The Results report concludes the final outcome of the project between Hrishikesh Deshmukh, Sabar Nimmagadda, Lloyd Quadros, and Bobby Wang, as well as any discoveries made in the process.

Seemingly ages ago, our project began with a shared team contract between the four of us. This contract was essential to the development of our group, and allowed us to function well together to create an operational final end result. Although outlining the method and frequency of communication between our group seemed excessive at first, we came to realize how important it was to implement the details of inter-group communication early in the work process. By committing to a weekly group meeting using Zoom's software and frequent discussions through text and email throughout the last few weeks, we were able to effectively plan and achieve our goals. Additionally, by laying out expectations for collaboration and a fair distribution of the workload - and consequently following those expectations - we were able to optimize our efficiency, total productivity, and quality of our final project. Each individual in our group was able to contribute to various components of the project, focusing on their distinct strengths rather than trying to help on every single deliverable. As a result, one of our greatest successes was having overall commitment between group members remain roughly equal while simultaneously maintaining diligence, effort, and enthusiasm.

To reiterate our initial project goals, we wanted to load the OpenFlights dataset into a directed graph, implement the Breadth First Search for our traversal algorithm, implement Dijkstra's algorithm for our covered option, implement the Landmark Path algorithm for our complex option, and effectively construct a test bench for our codebase. As a group, we were able to accomplish all of those project goals and have our code run dynamically on EWS. We only employed publicly installed libraries in tandem with our team's codebase to create our functional final product. Overall, we can say that our team's code successfully accomplishes our stated goals.

After committing our contracts and project goals, we began the journey of completing the final project by uploading data from the OpenFlights dataset. Although it was an immense set of data, we were able to upload it to our repository without any hiccups. We then proceeded to create a basic structure for our graphs as well as airport files to identify individual airports. This mostly involved parsing through the dataset and understanding what objects we were dealing with. The next significant hurdle was creating a makefile that would compile our files and allow us to run our program. It was quite a challenge, but we discovered the template made available by one of the TAs as well as example makefiles included in previous MPs. Our group applied this code and was able to successfully compile our own code. The following week we thoroughly examined our graph implementation. It became increasingly difficult to patch bugs in our code without knowing if our fix would generate additional issues. We determined that producing a test bench and preemptively testing our functions was in fact the most practical method of discerning bugs in our code and ensuring the functionality of our methods. After this realization, we resolved numerous complications. Our graph was able to parse data properly, add vertices and edges for that data, and automatically forge an edge between any newly added vertices. The created nodes/vertices store the latitude, longitude, and name of the airports. To expand on that idea, we assumed in our graph construction that we are flying Cessna airplanes with a maximum flying distance of 800 miles. As a result, if airports were within 800 miles of each other, the edge weight was the distance between them, and zero if the airports were outside the 800 mile range of each other. These edges were stored in an adjacencyMatrix within our graph data structure. Another challenge during the workflow process was that individuals in our group would work on the codebase and commit to the repository at the same time, which negatively affected our performance. However, we discovered that communicating at a higher frequency led to far fewer commit collisions. By continuing this high level of personal communication for the duration of the final project, our team members were able to individually contribute to the deliverables without impeding the progress of others.

In our third and final full week of work time, our group implemented the meat and bones of the project - the Breadth First Search algorithm, Dijkstra's algorithm, and the Landmark Path algorithm. While these various algorithms were problematic due to their complexity, we identified the value of employing digital resources and pseudocode to comprehend the minute details of each implementation; any convoluted snippets of code were properly cited as well. This discovery led to the facilitation of writing the actual code for these components of the project. We completed the final touches of our project by accurately documenting, organizing, and optimizing our codebase; providing usage and build documentation in the Readme; and writing auxiliary components such as the final portion of the weekly Development log. As we had hoped, our group enjoyed a productive, engaging, and meaningful final project for CS 225. Thank you!