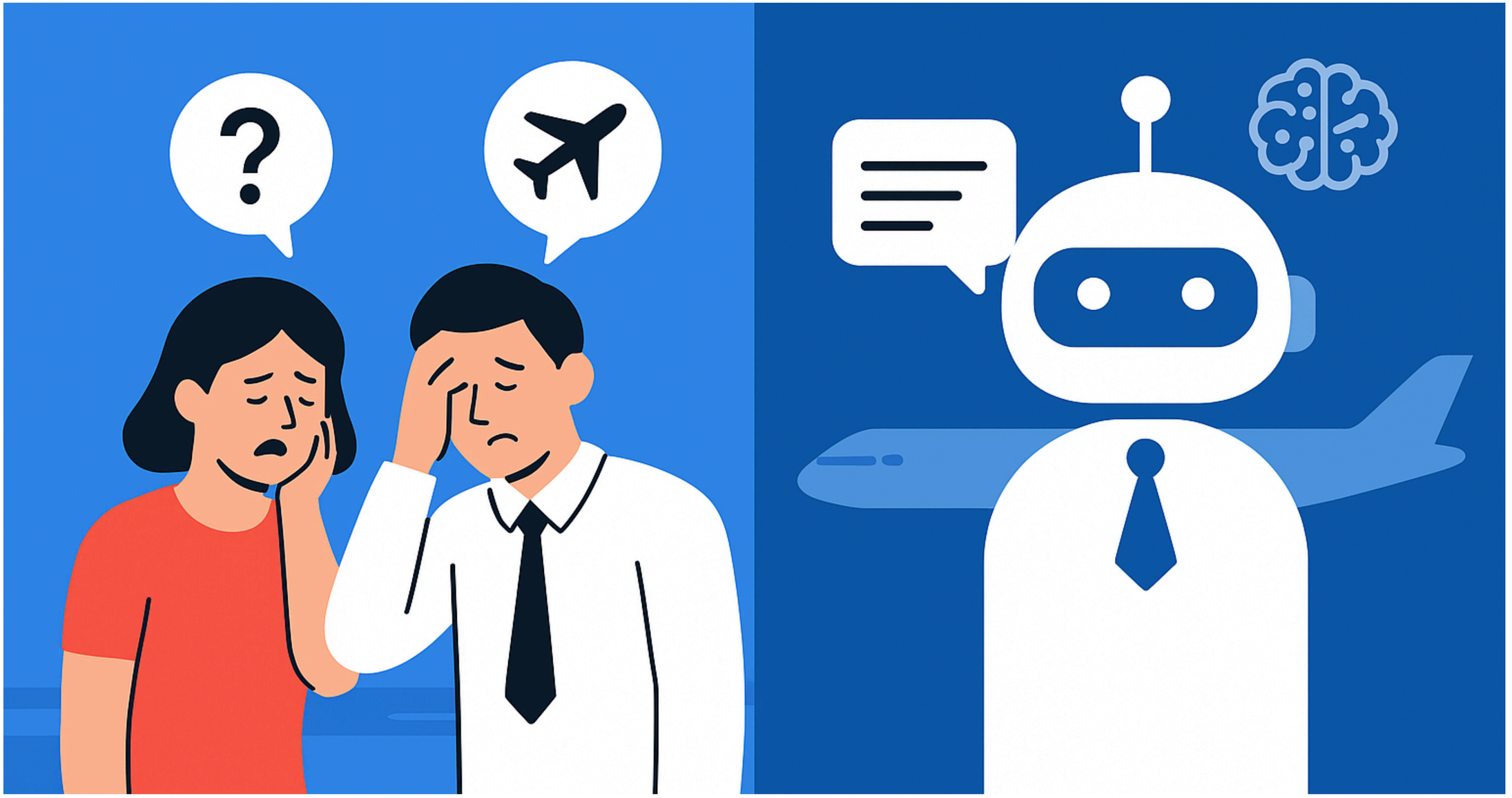
AI-POWERED AIRLINE CUSTOMER SUPPORT ASSISTANT

Real-Time Conversational System with Policy Adherence

Presentation by **SheCode**

Iniyaa N (22z226)
M.S. Padmavathi(22z234)
Madhisha S (22z235)
Thamina anzum A (22z267)



PROBLEM STATEMENT

Airline customer support handles thousands of repetitive queries daily, like flight status, cancellations, and travel policies. Manual responses are slow, inconsistent, and increase agent workload. Customers expect quick, accurate, and natural replies, which traditional systems often fail to deliver. Therefore, an intelligent, context-aware system is needed to automate responses while keeping a human-like tone and following airline policies.

FUNCTIONAL REQUIREMENTS

1. **Input:** Customer's text message or query.

2. **Output:** Contextual, airline-policy-based response.

3. **Policy Adherence:** Responses must follow airline standards.

4. **Agentic Flow:** Handles tasks step-by-step (e.g., fetch → confirm → cancel).

5. **Multi-Turn Conversation:** Supports back-and-forth chat.

6. **Context Retention:** Remembers flight or user details.

7. **Generalization:** Extendable to other domains like hotels or banking.

NON - FUNCTIONAL REQUIREMENTS

1. **Low Latency:** Replies must appear instantly.

2. **Offline Operation:** Should work without the internet.

3. **Scalability:** Easily adaptable to multiple industries.

4. **Human-Like:** Natural and polite tone.

5. **Maintainability:** Modular structure for easy updates.

LLM MODEL - OLLAMA

Model Used: Ollama LLM (small & efficient for offline use)

Purpose: Understands customer messages & generates natural responses

Features:

- Handles multi-turn conversations
- Retains context (flight details, PNR)
- Follows airline policies for accurate replies

Benefits: Low latency, works offline, lightweight for real-time demo

Adaptable: Can be extended to other industries (hotels, banking)

RAG & FAISS

RAG (Retrieval-Augmented Generation):

Combines retrieved documents with LLM for accurate responses.

Ensures answers follow airline policies.

FAISS (Facebook AI Similarity Search):

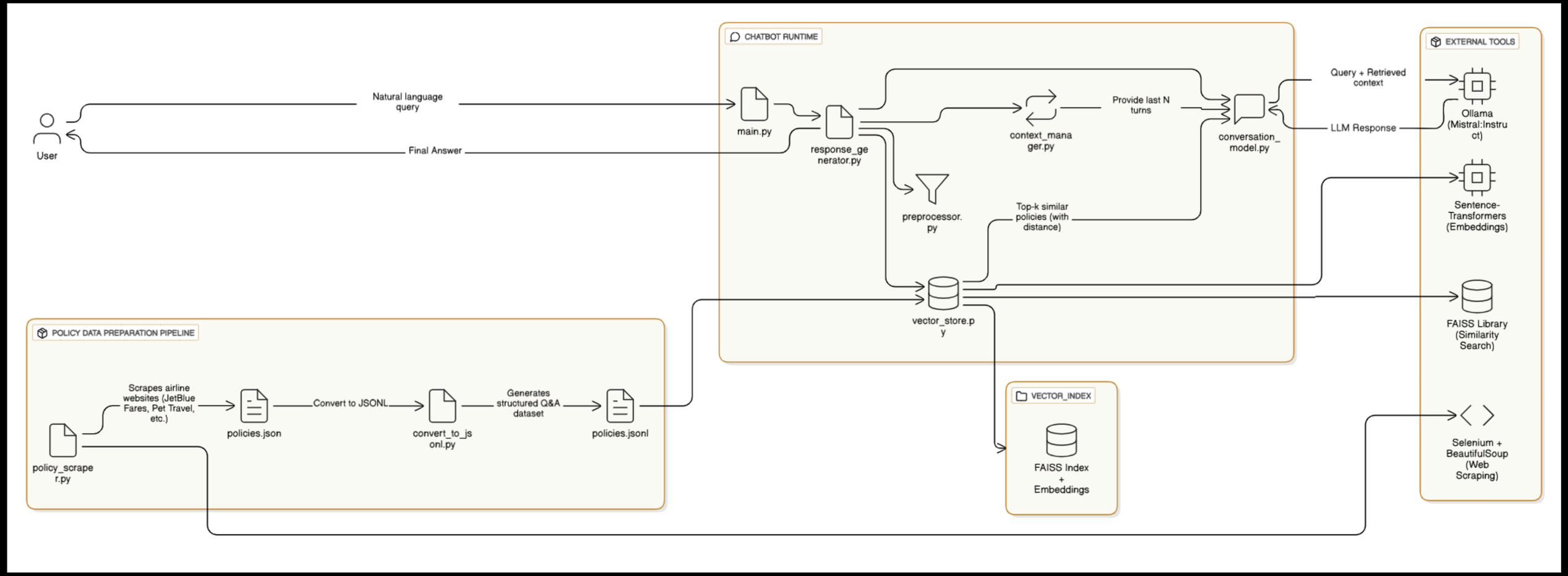
Vector database for fast document retrieval.

Stores airline policies, FAQs, and flight info embeddings.

Workflow:

Query → FAISS retrieves relevant info → LLM generates natural response.

SYSTEM ARCHITECTURE



[diagram link]

Web Scraping

Goal:

To extract airline policy data (like Fares, FAQs, and Pet Travel guidelines) from JetBlue's official website.

Process Workflow:

- Access JetBlue policy pages using Selenium.
- Parse the content using BeautifulSoup (FAQs, Pet Travel Checklist, Fare details).
- Clean and format extracted data into Question–Answer pairs.
- Save structured results into policies.json.
- Convert JSON → JSONL for RAG-based chatbot integration.

Retrieval-Augmented Generation (RAG)

Concept:
RAG combines information retrieval with language generation.

In our project:

1. **Vectorization:**
   - Policies are converted into embeddings using SentenceTransformer – **all-MiniLM-L6-v2**.
   - Each Q&A pair → dense vector stored in **FAISS index**.
2. **Query Processing:**
   - User query → embedding → similarity search in FAISS.
   - Retrieves **Top-k most relevant policies**.
3. **Filtering:**
   - Distance threshold ensures only on-topic airline queries are kept.
   - Off-topic queries are rejected politely.

Vector Database (FAISS)

Role of FAISS:

- Efficiently searches through thousands of vector embeddings.
- Uses **L2 distance** to find similar policy questions.
- **Stores:**
  - policy_index.index → FAISS index
  - embeddings.npz → vector matrix
  - metadata.json → text references

Why FAISS?

- Fast, scalable, supports similarity search within milliseconds.

LLM Integration (Ollama + Mistral)

Steps:

- The retrieved policy context is passed to the LLM (mistral:instruct model via Ollama).
- The LLM is prompted with:
    - **Context:** <retrieved policy snippets>
    - **User Query:** <question>
- The LLM generates a concise, accurate response using airline policies only.

Libraries used:

- **ollama** for LLM communication
- **conversation_model.py** → handles model calls, error handling

Summary of files in the Project

Stage	File	Role
Pre-processing	preprocessor.py	Clean text for embedding
Data Formatting	convert_to_jsonl.py	Convert JSON → JSONL
Embedding + Retrieval	vector_store.py	Build FAISS index, retrieve policies
Context Management	context_manager.py	Maintain chat history
LLM Connection	conversation_model.py	Send prompt to Mistral model
RAG Pipeline	response_generator.py	Orchestrate retrieval + LLM
Integration / Execution	main.py	Run chatbot with all modules

TECH STACK USED

- **Python 3.11.0:** Main programming language
- **Ollama:** Local LLM inference (Mistral Instruct model)
- **FAISS:** Facebook AI Similarity Search for vector indexing
- **Sentence Transformers:** all-MiniLM-L6-v2 for embedding generation
- **NumPy:** Numerical computations and vector operations

THANK YOU