Project 3 Proposal

I. **Team Name: J Cubed**

II. **Team Members: Jack Wang, Jan Torruellas, Jason Li**

III. **Project Title: J Cubed Flights™**

1. **Problem:** What problem are we trying to solve? [0.25 point]
   - Find the fastest flight route from one airport to another.
2. **Motivation:** Why is this a problem? [0.25 point]
   - People want to get to their destination as quickly as possible.
3. **Features**: When do we know that we have solved the problem? [0.25 point]
   - We can have two types of tests:
     1. Test Type 1: Small set of test cases that we find the proper flight times from google searching. If the resulting flight returned by the program is +/- an acceptable time difference, then the test case will be considered passed.
     2. Test Type 2: Large batch of randomly generated start and end destinations. This is to ensure that our algorithms are able to find a flight path between a large number of random airports.
4. **Data**: (Public data set we will be using and the link to the public data set) **or** (Schema of randomly generated data - i.e. what are the different columns in our dataset and the respective data types) [0.25 point]
   - The flight data we will be using is from the following link:
     1. https://www.kaggle.com/datasets/vishwanathmuthuraman/domestic-flight-data
     2. It contains over 200k different US domestic flights, each with multiple data attributes, so we will be well above the 100k data points requirement
   - We also have a dataset of US airport abbreviations with their corresponding Airport Full Name, the link is here:
     1. https://www.kaggle.com/datasets/aravindram11/list-of-us-airports
5. **Tools:** Programming languages or any tools/frameworks we will be using [0.25 point]
   - Programming language: C++
   - Tools: CLion IDE, GitHub Desktop (for version control)
   - Frameworks: None planned as of right now
6. **Visuals:** Wireframes/Sketches of the interface or the menu driven program [0.25 points]
   - US map displaying the selected start and end airports, as well as the 3 fastest paths to reach the destination.
   - We will display the results as follows:
     1. A -> B -> C (Average total time: ….)
     2. A->C (Average total time: ….)
     3. A -> D -> C (Average total time: ….)
7. **Strategy**: Preliminary *algorithms* or *data structures* you may want to implement and how would you represent the data [0.25 points]
   - **Data structures**:
     1. Graph as data structure to store flight data and airport names
        a. Graph Nodes:

        - Will be objects of an Airport Class
        - Stores the Airport abbreviations (**string**)
        - Each node will have multiple Directed Edge children:
            - Will be objects of a DirectedEdge Class
            - Between any two graph nodes there will be 2 directed edges (one in each direction)
            - Store direction of flight (two **string** values, departure and destination Airport abbreviation)
            - total of all inserted flights in this path (**int**, in minutes)
            - number inserted flights in this path, (**int**)
            - average flight time, which is found by the flight time total divided by number of flights (**int**, rounded down to the nearest minute/flight)
    2. HashTable to store key-value pairs of the Airline Codes with their corresponding Airport Name. These will be stored as **string** key/value pairs.
- **Algorithms**: For comparisons, we will use std::chrono to find the elapsed execution time to find the 3 fastest paths.
    1. Dijkstra's algorithm - for searching for the shortest (time-wise) path.
        a. When calculating flight time, if the algorithm needs to pass through multiple airports, we will add a reasonable penalty amount of time (TBD) to represent layover time between connecting flights. Layover times can vary anywhere from under an hour to several days, so we will be debating this penalty time for a bit.
    2. Breadth First Search - also for searching for the shortest (time-wise) path

8. **Distribution of Responsibility and Roles:** Who is responsible for what? [0.25 points]
    - Jason: create the Algorithm functions
    - Jan: create the Graph Nodes & Graph Object
    - Jack: Parse the CSV files

IV. References

- https://www.kaggle.com
- https://en.wikipedia.org/wiki/Graph_(abstract_data_type)
- https://en.wikipedia.org/wiki/Hash_table
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- https://en.wikipedia.org/wiki/Breadth-first_search
- Data Structures and Algorithm Analysis in C++