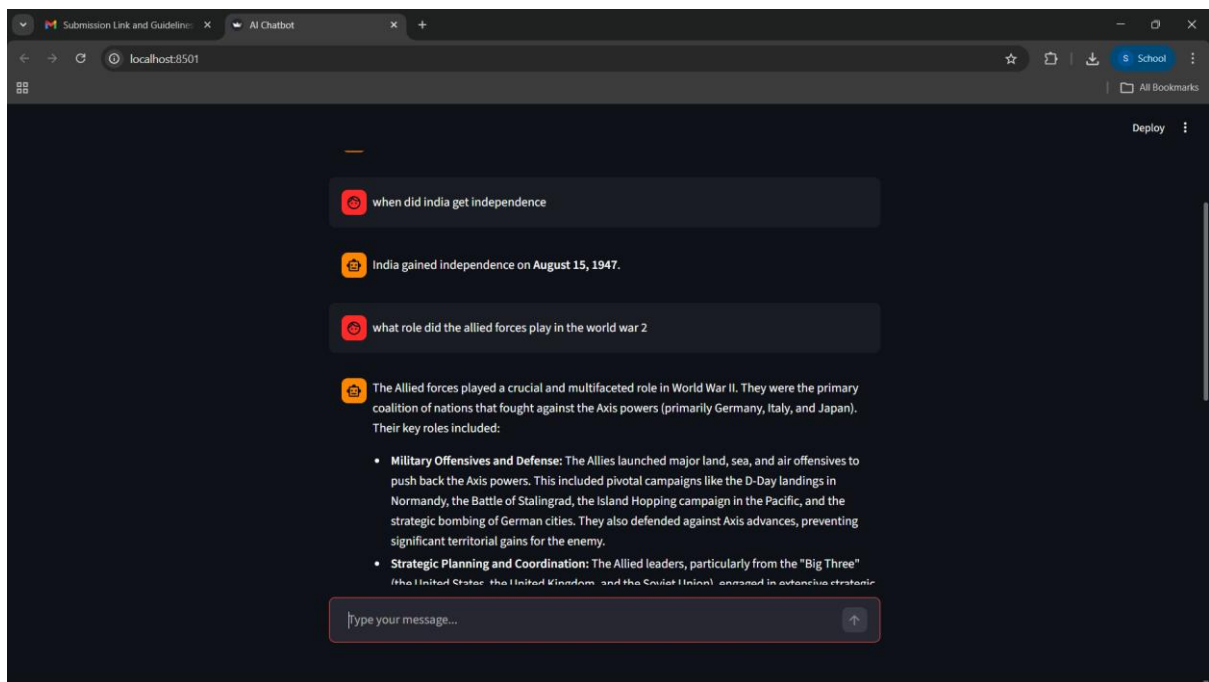
- Problem: Connection error when Streamlit tried to access FAsTAPI
- Solution: Running FastAPI and Streamlit in separate terminals and verifying backend availability using Swagger UI.

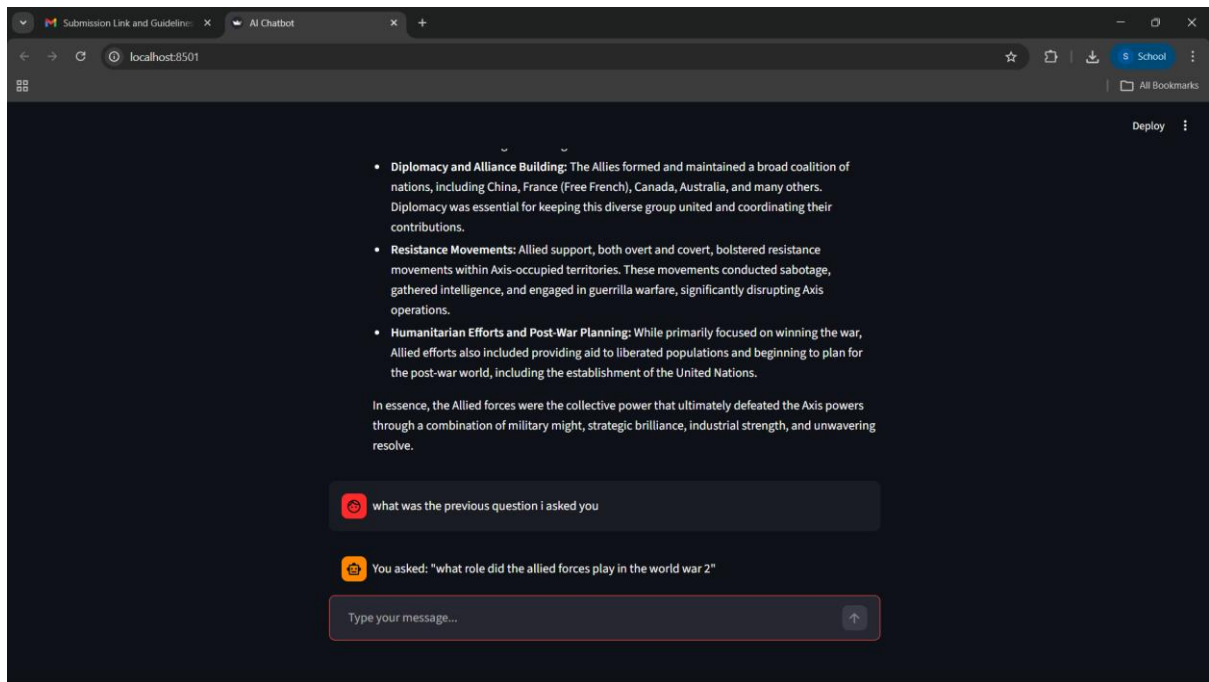
Challenge2:

- Problem: API key exposure in source code.
- Solution: Used environment variables and .env file with .gitignore

9 . Results and Screenshots







As you can see the chatbot generates meaningful responses and also has conversation memory.

10. Future Enchantments

- User authentication
- Persistent chat history using a database
- Deployment on cloud platforms
- Multi language support
- Rate limiting and token usage tracking