Real Time Aircraft Data React Application

Kenya Kimata
Deployment Link: https://sleepypenguin763.github.io/Development/

Overview:

Main purpose of this project was to extend the course work from CSCI 1300 (UI/UX)'s Development assignment, where I made a react application that shows live aviation data. This webpage only utilizes free API and dataset that everyone can access, so there are no Geocoding involved, and the data that is presented on the website is not fully accurate. The feature of this website can also be considered as a text version of FlightRadar24.
However, this application consists of unique features that other services such as FlightRadar24 does not have. That is, a user can bookmark a flight. This allows the user to view multiple desired flights instead of scrolling through the page back and forth. Furthermore, users can filter and sort with desired options displayed on the website. Although this does not bring much to the user experience, if a user is specifically interested in tracking long-distance / short-distance flights, they can filter out undesired flights.

Goal and Value of this Application:

The goal of this application is to see the LIVE information regarding the airplanes flying around in the world. The website can get the data of most airplanes, which are claimed via open-source API called OpenSky (will be noted as the API moving forward). This API claims the LIVE data of aircraft flying around the world with its ICAO24(basically id), flight's callsign, current position and velocity. In order to get the details of the route, I used vradarserver/standing-data dataset (will be noted as the dataset moving forward) where I can map flight's callsign to its routes and airport/aircraft data, since OpenSky API does not take care of that. There are two

python scripts I wrote where it converts csv to JSON file, and then to convert JSON file to HTML format JSON to use it as a mock REST API. The logo of each airline is taken from this GitHub repository. It does not include all possible airline logo, however.

Mock API I created:

There are two places to access the API I created. One repository is specifically dedicated to store route data, which can be accessed here. I have attached a sample directory setup image below, but please read the README in the repository to learn how to use this API. The other API stores the data regarding airports, airlines, countries, and aircrafts, which can be accessed here.
These mock REST APIs were originally stored as a csv in *vradarserver/standing-data* GitHub repository. I made a Python script that converts these csv files to a directory-JSON pair, which was then uploaded to my mock REST API GitHub repositories.

Mock API Setup for flight routes:

Directory Setup

```
main
├── routes
|   ├── Folder name: First 3 character of ICAO airline code
|       ├── Folder name: Full flight callsign (i.e.: UAL1)
|           ├── Callsign — index.html
|           ├── Code — index.html
|           ├── Number — index.html
|           ├── AirlineCode — index.html
|           ├── AirportCodes — index.html
|       ├── index.html
|   ├── index.html
```

data you want	REST API URL
All Routes of Given Airline[Airline=ANA]	https://sleepypenguin763.github.io/Flight-Routes/routes/ANA/
Routes of Given Callsign[Airline=ANA1]	https://sleepypenguin763.github.io/Flight-Routes/routes/ANA/ANA1

Troubles and Issues:

One might wonder what the hardest portion of this project was. Personally, finding open-source dataset and cleaning / processing the data took a lot of time. Writing a python script that makes directories and JSON files also took a long time, especially

since each iteration of running the script took roughly 20-30 minutes. This is because the dataset containing the routes data is extremely large, and even deploying the JSON files as HTML on GitHub takes over 20 minutes to complete. Furthermore, the dataset I have been using changes frequently, which required frequent update with my mock API. As a future task, I am considering creating an automated code that updates my API when the dataset repository gets updated.

Sample Webpage:

The image of the sample website view is attached at the bottom of this document. Currently, the website initially loads predefined dataset. This is to reduce the number of API calls and to load the page faster. Please click the orange button at the bottom of the page to load live data.

Documentation:

I have attached the PDF of my README for this project. You can also access it here.

Sources:

OpenSky: https://openskynetwork.github.io/opensky-api/rest.html
Dataset for Mapping Callsigns to Routes: https://github.com/vradarserver/standing-data
Airline Logos: https://github.com/sexym0nk3y/airline-logos

Live Flight Detail Viewer

Speed:

Altitude:

Route Distance:

Show Flights with unknown route distance

Apply Filter(s) and Options!

RESET ALL FILTERS

SHOW UNRECOGNIZED FLIGHTS

VIEW BOOKMARKED FLIGHTS

Sort By

Default

SORT

AZU4613

Airline: Azul Linhas Aéreas Brasileiras

Route (ICAO): SBZM-SBKP

Route: Juiz de Fora => Campinas

Total Route Distance: 440.20 km

Velocity: 413.028 km/h

Geographic Altitude: 6385.56

Longitude / Latitude: -44.5473 / -21.6398

FIN5LA

Airline: Finnair

Route (ICAO): EFHK-ESSA

Route: Helsinki => Stockholm

Total Route Distance: 398.57 km

Velocity: 451.836 km/h

Geographic Altitude: 5311.14

Longitude / Latitude: 19.406 / 59.7784

WEN3195

Airline: WestJet Encore

Route (ICAO): CYYC-CYQU

Route: Calgary => Grande Prairie

Total Route Distance: 556.18 km

Velocity: 477.108 km/h

Geographic Altitude: 5585.46

Longitude / Latitude: -114.6407 / 51.355

AAL1113

Airline: American Airlines

Route (ICAO): MMSD-KDFW

Route: San José del Cabo => Dallas-Fort Worth

Total Route Distance: 1647.94 km

Velocity: 484.668 km/h

Geographic Altitude: 5402.58

Longitude / Latitude: -106.0412 / 40.0811

LOAD PREDEFINED DATA (PLEASE SELECT THIS WHEN YOU CAN NOT LOAD THE PAGE)

Development

Link to Deployed Website

https://sleepypenguin763.github.io/Development/

> **Warning**
>
> It will take roughly 30 seconds to load the page if you decide to load the LIVE Data by clicking on the button, and this is because there are a lot of API fetching going on. You can see the loading progress bar once you get to the page, but if the progress bar does not reach 100% even after waiting for a minuite or so, please refresh the page. Also, once you can see the content of the page, please DO NOT refresh the page as this will trigger new set of API fetching, and OpenSky has limited free API calls per timeframe.
>
> **Note**
>
> Initially, the application will load predefined data, which only consists of ~1000 data points. This is to reduce the time for you when grading the assignment. The content of this data will most likely be outdated, since the data is taken from past flights. If you want to load the page with LIVE data, please click on the orange button on the loading screen / main screen. (It is more cool to see this since you can search the flight with callsign on Google, and you'll see that the data mostly matches with what is being displayed on my screen)

API

Here is where the mock REST API regarding the airline, aircraft, airports, and country lives. Please read the README in side this repo to see how the Mock REST API is organized.

Goal and Value of the Application

The goal of this application is to see the LIVE information regarding the airplanes flying around in the world. The website can get the data of most airplanes, which are claimed via open source API called OpenSky. This API claims the LIVE data of aircraft flying around the world with its ICAO24(basically id), flight's callsign, current position and velocity. In order to get the details of the route, I used this dataset where I can map flight's callsign to its routes and airport/aircraft data, since OpenSky API does not take care of that. There are two python scripts I wrote where it converts csv to JSON file, and then to convert JSON file to HTML format JSON to use it as an mock REST API. The logo of each airline is taken from this GitHub repository. It does not include all possible airline logo, however.

Usability Principles Considered

This is something that needs more improvement, especially since it is taking roughly 30seconds to initially load the page. However, this is hard to solve since the assignment requires everything to be on one page, so I was not able to implement pagination. However, the website is mostly user friendly in a sense that it is implemented using responsive designing principles.

Now, there is a checkbox stating 'Show Flights with unknown route distance'. You might be confused about the difference between this and 'Hide unrecognized flights'. These are TWO DIFFERENT BUTTONS. Flights with valid/invalid callsign can have undefined routes, and this is independent of callsign/airline being recognizable. For instance, it is possible that we have some flight XXX where we know the operator of the aircraft but we do not know where the origin/destination is. In this case, we can not determine the route distance, and by unchecking the checkbox saying 'Show Flights with unknown route distance', it will remove from the list.

> **Note**
>
> Unrecognized flights are defined as the following: a) if a flight has unknown callsign OR b) flight's operator is unknown. This filter will basically remove all private jets and ground vehicle(such as towing car).

Organization of Components

Each component consists of the following: - Airline: (shows airline name if available) - Route (ICAO): (shows route in ICAO format if available) - Route: (shows route in text format if available) - Total Route Distance: (shows total route distance if available in km. Calculated using the longitude / latitude data of origin/destination airport) - Velocity: (current plane's velocity in km/h) - Geographic Altitude: (current altitude of the plane in meters) - Longitude / Latitude: (current longitude and latitude of the plane)

Now, the "Route" and "Total Route Distance" component will not appear if the Airline/Route information is not available for the given callsign.

How Data is Passed Down Through Components

There are few files: App, APIRequests, Distance, Filter, ProgressBar, and Sort.js. App does the bulk of initial API triggering and rendering the component that you see on the page. APIRequests is the file that deals with fetching data from the mock API server or from OpenSky. Sort is the file that makes sorting options and implements sorting algorithm. Sorting will be done in ascending order. Filter is the file that deals with sorting slider (Altitude and speed). Distance is the file that is used to calculate the distance between two given geographics coordinates, using some math. It also calculates the aggregated distance when the user opens the bookmark section of the page. In terms of data handling, all the lists excluding the static one and the ones that are used when filtering are in states. This is because there are no need to store static array in states, and for filtering, we only used the filtered list once after the user clicks on "filter" (which also allows the "default" sorting option, which is what is held in the state initially when the data is fetched). Everything else uses state variables, and almost all function returns a React component, which in that case, uses props to pass down data. However, when there are no react component being returned, with some exception, I always used const instead of function. These exception functions are in sort.js, and this is because I am passing a function to the "sort()" function.

How the User Triggers State Changes

There are two sliders and one button at the top of the screen. Two sliders can filter the speed and altitude of the flight, and the page will show the airplanes that is flying in that range of altitude / speed. There are also a button that filters undefined airplanes (i.e. private jets / cesna etc.). The program will only show 1000 relevant flights in order to increase the page loading speed.

Aggregation

Aggregation system is implemented via "bookmark" feature. User can click on the icon right beside each flight's callsign to bookmark the flight. Then, the user can click on "Show Bookmarked flights" in order to view which flights have been bookmarked, and will show the aggregated total route distance if availabel. Since it is possible that certain flights do not have relevant information (i.e. private jets), it will also count for number of unknown route/callsigns.

Note About the Data Provided

Due to the larger volume of data being loaded, it is possible that sometimes it fails to load few data points and will show "Airline not found" even when the logo is present. However, this is rare, and it is also possible that the route of the flight is not plotted on the dataset I am using to map each callsigns to airline/routes. Also, the data presented on this website can be delayed by few minuites and might not be completely accurate. Lastly, it is possible that the dataset I am using to map the callsign to route is inaccurate (i.e. if the flights only operate on given route on specific day of the week, the data shown on the website might be completely wrong).

> **Warning**
>
> It seems to be that the routes in the U.S. are especially inaccurate. Please keep that in mind when utilizing this tool

Note About Filters

The definition of undefined airplane could either mean that the flight has missing callsign or that the flight's operator can not be identified (such as the private jets).