Architecture Documentation

This document outlines the technical architecture of the AeroTrack application, including the database schema and a breakdown of the application's structure.

1. System Overview

The application is a monolithic web service built using the **Flask** framework in Python. It follows a simple, single-file structure (app.py) for all backend logic and routing. It uses **Flask-SQLAlchemy** as an ORM to interact with a **SQLite** database. The frontend is rendered server-side using HTML templates with embedded CSS.

2. Database Schema

The application uses four main tables to store data: User, Passenger, Flight, and Shipment.

User Table

Stores user credentials for authentication.

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY	Unique identifier for the user.
username	VARCHAR(100)	UNIQUE, NOT NULL	The user's login name.
password_hash	VARCHAR(128)	NOT NULL	The securely hashed password.

Flight Table

Stores all flight information.

Column	Type	Constraints	Description
flight_no	VARCHAR(10)	PRIMARY KEY	Unique identifier for the flight (e.g., AI101).
frm	VARCHAR(50)	NOT NULL	The departure

			airport code.
too	VARCHAR(50)	NOT NULL	The arrival airport code.
dep_date	VARCHAR(20)		The departure date.
dep_time	VARCHAR(20)		The departure time.
arr_date	VARCHAR(20)		The arrival date.
arr_time	VARCHAR(20)		The arrival time.
status	ENUM	NOT NULL, DEFAULT	The current status of the flight (e.g., Scheduled, Delayed).

Passenger Table

Stores information about individual passengers.

Column	Type	Constraints	Description
pid	INTEGER	PRIMARY KEY	Unique identifier for the passenger.
name	VARCHAR(100)	NOT NULL	The passenger's full name.
age	INTEGER	NOT NULL	The passenger's age.
sex	VARCHAR(10)	NOT NULL	The passenger's gender.
address	VARCHAR(200)		The passenger's address.
contact	VARCHAR(50)		The passenger's contact number.

email	VARCHAR(100)		The passenger's email address.

Shipment Table

Stores information about cargo, linked to a specific flight.

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY	Unique identifier for the shipment.
contents	VARCHAR(200)	NOT NULL	A description of the shipment's contents.
weight_kg	FLOAT	NOT NULL	The weight of the shipment in kilograms.
category	VARCHAR(50)	NOT NULL	The shipment category (e.g., General, Fragile).
is_insured	BOOLEAN	DEFAULT False	Whether the shipment is insured.
flight_no	VARCHAR(10)	FOREIGN KEY	Links the shipment to a flight in the Flight table.
cost_per_kg	FLOAT	NOT NULL, DEFAULT	The cost to ship per kilogram.
handling_fee	FLOAT	NOT NULL, DEFAULT	A flat handling fee for the shipment.

3. Class & Module Breakdown

The application logic is centralized in **app.py**.

Models (within app.py)

- **User**: Manages user data and is used by Flask-Bcrypt for password verification.
- **Passenger**: Represents a passenger with personal details.
- **Flight**: Represents a flight and includes an Enum for FlightStatus to ensure data consistency.
- **Shipment**: Represents a piece of cargo. It is linked to a Flight via a foreign key and contains a @property method total_cost to calculate the shipping cost dynamically. This demonstrates the "calculated field" requirement.

Routes (within app.py)

- **Authentication Routes** (/login, /register, /logout): Handle user session management.
- **Data Management Routes** (/flights, /passengers, /shipments): Display lists of data and handle creation (POST requests from forms). These routes are protected by the @login_required decorator.
- **Edit Routes** (/edit_flight/<id>, etc.): Handle both displaying the edit form (GET request) and processing the updated data (POST request).
- **Delete Routes** (/delete_flight, etc.): Handle the deletion of records.