Final Project Report: AI-Chatbot-Aviation-Tracker

Submission Date: April 9, 2024
Submitted by: **Kaan Yazıcıoğlu (97364)** (SD4), **Caner Akcasu (97334)** (SD1)

0. Requirements & How to Run.

GitHub & Clone: https://github.com/kaanzapkinus/AI-Chatbot-Aviation-Tracker

To run the project, please follow these **steps**:

1 - Install Python (version 3.10 or above) on your system.

2 - Open a terminal in the project directory and **run this command** to install required packages:

```
pip install -r requirements.txt
```

3 - **Apply migrations** with this command:

```
python manage.py migrate
```

4 - **Start the Django server** by running:

```
python manage.py runserver
```

5 - Open your web browser and **go** to:

http://127.0.0.1:8000/

1. Introduction

The AI Chatbot Aviation Tracker is an improved version of our Django-based flight tracking system. The original project offered basic flight data, but now it includes an AI chatbot powered by the DeepSeek-R1 API. This lets users ask questions about flights using natural language and get real-time information quickly.

The system combines a strong backend with live data from the Aerodatabox API. It supports flight status updates, delay checks, and route details through a user-friendly chat interface. This report covers the project's goals, scope, technical setup, features, and challenges.

2. Objectives & Scope

Objectives:

- **Conversational Flight Tracking**: Allow users to communicate with the system using simple, natural language. The AI chatbot answers questions such as flight status, delays, or time of arrival, making the system easy to use even for non-technical users.

- **Real-Time Data Integration**: Use trusted external APIs to collect the most recent flight data, including departure and arrival times, delays, route information, and live updates. This helps users stay informed with accurate and up-to-date information.

- **Improved Accessibility**: Create a system that is accessible from any modern browser, with no need for complex installations or registrations.

Scope:

- **Backend Integration**: Upgrade the existing Django-based system by adding DeepSeek-R1, an AI language model hosted via Groq. This allows the backend to understand and respond to flight-related questions effectively.

- **Intuitive Chat Interface**: Build a smooth and interactive chat interface using HTML, CSS, and JavaScript. This design allows users to interact without page reloads, creating a faster and more modern user experience.

- **Data Sources**: Flight data is mainly fetched from the Aerodatabox API, accessed through a proxy (MagicAPI) to improve reliability and speed.

- **AI Integration & Context Management**: Instead of training a model from scratch, we integrated the DeepSeek-R1 API. The AI receives structured flight data and responds based on carefully designed prompts. We also applied custom rules to limit the AI's responses to only the specific flight in question, keeping answers short and relevant.

- **System Reliability**: The project includes error handling and fallback mechanisms to ensure the application works well even when data is incomplete or an API fails to respond.

3. Technology Stack

The project is constructed with a multi-layered architecture to efficiently manage presentation, intelligence, and data integration:

- **Presentation Layer:**
    - ***Django****:* Serves as the main web framework for routing and backend processes.
    - ***HTML/CSS/JavaScript****:* Provide dynamic UI updates and a modern, responsive design that avoids full page reloads.
- **Intelligence Layer:**
    - ***DeepSeek-R1-70B api****:* The central AI model deployed to process and respond to natural language queries.
    - ***Groq LPU****:* Optimizes AI inference for faster response times in a real-time tracking environment.
- **Data Layer:**
    - ***Aerodatabox API****:* Supplies the primary flight data including real-time status, departure/arrival details, and more.
    - ***Groq API****:* Provides access to the DeepSeek-R1 70B model, used for fast and accurate AI responses in natural language tasks.

4. Implementation

User Interface & Experience:

- The front-end uses custom CSS styles inspired by Tailwind design, but Tailwind itself is not used.
  UI components include:

  • **Flight Search Form:** A dedicated section that validates flight number inputs (e.g., "TK123") and displays dynamic flight information, such as departure/arrival times and flight progress.
  • **Dynamic Chatbox:** A floating chat widget that supports natural language queries. Users can ask questions like “Is TK1991 delayed?” and receive context-rich responses along with predictive notifications.
  • **Visual Flight Tracker:** Interactive visual elements such as progress bars and status badges present key flight details, enhancing data comprehension.

AI Integration & Functionality

- **API-Based AI Integration**: The chatbot uses the pre-trained **DeepSeek-R1-distill-llama-70b** model hosted on **Groq**, integrated via the **LangChain** framework.

- **Custom Endpoints**:

  - *search_flight* –Fetches and formats real-time flight data from Aerodatabox via MagicAPI.

  - *chat_assistant* Sends user queries and flight data to the AI, then returns structured responses.

- **Prompt Engineering**: Specific rules are applied to guide the AI, including:
  • Limiting responses to **2 sentences**
  • Standardizing **date and time formats**
  • Restricting replies to **only the relevant flight**
  • Enforcing **language** and **unit** consistency

- **Data Handling**: The system includes error handling for API failures and selects the most relevant flight if multiple results exist. It also calculates **flight progress** and **time remaining** using dynamic logic.

- **Response Processing**: AI replies are cleaned to remove formatting artifacts (like markdown symbols), ensuring the user receives clear and consistent responses.

Performance Optimizations:

- **Low Latency Handling:** The integration of Groq LPU accelerates AI inference, ensuring that the response times meet the requirements of a real-time tracking system.

- **Asynchronous Data Processing:** Asynchronous JavaScript techniques (AJAX/Fetch API) are employed on the front-end to seamlessly update flight data without interruptions.

Challenges Faced:

- **API Reliability:** Issues with the Xmagic API limited its integration, which necessitated a strong reliance on the Aerodatabox API.

- **Data Consistency:** Ensuring that real-time data remains consistent across multiple dynamic updates was challenging and required careful management of asynchronous data flows.

- **User Experience Enhancements:** Iterative testing was necessary to balance rich feature integration with overall system performance, leading to continuous improvements in both backend processing and frontend responsiveness.

---

5. Conclusion

The AI Chatbot Aviation Tracker successfully brings together a Django backend, real-time flight data, and natural language AI to deliver a modern flight tracking experience. The system allows users to ask simple questions about flights—such as delays, departure times, or arrival details—and receive clear and accurate answers instantly through an interactive chat interface.

Throughout the development process, we aimed to build a solution that is fast, reliable, and easy to use. By using trusted APIs and carefully structured prompts, the AI component was able to provide relevant responses without going off-topic. Features like visual flight progress, smooth data updates, and clean design helped create a better user experience overall.

All key objectives of the project have been met. The system works without the need for future updates or additions. It is complete, functional, and capable of supporting real-world usage. The AI Chatbot Aviation Tracker shows how modern technologies like large language models and live APIs can be combined effectively in a web-based application.