Team Contributions

Keith Wang — UI/Routes & Validation

- Focus: front-end templates, Flask routes, and business validation
    - Staff flight views & filters: Default "next 30 days," period toggle (Default/Current/Future/Past), date-range/IATA/city filters, dynamic query construction.
    - Create Flight: form + airplane ownership check; after creating, show the next 30 days list.
    - Add Airplane: form/route restricted to the staff of the airline.
    - Customer search & purchase: show only purchasable flights; on purchase, enforce name-on-card must match account name and card number must be digits; block purchase of departed/cancelled flights; safe ticket_ID generation.
    - Reports/Ratings UI: templates, tables, navigation, and user feedback via flash().
    - Docs/Runbook: authored and maintained README_file.md and USE_CASES.md (setup, dependencies, and use-case→SQL mapping).

Eric Lin — Database & Flask Back-End

- Focus: schema, data seeding, and core Flask back-end
    - Schema & data: designed Customer/Staff/Airplane/Flight/Ticket/Rating, foreign/compound keys and CHECK/ENUM constraints; initialization and seed data.
    - Auth & session (Flask):
        - Login/registration routes (per project, password checked via md5), role separation Customer/Staff;
        - Session variables and login guards (e.g., as_customer() / as_staff()).
    - Business routes (Flask):
        - Change Status: on-time/delayed/cancelled with authorization and idempotent feedback;
        - Staff reports: last_month / last_year / range using DATE_SUB/DATE_FORMAT;
        - Ratings: submit/view ratings and comments, compute flight averages;
        - Customer – My Flights: list purchased flights with time filters;
        - Shared helpers: build_staff_query() for dynamic SQL, unified fetchone/fetchall, and error handling.
    - Performance & safety: added indexes for frequent predicates; all queries use prepared statements; centralized input sanitization and flash error messages.

Joint Work — Integration & Testing

- End-to-end scenarios:
    - Staff: view -> add airplane -> create flight -> change status -> view ratings/reports.
    - Customer: search -> purchase -> my flights -> rate/comment.
- Edge cases & consistency: time zones/date formats, empty states, idempotent actions, role checks, expired sessions.
- Code review: ensured prepared statements, role-based access control, default 30-day display, reports, and basic XSS hygiene are fully met.

Summary

- Keith: Flask routes + templates + validation (UI/flow heavy).
- Eric: database design + authentication/session + reports/status/ratings routes (back-end heavy).

- Together: delivered all Part 3 required use cases and Additional Requirements.