Task

Given a JSON document representing a backend database, expose a search and aggregation endpoint through a new .NET API that serves data.

JSON File

The JSON file contains flight delay data using a dataset from CORGIS. Records contain statistics about flight delays grouped by airports and months.

Find the file here: https://think.cs.vt.edu/corgis/datasets/json/airlines/airlines.json

For example, this record shows that Seattle-Tacoma International Airport experienced 1,177 delays in August 2006, which were caused by aircraft being late.


```
{
    "Airport": {
        "Code": "SEA",
        "Name": "Seattle, WA: Seattle/Tacoma International"
    },
    "Time": {
        "Label": "2006/08",
        "Month": 8,
        "Month Name": "August",
        "Year": 2006
    },
    "Statistics": {
        "# of Delays": {
            "Carrier": 756,
            "Late Aircraft": 1108,
            "National Aviation System": 792,
            "Security": 49,
            "Weather": 64
        },
        "Carriers": {
            "Names": "American Airlines Inc.,Alaska Airlines Inc.,JetBlue Airways,Continental Air Lines Inc.,Delta Air Lines Inc.,Fr
            "Total": 14
        },
        "Flights": {
            "Cancelled": 63,
            "Delayed": 2771,
            "Diverted": 10,
            "On Time": 7443,
            "Total": 10287
        },
        "Minutes Delayed": {
            "Carrier": 36273,
            "Late Aircraft": 63026,
            "National Aviation System": 28023,
            "Security": 2033,
            "Total": 134014,
            "Weather": 4659
        }
    }
}
```


API Functionality

Please build an endpoint that adheres to the following structure, exposing

```
POST /api/airport-cancellations/search
```

With the following interfaces, expressed in TypeScript:

```
interface IAirportMonth { // directly from JSON file
    Airport: {
        code: string;
        name: string;
    };
    // ... other properties ...
}

interface IRequestBody {
    skip?: number;
    take?: number;

    query?: {
        airportCode?: string; // based on `airport.code`

        delayCount?: { // based on the sum of values in `Statistics['# of Delays']`
            minimum?: number;
            maximum?: number;
        };

        flightCount?: { // based on `statistics.Flights.Total`
            minimum?: number;
            maximum?: number;
        };

        since?: Date; // based on `Time.Label`
        until?: Date;
    }
}

interface IResponseBody {
    results: IAirportMonth[];

    totalCount: number; // ignoring pagination, how many results matched the query?
}
```

Details aren't particularly important, but the idea is to build out an API that exposes some form of search and pagination functionality.

Example Requests

1. The requester wants to retrieve a page of 10 rows from Sea-Tac:

   ```
   {
       "take": 10,

       "query": {
           "airportCode": "SEA"
       }
   }
   ```

2. The requester wants the second page from the above query:

   ```
   {
       "skip": 10,
       "take": 10,

       "query": {
           "airportCode": "SEA"
       }
   }
   ```

3. The requester wants only rows that have fewer than 10,000 flights:

```
{
    "take": 10,

    "query": {
        "flightCount": {
            "maximum": 10001
        }
    }
}
```

Design Objectives

Consider this an minimum viable product, or a proof of concept. Try to structure it with the knowledge that a UX designer or business user may want changes made--significant or otherwise.

Successful design would make use of sensible variable and function names, easily-followable application flow, and use of modularity when relevant.

Comments are welcome as ever, but try to write code that doesn't *need* excessive commenting to communicate its purpose.

Non-Objectives

There is no need to host the API anywhere; a local copy is fine.

It would be easy to spend days building out a highly-functional API for diving deep into this data, but that's not the point of this exercise. Invest time in code structure and modularity, and try to show off a little on the UI itself, but parts are bound to be duct-taped together.

> **Note!**
>
> Part of the task is making decisions about which pieces deserve effort and which don't!

Deliverables

Please deliver code that accomplishes the functionality objectives, or shows progress toward doing so. The code can be hosted in a public Git repository, or simply sent as a Zip file.

In addition, think about (and perhaps write down) bullet points of things you'd like to do with more time working on this mock project. Are there other UI elements you would add, changes you'd request of the API, or even changes to the code you deliver, if this were your full-time commitment?