CS/IDS 121 Final Project Reflection

Tia Abraham
tabraham@caltech.edu

March 17, 2025

Part L0. Introduction

Part 1

What application did you design and implement? What was the motivation for your application?

The **Flight Carbon Footprint Tracker** is a database-backed application designed to estimate and log carbon emissions from air travel. The motivation for this application stems from the growing need for individuals and organizations to track and reduce their environmental impact, particularly in the aviation sector. The application allows users to calculate emissions for specific flights, store trip records, and view aggregated emissions data. The dataset used includes real-world aircraft models, airport locations, and flight routes sourced from OpenFlights, ensuring accurate and realistic estimations.

Part 2

What was the dataset and rationale behind finding your dataset?

The primary dataset used in this application consists of:

- Aircrafts, countries, airports, and flight routes sourced from OpenFlights.
- The dataset was cleaned to remove null values, ensuring primary key constraints and referential integrity.
- Additional synthetic data for users and trips was generated based on real-world flight patterns and past personal trips.

Part 3

Who are the users for your application?

Client Interface User(s)

The client interface is designed for individual travelers who want to:

- Calculate estimated CO_2 emissions for flights between airports.
- Store trip records for future reference.
- View past trips and analyze emissions data by month, year, or country.
- Manage their accounts, including password changes.

Admin Interface User(s)

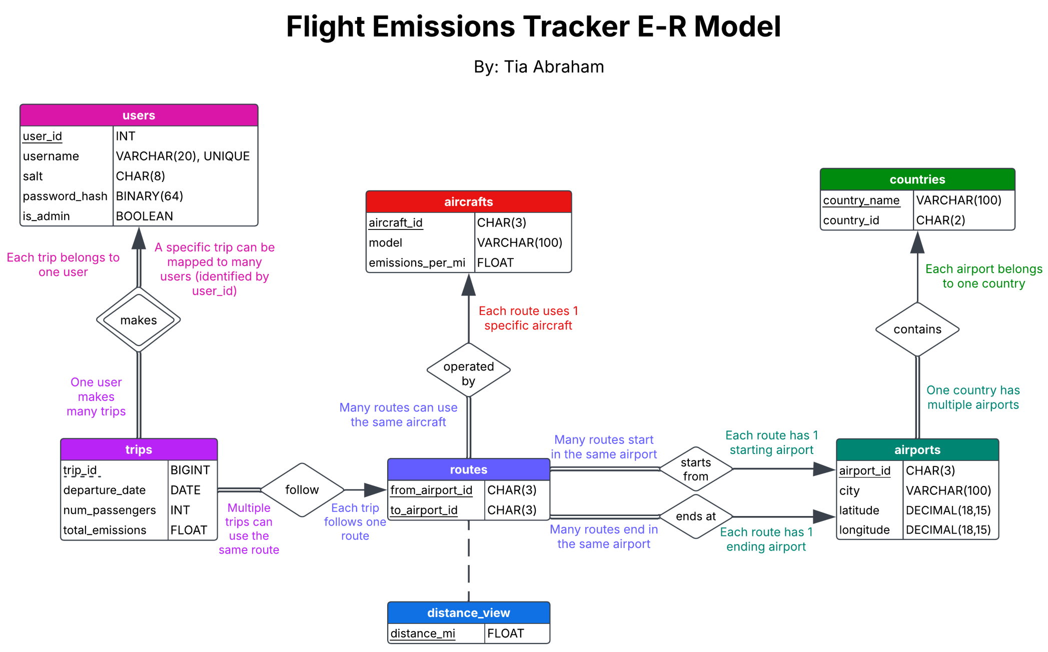
The admin interface is designed for system administrators responsible for maintaining the database and user management. Admin functionalities include:

- Resetting user passwords.
- Granting admin privileges to specific users.
- Updating aircraft emissions data.
- Adding new flight routes to the database.
- Managing database integrity by ensuring data consistency.

Part A. ER Diagrams and Updated UI Diagram

Part 1

In this answer, you should provide a full ER diagram of your system.



Flight Emissions Tracker E-R Model
By: Tia Abraham
users
user_id INT
username VARCHAR(20), UNIQUE
salt CHAR(8)
password_hash BINARY(64)
is_admin BOOLEAN
Each trip belongs to one user
A specific trip can be mapped to many users (identified by user_id)
makes
One user makes many trips
trips
trip_id BIGINT
departure_date DATE
num_passengers INT
total_emissions FLOAT
follow
Multiple trips can use the same route
Each trip follows one route
aircrafts
aircraft_id CHAR(3)
model VARCHAR(100)
emissions_per_mi FLOAT
Each route uses 1 specific aircraft
operated by
Many routes can use the same aircraft
routes
from_airport_id CHAR(3)
to_airport_id CHAR(3)
distance_view
distance_mi FLOAT
Many routes start in the same airport
starts from
Each route has 1 starting airport
ends at
Many routes end in the same airport
Each route has 1 ending airport
countries
country_name VARCHAR(100)
country_id CHAR(2)
Each airport belongs to one country
contains
One country has multiple airports
airports
airport_id CHAR(3)
city VARCHAR(100)
latitude DECIMAL(18,15)
longitude DECIMAL(18,15)

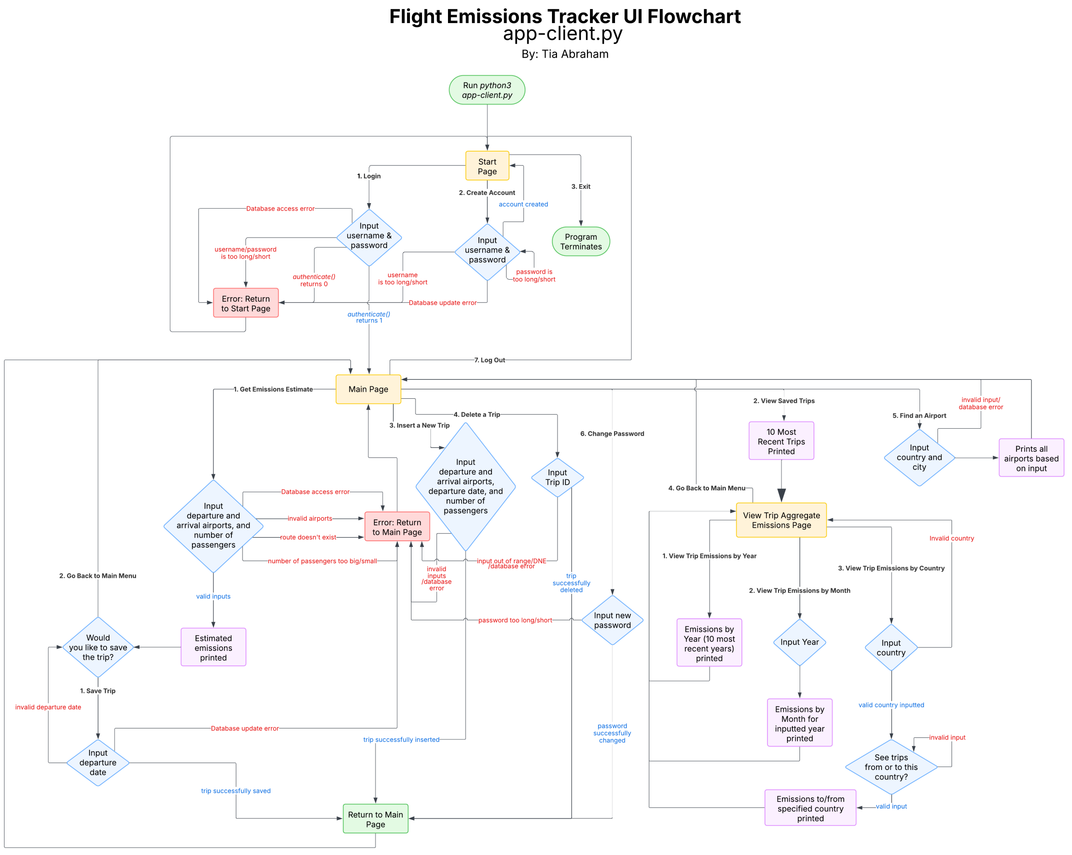
Links to Previous Iterations:

- **Milestone:** `https://drive.google.com/file/d/17WUA-XvGG7ve32iOrkzBLS1knFH8AWSH/view?usp=drive_link`
- **Proposal:** `https://drive.google.com/file/d/1pG7kfxnRb-FcDu8aCnoiFtYjY_F4uQsG/view?usp=drive_link`

Part 2

Also include any updated UI flowcharts from your proposal/milestone.

User Flowchart



Flight Emissions Tracker UI Flowchart
app-client.py
By: Tia Abraham
Run python3 app-client.py
Start Page
1. Login
2. Create Account
3. Exit
Program Terminates
Input username & password
Input username & password
account created
Database access error
username/password is too long/short
authenticate() returns 0
username is too long/short
password is too long/short
Database update error
Error: Return to Start Page
authenticate() returns 1
7. Log Out
Main Page
1. Get Emissions Estimate
3. Insert a New Trip
4. Delete a Trip
6. Change Password
2. View Saved Trips
5. Find an Airport
Input departure and arrival airports, and number of passengers
Database access error
invalid airports
route doesn't exist
number of passengers too big/small
Error: Return to Main Page
valid inputs
Estimated emissions printed
Would you like to save the trip?
2. Go Back to Main Menu
1. Save Trip
invalid departure date
Input departure date
Database update error
trip successfully saved
Input departure and arrival airports, departure date, and number of passengers
invalid inputs /database error
input out of range/DNE /database error
trip successfully inserted
Input Trip ID
trip successfully deleted
Input new password
password too long/short
password successfully changed
Return to Main Page
10 Most Recent Trips Printed
4. Go Back to Main Menu
View Trip Aggregate Emissions Page
1. View Trip Emissions by Year
2. View Trip Emissions by Month
3. View Trip Emissions by Country
Emissions by Year (10 most recent years) printed
Input Year
Emissions by Month for inputted year printed
Input country
Invalid country
valid country inputted
See trips from or to this country?
invalid input
valid input
Emissions to/from specified country printed
Input country and city
invalid input/ database error
Prints all airports based on input

Links to Previous Iterations of User Flowchart

- **Milestone:** `https://drive.google.com/file/d/1CqU-EMyTkt3d1Y6JEj1YIshQVfT2wYF4/view?usp=drive_link`
- **Proposal:** `https://drive.google.com/file/d/1CvzoVU_jbjGgOkdVkhlsjfVDnXSE5phF/view?usp=drive_link`

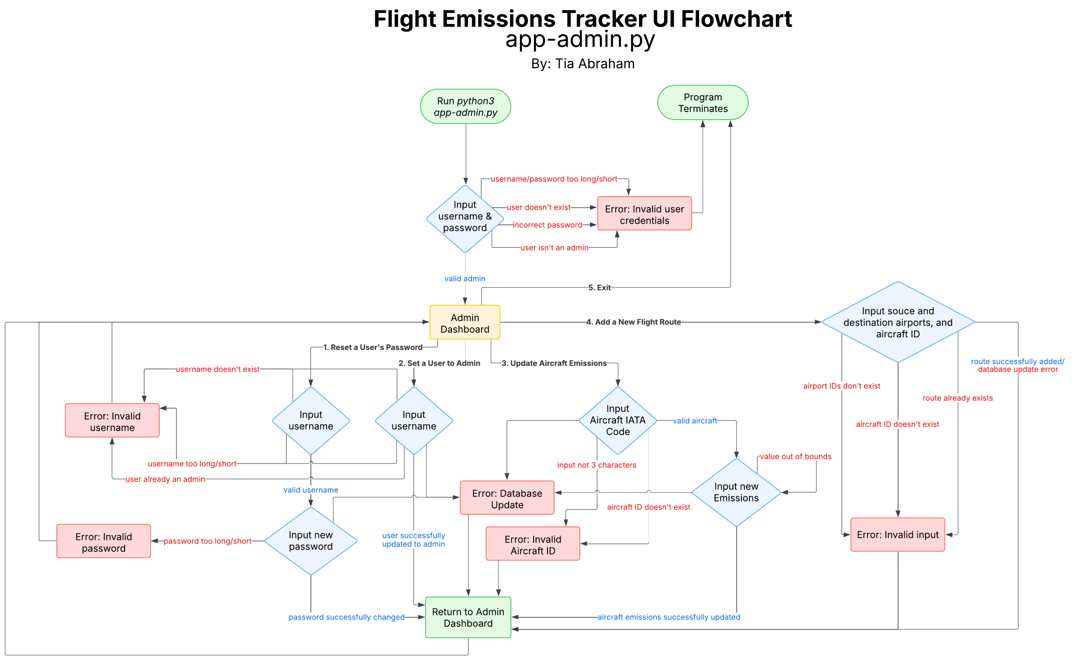
Summary of Changes

In the initial proposal flowchart, the flow was relatively minimal and focused primarily on basic functionality: logging in, registering, and calculating emissions for a trip. The core path allowed users to input trip details (origin, destination, and passengers), calculate emissions, and optionally save the trip—forming the foundation of the app.

By the time of the milestone submission, while still keeping the layout lightweight and hand-drawn, I began distinguishing between valid/invalid inputs and modularizing actions (like emissions calculations and trip saving) into separate logical branches. This was also the first appearance of database procedure references (e.g., ‘sp_add_user’, ‘calculate_trip_emissions’).

In the final flowchart, the application’s scope and specificity were fully realized. Implemented in Lucidchart, it reflects the complete suite of user-facing features, including seven main menu options: emissions estimates, trip insertion, deletion, viewing, airport lookup, password changes, and logout. Each pathway now handles edge cases such as invalid inputs and database access errors. The logic is more rigorous, offering detailed conditionals and user feedback. The flow is also visually structured into modular sections for clarity. This final version transforms what started as a simple utility into a robust, interactive user-facing application.

Admin Flowchart


Flight Emissions Tracker UI Flowchart
app-admin.py
By: Tia Abraham
Run *python3 app-admin.py*
Program Terminates
Input username & password
username/password too long/short
user doesn't exist
incorrect password
user isn't an admin
Error: Invalid user credentials
valid admin
5. Exit
Admin Dashboard
4. Add a New Flight Route
Input souce and destination airports, and aircraft ID
route successfully added/ database update error
airport IDs don't exist
route already exists
aircraft ID doesn't exist
Error: Invalid input
1. Reset a User's Password
2. Set a User to Admin
3. Update Aircraft Emissions
username doesn't exist
Error: Invalid username
Input username
Input username
username too long/short
user already an admin
valid username
Input Aircraft IATA Code
valid aircraft
input not 3 characters
value out of bounds
Input new Emissions
Error: Database Update
aircraft ID doesn't exist
Error: Invalid Aircraft ID
Error: Invalid password
password too long/short
Input new password
user successfully updated to admin
password successfully changed
Return to Admin Dashboard
aircraft emissions successfully updated

No previous iteration of admin flowchart.

Part B. DDL (Indexes)

Here, describe your process for choosing your index(es) and show that it is used by at least one query, which speeds up the performance of the same query on a version of the same table without that index.

Index

To optimize search queries for airports located in a specific country where the city name starts with a given letter, I created the following composite index:

```
CREATE INDEX idx_airports_country_city
ON airports (country_name, city(1));
```

This index is designed to efficiently filter records based on both the country and the first letter of the city name. The first part of the index, `country_name`, ensures quick filtering by country, while `city(1)` acts as a prefix index that speeds up queries using `LIKE 'X%'`, where X is any starting letter.

Justification and Performance Testing Results

Before adding the index, MySQL's query execution plan indicated that filtering by `city LIKE 'X%'` required a full scan of all cities within the given country. This resulted in unnecessary row scans, increasing query execution time. After applying the index, the database optimizer was able to use `idx_airports_country_city` to rapidly retrieve only relevant city names, significantly reducing query time.

To validate the effectiveness of the index, I measured the execution time of the following queries before and after indexing. The performance improvement is summarized in the `PROFILES` table:

```
+----------+------------+---------------------------------------------------------+
| Query_ID | Duration   | Query                                                   |
+----------+------------+---------------------------------------------------------+
|        3 | 0.00690000 | SELECT DISTINCT city FROM airports                      |
|          |            | WHERE country_name = 'United States' AND city LIKE 'S%' |
|        4 | 0.00415000 | SELECT DISTINCT city FROM airports                      |
|          |            | WHERE country_name = 'Australia' AND city LIKE 'S%'     |
|        5 | 0.05793200 | CREATE INDEX idx_airports_country_city                  |
|          |            | ON airports (country_name, city(1))                     |
|        6 | 0.00231600 | SELECT DISTINCT city FROM airports                      |
|          |            | WHERE country_name = 'United States' AND city LIKE 'S%' |
|        7 | 0.00206400 | SELECT DISTINCT city FROM airports                      |
|          |            | WHERE country_name = 'Australia' AND city LIKE 'S%'     |
+----------+------------+---------------------------------------------------------+
```

Using `SHOW PROFILING`, I confirmed that MySQL was leveraging the newly created index; the reduction in execution time demonstrates the efficiency of indexing strategies for text-based searches in large datasets.

Part G. Relational Algebra

Minimum of 3 non-trivial queries.

Total Emissions by Month for a Given Year

This query computes the total emissions for each month in a given year ('2025') for a specific user ('user_id = 1'). The result is grouped by month and sorted in ascending order.

SQL Query:

```
SELECT MONTH(departure_date) AS month,
       SUM(total_emissions) AS total_emissions
FROM trips
WHERE user_id = '1'
  AND YEAR(departure_date) = '2025'
GROUP BY MONTH(departure_date)
ORDER BY MONTH(departure_date);
```

Relational Algebra Expression:

$$\Pi_{\text{MONTH}(departure\_date),\ \textbf{sum}(total\_emissions)\ \textbf{as}\ \ total\_emissions}\left(\sigma_{\text{user\_id}=1\wedge\text{YEAR}(departure\_date)=2025}(\text{trips})\right)$$

View Trips Departing from a Specific Country

This query retrieves all trips for a specific user ('user_id = 1') that depart from airports located in the 'United States'. It returns the departure airport, arrival airport, departure date, number of passengers, and total emissions.

SQL Query:

```
SELECT t.from_airport_id, a1.city AS from_city,
       t.to_airport_id, a2.city AS to_city,
       t.departure_date, t.num_passengers, t.total_emissions
FROM trips t
JOIN airports a1 ON t.from_airport_id = a1.airport_id
JOIN airports a2 ON t.to_airport_id = a2.airport_id
WHERE t.user_id = '1'
  AND a1.country_name = 'United States'
ORDER BY t.departure_date DESC;
```

Relational Algebra Expression:

$$\Pi_{\text{from\_airport\_id, from\_city, to\_airport\_id, to\_city, departure\_date, num\_passengers, total\_emissions}}($$

$$\sigma_{\text{user\_id}=1 \wedge \text{country\_name}='United States' \wedge \text{t.from\_airport\_id}=\text{a1.airport\_id} \wedge \text{t.to\_airport\_id}=\text{a2.airport\_id}}(\text{trips} \times \text{airports} \times \text{airports}))$$

Find All Flights Using a Specific Aircraft Model

This original query retrieves all trips that were flown using an aircraft with model name ‘Boeing 737’. It joins the ‘trips’ table with the ‘routes’ table (which links flights to aircrafts) and the ‘aircraft’ table (which contains aircraft model names).

SQL Query:

```
SELECT t.from_airport_id, t.to_airport_id, t.departure_date,
       t.num_passengers, t.total_emissions, a.model
FROM trips t
JOIN routes r ON t.from_airport_id = r.from_airport_id
               AND t.to_airport_id = r.to_airport_id
JOIN aircrafts a ON r.aircraft_id = a.aircraft_id
WHERE a.model = 'Boeing 737-800' AND t.user_id = 1;
```

Relational Algebra Expression:

$$\Pi_{\text{from\_airport\_id, to\_airport\_id, departure\_date, num\_passengers, total\_emissions, model}}($$

$$\sigma_{\text{model}='Boeing737' \wedge \text{t.from\_airport\_id}=\text{r.from\_airport\_id} \wedge \text{t.to\_airport\_id}=\text{r.to\_airport\_id} \wedge \text{r.aircraft\_id}=\text{a.aircraft\_id}}($$

$$\text{trips} \times \text{routes} \times \text{aircrafts}))$$

Update Aircraft Emissions

This query updates the emissions per mile value for an aircraft with ‘aircraft_id = 100’ to ‘0.16’. The relational algebra expression follows the conventional form where we modify ‘*aircrafts*’ and assign it back to itself.

SQL Query:

```
UPDATE aircrafts SET emissions_per_mi = '0.16' WHERE aircraft_id = '100';
```

Relational Algebra Expression:

$$\begin{aligned} \text{aircrafts} \leftarrow & \left(\sigma_{\text{aircraft\_id} \neq '100'}(\text{aircrafts})\right) \\ & \cup \left(\Pi_{\text{aircraft\_id, model, emissions\_per\_mi} \to (aircraft_id, model, 0.16)}(\sigma_{\text{aircraft\_id}='100'}(\text{aircrafts}))\right) \end{aligned}$$

Part L1. Written Reflection Responses

Challenges and Limitations

List any problems (at least one) that came up in the design and implementation of your database/application (minimum 2-3 sentences). Include anything that may help us distinguish between conceptual issues or untested code vs. scoping/time constraints.

One of the most difficult challenges I encountered was ensuring referential integrity and proper flow between the SQL backend and the Python client, particularly when designing multi-column foreign keys like in the `routes` and `trips` tables. Debugging procedures like `sp_add_trip` proved surprisingly tricky because certain foreign key constraints weren’t immediately visible when errors arose, forcing me to write helper debugging procedures. Another conceptual issue came up when attempting to make a composite primary key increment properly by user, which I ultimately resolved by switching to a combined `user_id, trip_id` approach. Most of these difficulties were due to the natural complexity of integrating a full-stack app in a short amount of time rather than from untested code.

Future work

If you are particularly eager for a certain application (have your own start-up in mind?), it is easy to over-scope a final project, especially one that isn't a term-long project. You can list any stretch goals you might have "if you had the time" which staff can help give feedback on prioritizing (2-3 sentences).

If I had more time, I would implement more calculations on what flight emissions mean (i.e. converting the value to the amount of trees you'd have to plant to offset the emissions). I would also love to visualize emissions data over time using a lightweight frontend or charting library to generate CO_2 graphs for users. Finally, I'd consider expanding the dataset to modeern-day flight routes, since many routes are missing due to the dataset being dated from 2014.

Self-Evaluation

What is your expected grade for the Final Project (out of 100)? Justify what you think are the strongest points, especially pointing to demonstrated improvement in areas that may have had lower scores in assignments throughout the term. Also provide any notes here on areas that could be improved (and what you would do differently next time).

My expected grade for the Final Project is **100**. I believe the strongest points of my project include its completeness, real-world application, and clean integration of backend procedures with a functional Python interface. I implemented a wide range of SQL features (functions, procedures, joins, aggregates, etc.) while keeping the UI intuitive and robust with solid error handling. Compared to earlier assignments, I've improved significantly in database normalization, relational algebra reasoning, and connecting backend design with frontend execution. If I could do one thing differently, I might have started the UI flowchart and ER model drafts even earlier so I could iterate more gradually alongside implementation.

Expected final course grade: 100

Requested Feedback

To what extent would you like feedback on your project vs. a grade? We will prioritize feedback to students who read through it, particularly as a potential portfolio project to use in future job/grad school applications (if there are areas you would like to improve or learn more about, let us know!), but will otherwise keep written feedback brief.

I would love detailed feedback on the project, especially since I plan to use it as a portfolio piece for future job/internship applications. Feedback on how I structured the schema and any potential weaknesses in normalization or scalability would be particularly helpful, as well as any other user actions you would like implemented in the application.

Sharing Work

We like to share student examples to highlight diversity of projects and showcase the work done by students each term. Would you be open to any of the following being shared?

Anything: Y

I would prefer to be credited by name.

Other Comments

We would appreciate your feedback on what you found most helpful, and what you might find helpful to change.

I really appreciated how this project tied together the material from the entire term—from schema design and relational algebra to views, indexing, and full application integration. The weekly feedback and milestones were especially helpful in keeping me on track. The only thing I might suggest is teaching the E-R diagramming unit earlier, since it seemed like a very crucial component of the project and happened to be the last assignment of the course. Overall, this was one of the most rewarding classes I've taken.