Problem Statement 2 : Airline Chatbot

1. Problem Statement

Airline customer service teams handle thousands of repetitive requests daily, such as booking or cancelling flights, checking flight status, seat availability, and understanding airline policies (refunds, baggage, pets, etc.).

Each of these tasks involves collecting multiple details, validating them, and responding appropriately. Manual handling of such requests is **time-consuming, error-prone, and inconsistent** across customer interactions.

Hence, there is a strong need for an **automated, conversational system** that can efficiently handle these queries while maintaining context across multi-turn conversations.

2. Our Solution

We developed an **AI-driven conversational chatbot** that automates airline customer support operations.

The system can:

- Book and cancel flights.
- Check flight status and seat availability.
- Answer questions related to airline policies.
- Maintain context across multi-turn conversations.

The chatbot is built using **Flask** for the backend, a lightweight **HTML/CSS UI**, and **BERT** for intent detection and slot filling.

Although the BERT model has been trained successfully, it is not yet integrated into the live chatbot flow due to time constraints.

3. Objectives

- To design an intelligent assistant capable of handling common airline service tasks automatically.
- To use a **modular architecture** for easy integration with external airline APIs.
- To train and prepare a **BERT-based model** for Natural Language Understanding (NLU).
- To build a foundation that can be extended to other industries beyond airlines.

4. System Architecture

The chatbot follows a modular pipeline structure:

1. **User Message → Flask Backend**: Receives and processes input.
2. **NLU (BERT Model)**: Detects intent and extracts slots such as flight number, date, source, destination, and query type.

3. **Task Handler**: Routes requests to the relevant module (booking, cancellation, status check, policy info).
4. **API/Policy Knowledge Base**: Retrieves information or generates mock responses.
5. **Response Generation**: Sends user-friendly replies back to the interface.
6. **Session Management**: Maintains conversational context across multiple turns.

This modular architecture ensures scalability and easy future integration with airline systems.

5. Trained BERT Model for Intent & Slot Filling

We trained a **BERT-based NLU model** to identify user intents and extract required entities.
It can accurately classify intents such as *BookFlight*, *CancelFlight*, *FlightStatus*, and *PolicyQuery*.
It also extracts relevant slots like flight number, date, source, destination, and policy topic.

6. Implementation Overview

- **Frontend:** HTML, CSS (minimal prototype for chat interface).
- **Backend:** Flask-based server for managing user requests.
- **Database:** Mock data storage for bookings and policy information.
- **NLU Model:** BERT (fine-tuned for intent and slot classification).
- **Notification Logic:** Confirmation messages for booking/cancellation.

7. Challenges and Limitations

- **Time constraints** limited full API integration and UI enhancements.
- BERT NLU model not yet integrated into the real-time chatbot flow.
- Seat availability and cancellation APIs are currently mocked.
- The **UI is basic**, designed only for proof-of-concept purposes.
- Currently focused only on the airline industry.

Despite these challenges, the system design allows for **seamless future expansion and integration**.

8. Future Improvements / Roadmap

- Integrate real-time **airline APIs** for bookings and cancellations.
- Fully integrate **BERT NLU** for natural and flexible user interactions.
- Add an **admin dashboard** to manage policies, FAQs, and logs.
- Enhance UI with a modern design and better visualization tools.
- Expand the chatbot to other domains (hotels, trains, insurance).
- Deploy on **cloud platforms** for scalable and low-latency performance.

9. Results & Takeaways

- Built a working chatbot capable of handling airline support queries.
- Successfully trained and validated a BERT-based NLU system.
- Created a modular, future-ready architecture for scaling.
- Demonstrated proof of concept with multi-turn conversation handling.

10. Conclusion

This project presents a step towards automating airline support using AI-driven conversation systems.

By combining Flask, BERT, and modular task handling, we built a scalable architecture capable of supporting real-time interactions.

With full integration of APIs and the BERT model, the chatbot can evolve into a production-ready system serving multiple industries.