1. Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).
    a. We were able to implement the primary features that we wanted to include in our original project proposal, but one thing we didn't include is the most common reason for the delay. While we included a calculator for average delay, we didn't calculate the probability for a delay given the destination and airline. In addition, while users can search for the average delay and best recommended airline based on the origin and destination, we decided to calculate it based on the airport instead of using latitude and longitude. A feature that we added that that wasn't in the original proposal allowed users to search for reviews.

2. Discuss what you think your application achieved or failed to achieve regarding its usefulness.
    a. The application is able to show users the average delay times from each origin airport to each destination airport for every airline. This gives users an idea about which airline would best suit their needs. However, since our application is only based on historical data, it may not be very accurate for the status quo. It is reasonable to assume that since the time this data was taken, airlines have improved their efficiency.

3. Discuss if you changed the schema or source of the data for your application
    a. We did not change the schema or the source for our data. We are using the original dataset and the design we had come up with.

4. Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

    We made several changes to our original ER diagram and table implementation. One of these changes was to update the attribute names of the tables so that the foreign key of one table would have the same name as the primary key of the table using it. This made writing queries easier, as having the same name simplified the process of writing the "WHERE" clause.

    Additionally, we removed all of our many-to-many tables, which had made our diagram look cluttered. Instead, we used foreign keys to directly traverse two tables and find matching relationships. We made this change for two reasons: 1) The size of our database was limited, so it was not necessary to use an additional relationship table for matching; and 2) The functions using each many-to-many table were not used frequently - only 1-2 times - which made any loss in matching speed negligible.

    In my opinion, when dealing with a much larger database in practical use, it would be more suitable to use many-to-many relationship tables. This is because it would significantly increase matching speed.

5. Discuss what functionalities you added or removed. Why?
    a. One functionality in our original proposal that we did not include is informing users of the reason for delay. Originally, in addition to the average delay time and probability of delay, we had also intended to include the reason for the delay. However we did not include this functionality because upon further review of the dataset, we realized that most of the data points did not have a reason for delay listed. One feature that we added was the ability to search for reviews based on an airline. We thought that if users are able to leave reviews for airlines, then they should also be able to sort the reviews based on an airline.

6. Explain how you think your advanced database programs complement your application.
   Our application's functionalities are strengthened beyond our initial expectations thanks to our advanced database programs. The stored procedures we use leverage two advanced queries which provide the user with the best airline options based on the average delay times between the user's selected origin and destination. The system is intelligent and helps users improve their flight scheduling. Furthermore, with the trigger in place to record every update on the review table, we can trace every change made, protecting us against malicious attacks and enabling easy recovery of important data.

7. Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.
   a. Kelsi: One technical challenge that our team encountered was hosting our application on GCP. While the app was working locally, there were many issues with getting it hosted on GCP. These issues created internal server errors and also messed with the entire functionality of our application. We fixed this through watching videos on YouTube where people hosted Flask Apps on GCP. We also Googled many of our errors until the app worked. One piece of advice I would give is to make sure all versions of the packages used support all the functionalities and are compatible with each other (we have some issues with SQLAlchemy).
   b. Ashley: A technical challenge that we encountered involved rerouting and clearing the form after the user input data and submitting it. We were incorrectly using the redirect function, which either gave us an internal server error or simply just didn't work. We were able to fix this issue after searching how to clear forms with Flask. Overall, this was my first time working with Flask and GCP, so I would recommend watching tutorials on both and how to host Flask apps on GCP, which our group also had some trouble with.
   c. Ishq: One technical challenge that I encountered was connecting the database hosted on the GCP platform to our local application. Although some notes were posted regarding how to connect the database, some of the commands did not work properly. I would recommend that teams watch YouTube videos on how to connect a GCP hosted database to a Flask App, as that is something that many others have done before.
   d. Zhuohao: One technical challenge I encountered was ensuring the proper use of quotation marks within queries when connecting the backend with the database. This required careful attention and debugging. Additionally, it is important to note that the recommended database library is not advanced, which limits the usability of the cursor method.

8. Are there other things that changed comparing the final application with the original proposal?
   There were very few changes between the final application and the original proposal. The only significant change was that we initially planned to implement a machine learning algorithm to help users predict delay times based on weather reports. However, we ultimately decided not to pursue this feature due to difficulties with the interface between the Python program and the VM in GCP.

9. Describe future work that you think, other than the interface, that the application can improve on
   a. We can improve our application by making it more personalized to each user. For example, we can take region into account (so we can factor in weather or natural disasters) to explain possible delays. We can also allow users to input a flight number (if they have a flight) and calculate delay based on the user's own flight. Another thing we could do is create more interactions within our database and group airports that are near each other together to recommend airports that may have less of a delay to a destination.

10. Describe the final division of labor and how well you managed teamwork.
    a. Kelsi and Ashley worked most with setting up the frontend and creating that while Ishq and Zhuohao worked with integrating the databases and connecting it to the Flask App. All of us worked together to set up the databases and to plan the schemas/plan the application. We did this through meeting up in person to work through bug fixes or to discuss initial ideas. We also had Discord calls where we shared a screen and worked together.