DEVELOPMENT LOG:

November 14:

- First all team-member meeting on Discord at 2PM PST
- Team Contract: started, finished, and uploaded to our GitHub repository
- Project Goals: talked about potential implementations and ideas
- Current Idea: using Stanford SNAP, we can create a network of frequently purchased Amazon items to recommend to the user

November 17:

- Second meeting started at 6PM PST with all members
- We finalize the team contract
    - We noticed that the SNAP Stanford data set that we initially wanted to use didn't have sufficient metadata so the nodes and edges were codes but we didn't know what those codes meant.
    - Given this, we pivoted to choosing a project about finding the optimal path from a starting airport to an ending airport.

November 27:

- Third meeting started at 8PM PST with all members
- Assigned roles to each member
- We created a general timeline of when to finish initial tasks
    - Reading and translating the data into a graph / Data Set Parser by Dec 1
    - Find GUI library by Dec 1
    - The two algorithms and the graph class is dependent on the Data Set Parser so due date is TBA

December 1:

- We worked on parsing the data from OpenFlights into a graph
    - We created the Graph class
    - Created the constructor that reads from a routes file and an airplane file
    - Create necessary maps: one maps airports to latitude and longitude, another maps airport to all outgoing Edges (Edges contain start/dest airport and airline code)
- Next steps:
    - Create MakeFile
    - Add functions to graph class to create vertices and edges from the maps we created today
    - Find some GUI library for the graphical visualization

December 2 - afternoon:

- We build additional features in the graph class that supports creation on vertices and edges for the graph
- We are in the process of creating the MakeFile
- Also found potential methods to do the graphical output: CS225 pixel class, mapnik, a map API etc.

December 2 - night:

- The graph implementation is created and we successfully parse and store data into a graph structure
- Function created to calculate longitude and latitude data and store it as weights in the graph data structure
- Functions that printed airports and their latitude longitude as well as flight route information was created.
- main.cpp was created.
- Makefile was created.

December 3:

- Created Getters to publicly interact with the graph. This will be used in the algorithms as well as the graphical output.

December 7:

- Visualizer is created to output the graph structure onto the world map
- Visualizer class takes in a Graph in the constructor and uses the graph structure to output all of the airports and routes
- Two helper functions, markAirport and drawConnection were created to help mark the airport onto the world map and draw a line between the two airports on the world map
- printProjection method is created to output all of the routes and airports onto the world map. This method calls upon the two helper functions described above to print all the routes. To avoid the map being completely filled up with routes, it is advisable to use not more than 6000 routes.

December 8:

- Completed processing the data so there were no bugs such at the latitude and longitude being 0 when they were not found
- Completed the BFS traversal algorithm which returns true or false based on whether the destination airport is reachable from the start of airport
- Updated makefile to accommodate tests by creating an executable called “test” which runs all tests

December 9:

- Created tests for graphical output. Tests varied from smaller to larger inputs of airplanes and routes. Also, we tested edge cases where the slope of an edge connecting two airports was technically infinite (vertical line) which proved buggy at first, but was rectified in a conditional statement in the code.
- Made progress on Dijkstra’s algorithm to find the shortest path between the start and end airport
    - This achieves the main goal of the project and although took some time to debug successfully outputs the shortest path
- Completed tests for Dijkstra’s algorithm
- Completed the README for our project

December 10:

- Worked on and completed the RESULTS section of our project which is uploaded to GitHub
    - We decided to also include the outputs of print functions of the graph as it is a good way to understand the structure of the group. Though this was not one of the required algorithms.
- Created an output for the BFS to verify and display the usage of the BFS algorithm
- Worked on creating a google presentation for the final project presentation

December 11:

- Completed final testing
    - BFS testing
    - Graph structure testing
        - If hashmaps that represented the graph structure were initialized correctly
- Team meeting on zoom to conduct the final presentation
- Gave final presentation to TA