Airline Cheap Flight Search

Summary
Our project aims to help users find the cheapest airline flights by analyzing historical flight data and real-time airfare data. Finding a cheap flight can be difficult, especially with so many different airlines and pricing options. Using historical data about flight prices and airport locations, we hope to first inform users about the best time to book flights given a specific location for the cheapest price. Then, we will search for current flights and show the lowest available prices. Our goal is to provide a wide variety of pricing information to inform users about their flight choices. Users can be advised to buy tickets, or wait until a cheaper ticket opens up based on historical prices. Users will have the option to search flights, view data, and save flight information for future reference.

Description
The core functionality of our application is to find the most affordable flights for users based on their travel choices. Users can enter details such as departure location, destination, and travel dates. Our application will then search through our database of historical flight ticket prices, and will display information like average prices for a specific location or time, or overall common trends. Next, we will include an API that will gather information on current flights and display the least expensive options. Next, we hope to create a flight map to display the paths of possible flights. We will allow users to create accounts so that they can save the information they find on flights and view it later on. Users can update or remove their saved flights if they find better options.

The main problem that we want to solve is the difficulty of finding affordable plane tickets. These ticket prices often fluctuate, and can become very expensive. Airlines do not usually display the information needed to make an informed decision on what tickets to buy, so our goal is to provide a variety of information including current and past data so that users are not stuck with the most expensive option. By providing this data, we hope to help travelers find plane tickets without worrying as much about the cost.

Creative Component
A creative component that would enhance the functionality of our app is to optimize cost using layover flaps. We plan to implement a feature that allows you to find the cheapest flight to a destination on a specific day using layovers. This will effectively allow people to book multi-leg flights but intentionally stay in the layover city instead of continuing to the final ticketed destination, as layover flights can sometimes be cheaper than direct flights. We will be implementing Rapid API's flight scraper to analyze real-time airfare data. In doing so we will be able to pick apart layover flights to see if they create a cheaper trip to a desired destination. Our system will provide users with up-to-date pricing information so that they can make informed booking decisions and secure the best possible deals.

Usefulness

Our application is highly useful for any individual interested in exploring the world and traveling. They can enter a destination into our application and find the best time to travel there for the cheapest fare. A simple feature when finding this is filtering certain months out according to sections of the year when the user is available to travel. This involves a month filter on the table. A more complex feature is being able to search for flights to a specified location by including prices for flights with a layover, in which the user might be able to find a cheaper package by simply getting off the route. For example, if I want to visit Qatar, I may find 1 way tickets for Qatar for $1500, but another flight to India for $1200 with a stopover in Qatar. Using this information, I should purchase the trip to India and simply get off at my layover (not take the 2nd flight). There is a website called skiplagged that will find the best flights using the complex layover method mentioned earlier, though it does not provide the best time of year which our application provides using historical ticket sales.

Realness

Our first dataset (found from kaggle at https://www.kaggle.com/datasets/bhavikjikadara/us-airline-flight-routes-and-fares-1993-2024/data) contains historical information from airport to airport with fare prices. Specifically, we have over 245,000 tickets in this sample (cardinality of 245,955), ranging from 1993 until 2024, more than enough to depict average prices per time of year. With this information, we can find the time of year when tickets are cheapest, and return that time to the user depending on the specified filter. When downloading it, it is in csv format with a degree of 23, with columns representing the start airport code (ex. ORD), end airport code, fare, year, quarter, etc. The second dataset is very similar (found from kaggle at https://www.kaggle.com/datasets/dilwong/flightprices?resource=download), containing close to 6 million tickets (cardinality of 5999739). It maintains similar columns/attributes (degree 27) but includes extra flight packages that have layovers (which can help fulfill the complex/creative portion of the project). The specified time of the flight is much clearer, and allows us to access the exact date of the flight rather than the quarter found in the previous dataset. The file also downloads into CSV format.

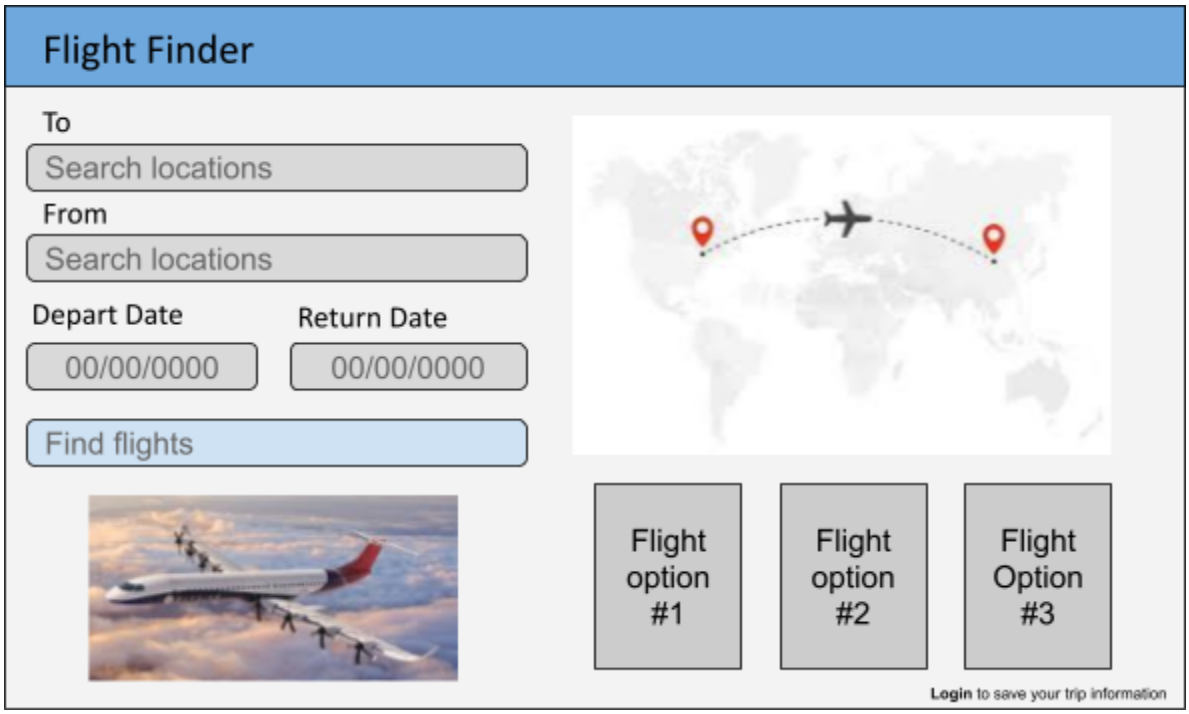
The third data source for our project is the Google Flight Scraper API which pulls real-time flights (from specific departing and arrival airports), prices, number of stops, and more info for a specified date. The information is returned in a structured JSON format which we can then add or use to update our tables with historical data.

Functionality

The goal of our application is to find the cheapest airline flights for users. This means that users should be able to provide all relevant information about their trip, so that we can output the best possible flights based on those details. We think users should have somewhere where they can input data about the location they want to visit, what airport they want to fly from, when they want to depart/return, and, optionally, more specific information about the flight such as the airline or budget. Users will input this data into a search field and our application will output

various flight options for them to choose from (ordered from cheapest to most expensive). These options would then direct users to the specific sites that they can purchase the tickets from. We also plan on having some sort of account system so that users can save the trip/flight information that our platform provides them. Users could then update or delete this information from their account whenever they choose.

Low-Fidelity UI Design



On the left side of our webpage, we have the search bars where users can enter in their flight information. This includes the starting destination, ending destination, departure date, and return date. The find flights button searches for flights that match the user input. Once options are returned, users can filter for more specific information like airline, cost, baggage, etc. On the right side of the webpage, our flight options will be displayed along with a map illustrating the flight path. The map is especially important for flights which are connecting so users can visualize where they will be and how layovers will affect their travel and/or cost.

Project Work Distribution

In our project, Neil will work on the front end while Michael, Cal, and Evan will work on the backend. We think that the backend will be a lot more time intensive since we will be dealing with data storage and data parsing. The frontend developer will be in charge of making sure that users have search bars where they can input their flight information. It is also necessary that a map is displayed with flight options as shown above. For the backend, Cal and Evan will work on the user account data. This will involve creating a database which stores specific users that have created an account though our platform as well as any trip information they saved. This task will also involve figuring out how to best store the trip information that users have saved and then updating or deleting data based on what users do through their accounts. Michael will

work on parsing through the flight data given by the API and figuring out how to return the best flight options given the user input. Most likely this will require parsing through the flight data and searching for data which matches the starting destination, ending destination, departure date, and return date and is ordered by ticket price.