SkyTrack Project Report

Compared to our original Stage 1 proposal, the core idea of SkyTrack, tracking flight statuses and managing flight/passenger information, remained intact. However, some features, such as in-depth heatmaps by time of day and complex connection delay alerts, were simplified. We prioritized building a functional and stable core web application for tracking flights, focusing more on usability for passengers rather than building out multiple layers of data analytics immediately.

SkyTrack successfully allows passengers to track their flights. The login system was implemented and we also implemented booking, flight search, and status updates. We also implemented some more advanced data visualizations, such as heatmaps and airline rankings. The system meets the goal of providing an accessible and user-friendly way for users to view relevant flight information and manage flight statuses.

Originally, we planned to use a dataset with real-time worldwide data. However, for practical reasons, we instead went with a manageable sample size dataset of US domestic flights from 2019-2023 from the US Department of Transportation and Bureau of Transportation Statistics. The schema stayed relatively close to our design, but we adjusted some attributes to streamline development. Our original UML diagram and relational schema from Stage 2 were mostly preserved but we made some changes. For example the FlightDate, DeptTime, and FlightNumber in the Booking Table were made to be foreign keys instead of embedding redundant flight data. We also made Booking a relation instead of an entity by itself. We also split out the flight status as its own entity to maintain 3NF, and simplified the Status table's connection. We integrated the Airline table into the Flight table to simplify the design. If we were to redesign, a more suitable model might normalize Passenger details into a separate schema for loyalty program integration (frequent flyers), anticipating future expansion.

We removed the functionality of pinning flights to the UI, users can fetch their flights manually through the flight fetch process.

The more advanced database parts of the project are related to the application part of flight delay heat map and airline performance graph. It complements the application as it gives more insight and a graphical representation to flight tracking. We also utilized indexes to improve query times for flight lookups and Explain Analyze to test and optimize query costs.

There were some technical challenges we faced throughout the project. For example, implementing the User/Admin roles on the frontend was tricky. Future teams should implement role-based web pages so admins may have their functionalities in a dedicated space. Building responsive frontend components for the dashboards was hard due to limited time. Future teams could implement more graphical components to better inform users about flight information. Handling edge cases like cancelled flights required additional backend logic. A future team should pre-validate flights before confirming a booking to the database. Setting up Google Cloud SQL connections was time-consuming due to network issues. Future teams should continue building up the cloud infrastructure and utilize other GCP applications to improve the application.

For future work beyond interface improvements, the application can be improved by adding support to multi-leg flights by implementing flight chaining and connection delay alerts, ensuring that passengers are warned if delays may impact connecting flights. Additionally, the application could add a frequent flyer management system that tracks user flight histories and provides personalized analytics. The application could also, as discussed before, integrate real-time data for live flight updates, rather than relying on static CSV data, which will significantly enhance the system's accuracy and relevance.

For the final division of labor we split the work differently from the initial proposal. Ashwin Jain focused on the frontend application and connecting it to the backend GCP SQL Database. Arshiya Gupta focused on advanced database features (Transactions, stored procedures, Triggers and constraints), as well as creative components (data querying and plotting of bar graphs as well as making it interactive). Jiaxi Huang focused on database normalization, advanced SQL queries, indexing, and demo presentation. Rafael Wersom focused on database design (UML, Relational Schema, DDL commands, etc.) and writing/editing the stage reports. However, there were multiple instances that teammates helped each other by delegating tasks to finish the stages on time. In the end, all teammates performed adequately and there were no cases of a teammate not doing their part.

Project Video:

https://drive.google.com/file/d/1PczsZ37FwGbJzjDbZsxe0_8Ta7TcIWKE/view