Team 31 Project Reflection Report

The final implementation of our project mostly aligned with our original proposal. The users are able to query our extensive database of scraped flight prices, contribute their own data, and view helpful aggregations to help contextualize prices they may be encountering in their own searches. One modification we made from our original proposal can be found in the relationship between user-contributed data and the existing preloaded data in the app. While we planned to focus on attribute-based slices of data for comparison between profiles, our final project is more focused on aggregating the user-provided data against the whole of the available data based on the location details of the flight instead of the attributes of the user.

Our app achieves the main objectives we intended: users can explore an extensive set of records of historic flight prices, contribute their own flight prices, and perform some basic comparisons to help contextualize prices found in the market. While our application did not focus on trends by user profile, we augmented the overall search functionality by allowing the user to quantify how many flights they have contributed to the database, and providing immediate feedback in the form of price averages which immediately update to account for a new flight price provided by a user.

We implemented our database schema as described in Stage 2 of the project without significant changes, and the source of the scraped data used in our application was as proposed. The relationships between our tables and schema match our ER diagram submitted in Stage 2. We did modify the data types used by some columns, most notably using `SERIAL` columns for attributes like `leg_id`, `route_id`, and `profile_id` to ensure the uniqueness constraints our application needed in these areas of the data model. The data type of these columns ended up being essential when we implemented our trigger and transaction, allowing for more seamless insert commands than the original `INT` columns planned for these attributes.

In addition to the features mentioned allowing users to contribute their own pricing data to the database, we also implemented functionality requiring the user to correctly enter their own username and password to successfully add a price associated with their profile. This helps ensure accuracy when we update the user with the count of prices they have contributed.

The advanced database components of our implementation contribute to the application in two key areas: useful functionality and database controls. In terms of controls, we used a transaction to ensure that insertions to multiple tables are all successful before being committed after a user provides flight data. This ensures our database relationships are maintained, and that the user has a consistent experience with our application. Further, our advanced queries give the user added functionality by allowing for viewing the database records and their own contributions in aggregate.

Here are some technical challenges that the team encountered, coupled with general development advice to future teams working on a similar project:

1. Due to some of our initial ideas around profile usage, the data model contains some specific relationships which are correctly implemented, but could be a challenge for others to maintain without the same understanding of our assumptions. For example, the `profile` and `human_user` tables are both updated with each new profile creation, and we had to line up our application endpoints to match the actual table relationship. This

type of subclass relationship required careful consideration when used in our trigger, but could be helpful downstream for future improvements.

2. Coordinating the development of a full stack application. There will be work that needs to be done on the frontend and work that needs to be done on the backend. Making sure there's a process where both teams can develop and test in parallel is super helpful. Whether that's through containerizing these components, etc. Making sure the development pipeline is efficient will make the project more fun than annoying. You don't want to be in a position where you're constantly waiting on something to be created in the backend in order for the frontend to succeed or vice versa.
3. Making sure that everyone has a thorough and equal understanding of the schema of the database you're working on. It's not good to just have one person who knows the whole layout of the database, so be sure to have enough team meetings so that everyone can confidently describe and manipulate the database if necessary.
4. Defining the entities in a way that minimizes repeated information - we choose to make each pricing instance correspond to a roundtrip which in turn consists of 2 leg_id that give the actual characteristics of the flight in question. In reality a user could submit information about a trip with more than 2 leg_ids (multiple stopovers etc.) but we choose to keep it simpler for this implementation

Again, our application largely matched the original proposal, except for the shift towards flight-based aggregation instead of profile-based aggregation. Future work on this application could include expanding the functionality around how users and their contributed flight prices can interact with the database. One potential path could be to lean into our proposal direction and leverage more profile characteristics through aggregation to understand changes in price trends between users with different ages, genders, or web browsing habits. Another possibility would be to allow the user greater control over their flight contributions; being able to modify a submitted price, or remove one entirely could help the application feel more interactive for users.

The team worked cooperatively, in line with our planned division of labor in the original project proposal. Jackie lead frontend development to create the user interface of our application, Shawn developed the backend to support all of the functionality our application uses, Aditya had greatest familiarity with our initial dataset and ensured our implementation matched design, and Chris drove documentation and staged releases to keep the project moving for each checkpoint. Our team was extremely communicative, working cooperatively to succeed in all areas of the project.