Flight Analytics Dashboard

Detailed Project Documentation

Introduction

The **Flight Analytics Dashboard** is a data-driven web application designed to analyze and visualize **air traffic activity, airport information, flight details, and operational insights** with a focus on **Indian airspace**.

Due to restrictions in paid aviation APIs, this project intentionally uses **free and open aviation data sources**, combined with **local database simulation**, to demonstrate real-world analytics workflows.

Objectives of the Project

The main objectives of this project are:

- To collect and analyze **aircraft and flight-related data**
- To visualize **live air traffic over India**
- To design a **relational database schema** for aviation analytics
- To build an interactive **dashboard using Streamlit**
- To simulate **flight search, delay analysis, and airport analytics**
- To handle **API limitations and real-world data challenges**

Technologies Used

Programming & Frameworks

- **Python 3.10**
- **Streamlit** – Web dashboard
- **Pandas** – Data processing
- **Requests** – API integration
- **Plotly** – Interactive charts
- **AG Grid** – Data tables

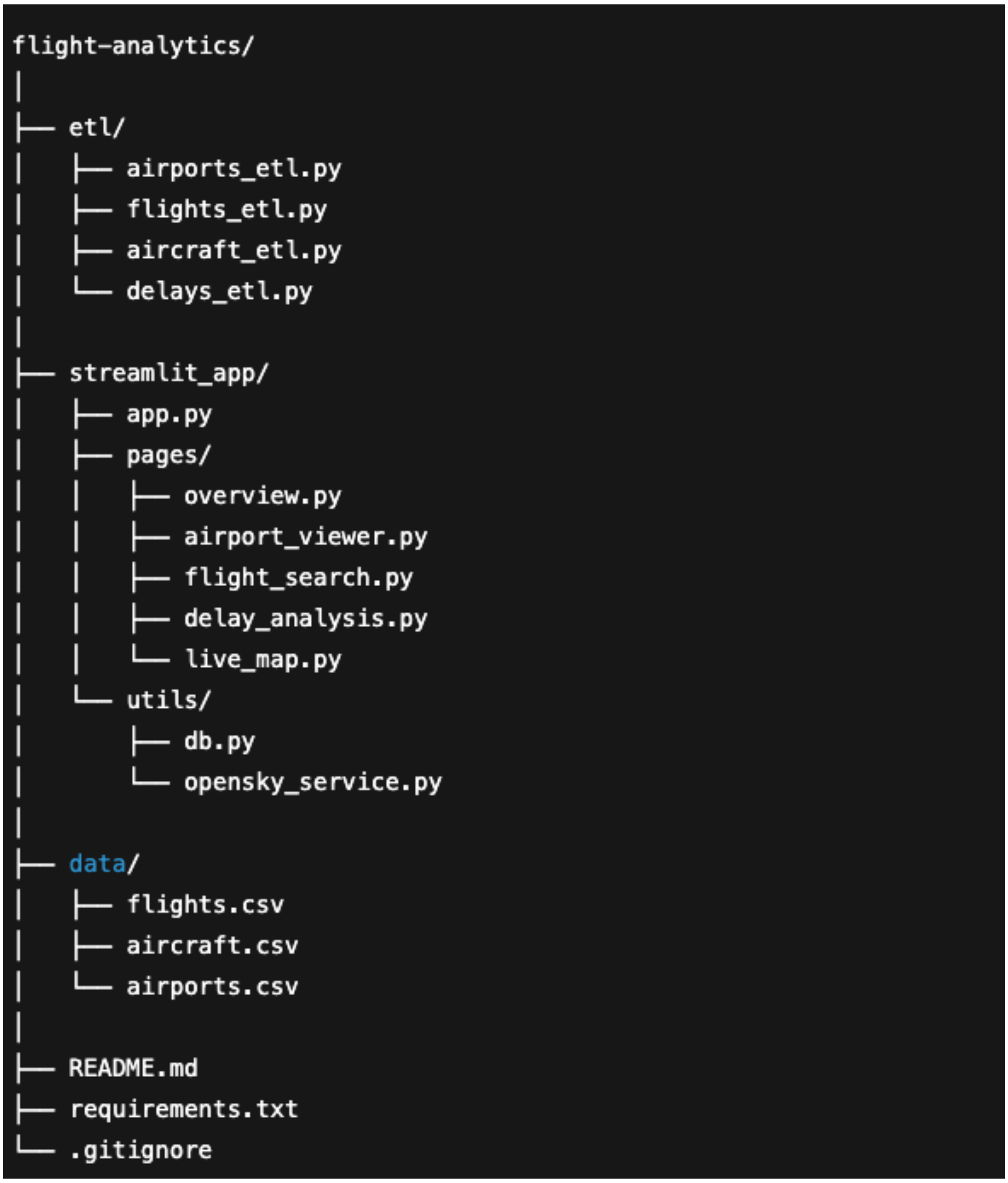
Database

- PostgreSQL
- SQLAlchemy
- psycopg2

APIs & Data Sources

- OpenFlights Airport Database (GitHub)
- OpenSky Network (Free Tier)
- ADS-B Exchange (Public endpoint – experimental)

Project Architecture



```
flight-analytics/
│
├── etl/
│   ├── airports_etl.py
│   ├── flights_etl.py
│   ├── aircraft_etl.py
│   └── delays_etl.py
│
├── streamlit_app/
│   ├── app.py
│   ├── pages/
│   │   ├── overview.py
│   │   ├── airport_viewer.py
│   │   ├── flight_search.py
│   │   ├── delay_analysis.py
│   │   └── live_map.py
│   └── utils/
│       ├── db.py
│       └── opensky_service.py
│
├── data/
│   ├── flights.csv
│   ├── aircraft.csv
│   └── airports.csv
│
├── README.md
├── requirements.txt
└── .gitignore
```

Workflow of the Project

Step 1: Data Collection

- Airports loaded from **OpenFlights GitHub dataset**
- Flights and aircraft seeded using **CSV-based simulation**
- Live aircraft positions fetched via **OpenSky**

Step 2: ETL (Extract, Transform, Load)

- Extract raw data
- Clean missing coordinates
- Normalize columns
- Load into PostgreSQL tables

Step 3: Database Storage

- Structured relational tables
- Indexed primary keys
- Ready for analytical queries

Step 4: Dashboard Visualization

- KPIs (counts, averages)
- Interactive tables
- Live maps
- Charts and indicators

Database Schema Design

Airports Table

```
airports (
    iata_code TEXT PRIMARY KEY,
    icao_code TEXT,
    name TEXT,
    city TEXT,
    country TEXT,
    latitude REAL,
    longitude REAL,
    timezone TEXT
)
```

Aircraft Table

```
aircraft (
    registration TEXT PRIMARY KEY,
    model TEXT,
    manufacturer TEXT,
    icao_type_code TEXT,
    owner TEXT
)
```

Flights Table

```
flights (
    flight_id TEXT PRIMARY KEY,
    flight_number TEXT,
    aircraft_registration TEXT,
    origin_iata TEXT,
    destination_iata TEXT,
    scheduled_departure TIMESTAMP,
    actual_departure TIMESTAMP,
    scheduled_arrival TIMESTAMP,
    actual_arrival TIMESTAMP,
    status TEXT,
    airline_code TEXT
)
```

Airport Delays Table

```
airport_delays (
    id SERIAL PRIMARY KEY,
    airport_iata TEXT,
    delay_date DATE,
    total_flights INTEGER,
    delayed_flights INTEGER,
    avg_delay_min INTEGER,
    median_delay_min INTEGER,
    canceled_flights INTEGER
)
```

Dashboard Modules Explanation

Overview Dashboard

- Total airports
- Total flights
- Total aircraft
- Live aircraft over India
- Average speed & altitude (derived)

Airport Explorer

- Indian airports only
- Airport selection
- Nearby live aircraft
- Map visualization
- Grounded vs airborne aircraft

Flight Search

- Search by flight number
- Search by airline
- Filter by origin & destination
- Status-based filtering

Delay Analysis

- Historical delay simulation
- Live congestion indicators
- Speed & altitude analytics
- Congestion risk score

Live Map

- Real-time aircraft plotting
- India-only bounding box
- Auto-refresh mechanism

API Limitations & Workarounds

Challenges Faced

- OpenSky free API does not provide:
    - Routes
    - Departure/arrival times
    - Airport delays
- Rate limits and anonymous access restrictions
- ADS-B Exchange public endpoints unstable

Solutions Implemented

- Used **bounding box filtering** for India
- Simulated missing attributes using CSV data
- Derived delay indicators analytically
- Centralized API calls to reduce overload
- Implemented caching using @st.cache_data

Key Insights from the Project

- Free aviation APIs are **state-based**, not schedule-based
- Real-world projects require **data simulation**
- API reliability is a real production challenge
- Separation of ETL, database, and UI improves maintainability
- Streamlit is highly effective for rapid analytics dashboards

Learning Outcomes

- Hands-on experience with **ETL pipelines**
- Designing **normalized relational schemas**
- Integrating **live APIs**
- Handling **real-time data issues**
- Building a **full-stack data analytics project**

Future Enhancements

- Integrate **ML delay prediction**
- Add airline-wise performance analytics
- Deploy using Docker
- Scheduled ETL automation
- Advanced geospatial visualizations

Conclusion

This project successfully demonstrates how **aviation analytics systems** can be built using **free data sources**, proper database design, and interactive dashboards. Despite API limitations, meaningful insights were generated using analytical techniques and simulated data.

The project reflects **industry-level thinking**, making it suitable for academic submission, portfolio showcase, and technical interviews.