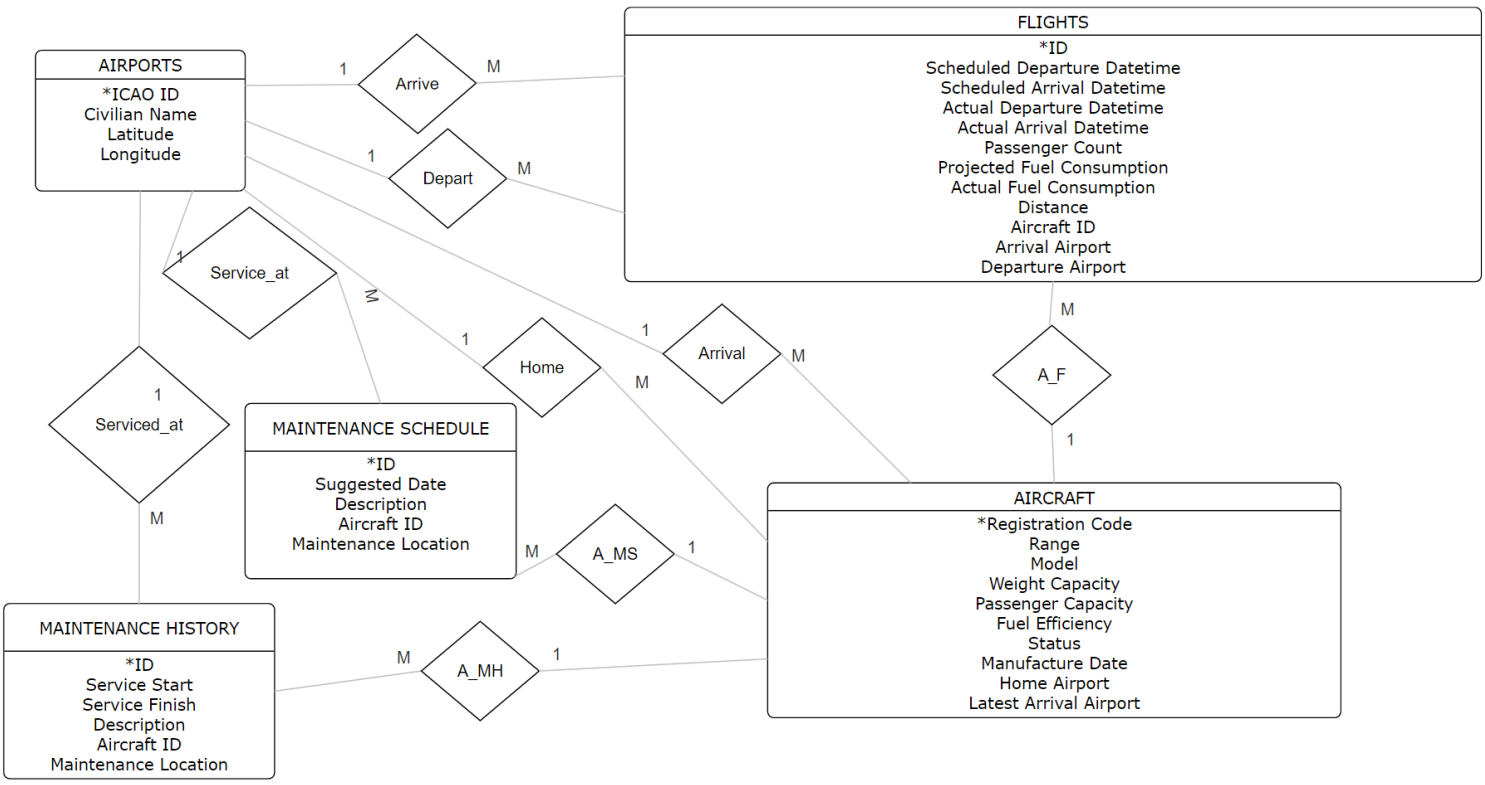
Rohith Velmurugan
Luke Roth

Problem Statement: In this project, we aim to develop a Flight Operations Database that stores and manages critical information related to a given airline's flight operations. The primary objective for the program is to provide a system for tracking flights, aircraft details, fuel consumption, and scheduling. By centralizing this data, the database will enable more efficient airline management, allowing staff and renters to easily access and analyze flight and aircraft information.

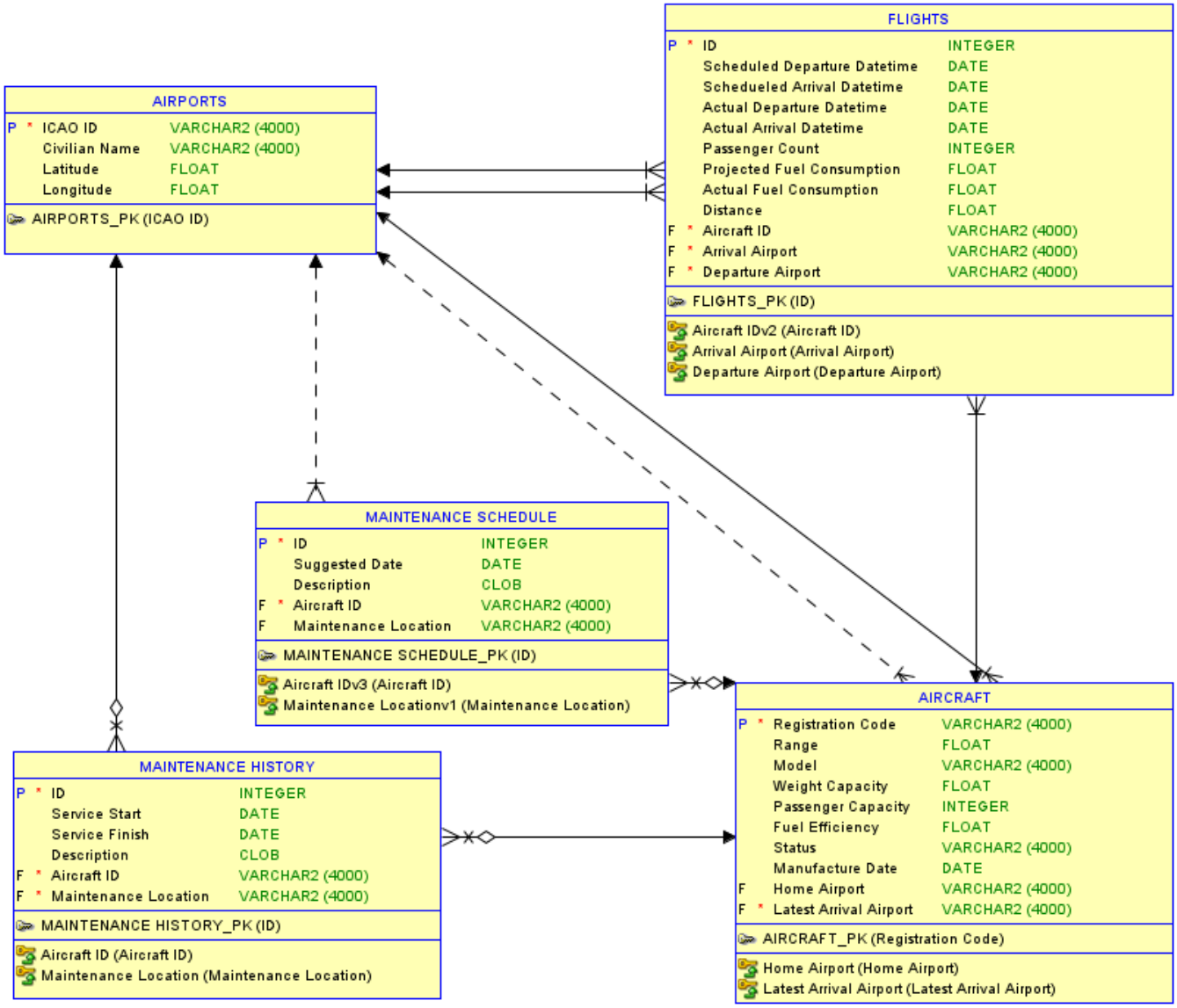
This system will track key metrics such as average distance traveled by flights, fuel consumption per route, and aircraft-specific information such as capacity, model, and maintenance history. Additionally, the system will include tools for managing flight schedules and ensuring no scheduling conflicts occur for the airline's fleet.

The database will also allow users to securely input, search, and update flight and aircraft data while providing analytics to help optimize airline operations. By the end of the semester, our goal is to deliver a fully functional computer application with a user-friendly interface to manage this data effectively.

Conceptual Design:



Logical Design:



Summary Table of Data Types:

Table	Attribute	Type	Constraint
AIRPORTS	ICAO ID	String	Primary Key
AIRPORTS	Civilian Name	String	
AIRPORTS	Latitude	Latitude	NOT NULL
AIRPORTS	Longitude	Longitude	NOT NULL
FLIGHTS	ID	Integer	Primary Key
FLIGHTS	Scheduled Departure Datetime	Datetime	
FLIGHTS	Scheduled Arrival Datetime	Datetime	
FLIGHTS	Actual Departure Datetime	Datetime	
FLIGHTS	Actual Arrival Datetime	Datetime	
FLIGHTS	Passenger Count	Integer	
FLIGHTS	Projected Fuel Consumption	Float	
FLIGHTS	Actual Fuel Consumption	Float	
FLIGHTS	Distance	Float	
FLIGHTS	Aircraft ID	String	Foreign Key
FLIGHTS	Arrival Airport	String	Foreign Key
FLIGHTS	Departure Airport	String	Foreign Key
MAINTENANCE SCHEDULE	ID	Integer	Primary Key
MAINTENANCE SCHEDULE	Suggested Date	Date	
MAINTENANCE SCHEDULE	Description	Text	

Summary Table of Data Types (Continued):

MAINTENANCE SCHEDULE	Aircraft ID	String	Foreign Key
MAINTENANCE SCHEDULE	Maintenance Location	String	Foreign Key
MAINTENANCE HISTORY	ID	Integer	Primary Key
MAINTENANCE HISTORY	Service Start	Date	
MAINTENANCE HISTORY	Service Finish	Date	
MAINTENANCE HISTORY	Description	Text	
MAINTENANCE HISTORY	Aircraft ID	String	Foreign Key
MAINTENANCE HISTORY	Maintenance Location	String	Foreign Key
AIRCRAFT	Registration Code	String	Primary Key
AIRCRAFT	Range	Float	
AIRCRAFT	Model	String	
AIRCRAFT	Weight Capacity	Float	
AIRCRAFT	Passenger Capacity	Integer	
AIRCRAFT	Fuel Efficiency	Float	
AIRCRAFT	Status	String	
AIRCRAFT	Manufacture Date	Date	
AIRCRAFT	Home Airport	String	Foreign Key
AIRCRAFT	Latest Arrival Airport	String	Foreign Key

Functional Requirements:

1) Calculate and display average flight distances
    a) Would need the range from the Aircraft entity set and then divide it by total number of flights calculated by code

2) Fuel consumption per flight
    a) Manual input of the fuel consumption per flight

3) Generate performance reports showing total distance traveled by specific aircraft or total fuel consumed over a given period, utilization percentage, suggested maintenance date, average passenger capacity utilized based on time in air, last serviced date, and flights per month over a 12 month period.
    a) Use the aircraft entity set as well as the maintenance history entity set to populate the data for the report. Total flights of a specific aircraft are parsed for a certain date range of 12 months. Parse through a specific aircrafts passenger capacity history

4) Assign aircraft to flights and manage flight schedules, ensuring no scheduling conflicts occur
    a) Use Flights entity set departure & arrival ICAO code property and the scheduled arrival and departure times on the same route at the same time. Flights scheduled departure time has to be after flight actual arrival time is not NULL.

5) Data export in common formats such as CSV or Excel
    a) Have the ability to download the maintenance history, flights, utilization history, of each specific aircraft. Need to utilize the entire maintenance history set.

6) Include regular database backups to ensure data safety and prevent data loss
    a) Configured MySQL to provide automatic backups of the database in regular time intervals.

High Level Pseudocode:

1) *Function AssignAircraftToFlight*
   // This function assigns an aircraft to a selected flight.
   // It accesses the FLIGHTS and AIRCRAFT tables.
   Input: aircraft ID, flight ID
   (1) Retrieve "Scheduled Departure Datetime" and "Scheduled Arrival Datetime" for the flight from the FLIGHTS table where flight ID matches.
   (2) Check the availability of the aircraft in the FLIGHTS table by ensuring no scheduling conflicts between "Scheduled Departure Datetime" and "Scheduled Arrival Datetime" for the selected aircraft
   (3) If no conflicts exist, update the LIGHTS table by assigning the aircraft ID to the flight
   (4) Display confirmation that the aircraft has been successfully assigned

2) *Function CalculateAverageFlightDistance*
   // This function calculates the average flight distance.
   // It accesses the FLIGHTS table.
   (1) Retrieve the total distance of all flights from the FLIGHTS table (sum the Distance attribute).
   (2) Retrieve the total number of flights from the "FLIGHTS" table (count entries in the "ID" column).
   (3) Calculate the average flight distance by dividing the total distance by the total number of flights.
   (4) Display the average flight distance.

3) *Function InsertMaintenanceRecord*
   // This function inserts a new maintenance record for an aircraft.
   // It accesses the MAINTENANCE HISTORY and AIRCRAFT tables.
   Input: aircraft ID, service start date, service finish date, maintenance description, maintenance location
   (1) Insert the new maintenance record into the MAINTENANCE HISTORY table with the provided details.
   (2) Update the AIRCRAFT table to reflect the new maintenance event
   (3) Display confirmation that the new maintenance record has been successfully added.

4) *Function DeleteFlightRecord*
   // This function deletes a flight record.
   // It accesses the FLIGHTS table.
   Input: flight ID
   (1) Check if the flight ID exists in the FLIGHTS table.

(2) If it exists, delete the flight record from the FLIGHTS table where ID matches.
(3) Display confirmation that the flight record has been successfully deleted.

5) *Function GeneratePerformanceReport*
// This function generates performance reports for an aircraft.
// It accesses the FLIGHTS, MAINTENANCE HISTORY, and AIRCRAFT tables.
Input: aircraft ID, start date, end date
(1) Retrieve flight data from the FLIGHTS table for the specified aircraft between the start and end dates
(2) Calculate the total distance traveled by summing "Distance" from the "FLIGHTS" table
(3) Calculate total fuel consumed by summing "Actual Fuel Consumption" from the "FLIGHTS" table
(4) Retrieve maintenance history from the MAINTENANCE HISTORY table for the specified aircraft
(5) Retrieve the last serviced date from the MAINTENANCE HISTORY table
(6) Calculate average passenger capacity by dividing "Passenger Count" by "Passenger Capacity" from the FLIGHTS and AIRCRAFT tables
(7) Generate a performance graph for the number of flights per month over the last 12 months using the FLIGHTS table
(8) Display the performance report

6) *Function ExportFlightData*
// This function exports flight data to a CSV or Excel file.
// It accesses the FLIGHTS table.
Input: CSV or Excel
(1) Retrieve all flight data from the FLIGHTS table
(2) Write the retrieved data into a file in the specified format: CSV or Excel
(3) Provide a download link for the exported file

7) *Function BackupDatabase*
// This function creates a backup of the entire database.
// It accesses all the tables in the database.
(1) Retrieve all data from all the tables
(2) Write the retrieved data to a backup file
(3) Store the backup file securely
(4) Display confirmation that the backup has been completed successfully

Installation Instructions:

Step 1: Have Python 3.11 or above installed from python.org

Step 2: Install MySQL Server and enable it to run as a windows service

Step 3: After cloning the repository, navigate to it's directory. Run the following commands to create and activate your virtual environment: *python -m venv .env .env/Scripts/activate*
Note: Your Powershell Execution Policy must be set to at least as permissive as remote-signed in order to run this properly on Windows.

Step 4: Install the required dependencies using: pip install -r requirements.txt

Step 5: Open initialize.py and verify:

host="localhost"
user="root"
passwd="admin" ← Update to match your MySQL root password

Step 6: Run the script to create the database and populate tables using the command:
python initialize.py

Step 7: Start the server using the command: *python run.py*

Step 8: To access the web application open a browser and head to this link: *http://127.0.0.1:5000*

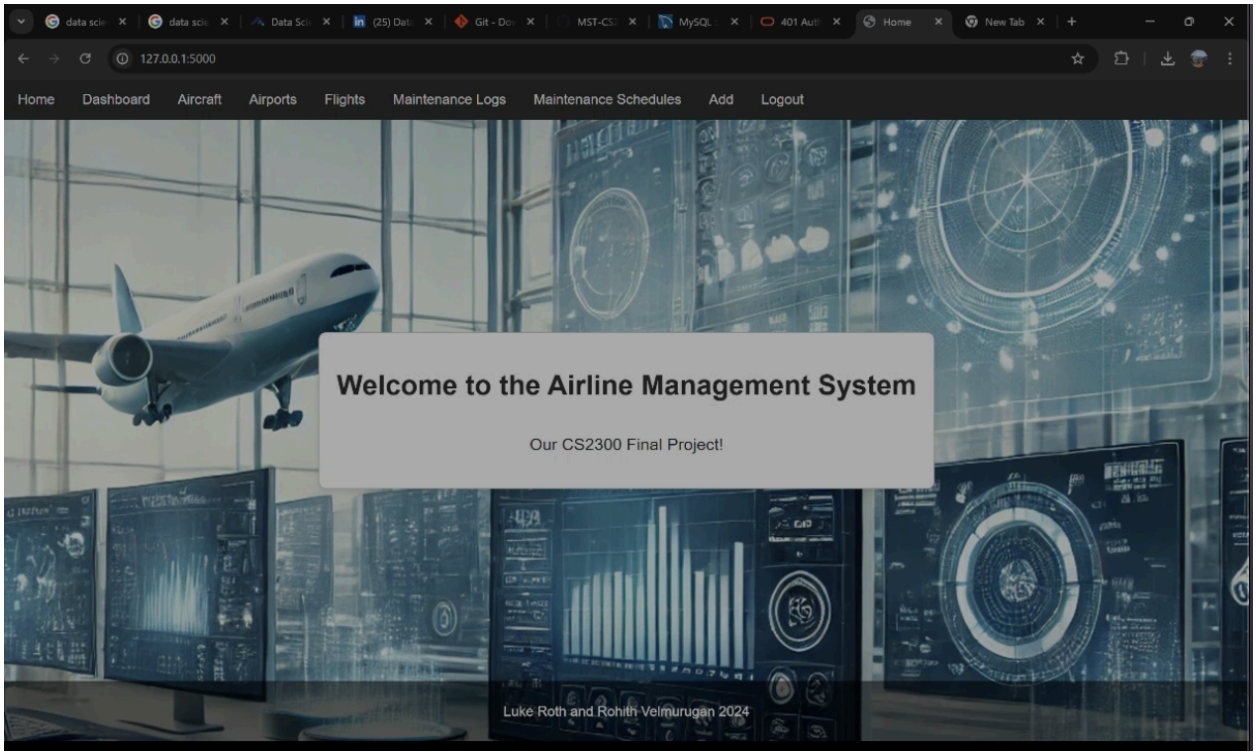
Troubleshooting:

- Make sure you are accessing the right directory
- Make sure the MySQL service is running
- Locate and start MySQL if it is not already running.
- Ensure the username and password in initialize.py match your MySQL credentials.
- Ensure all required Python packages are installed using *pip install -r requirements.txt*
- Verify Flask is running on the expected port http://127.0.0.1:5000
- Ensure no other application is using the same port

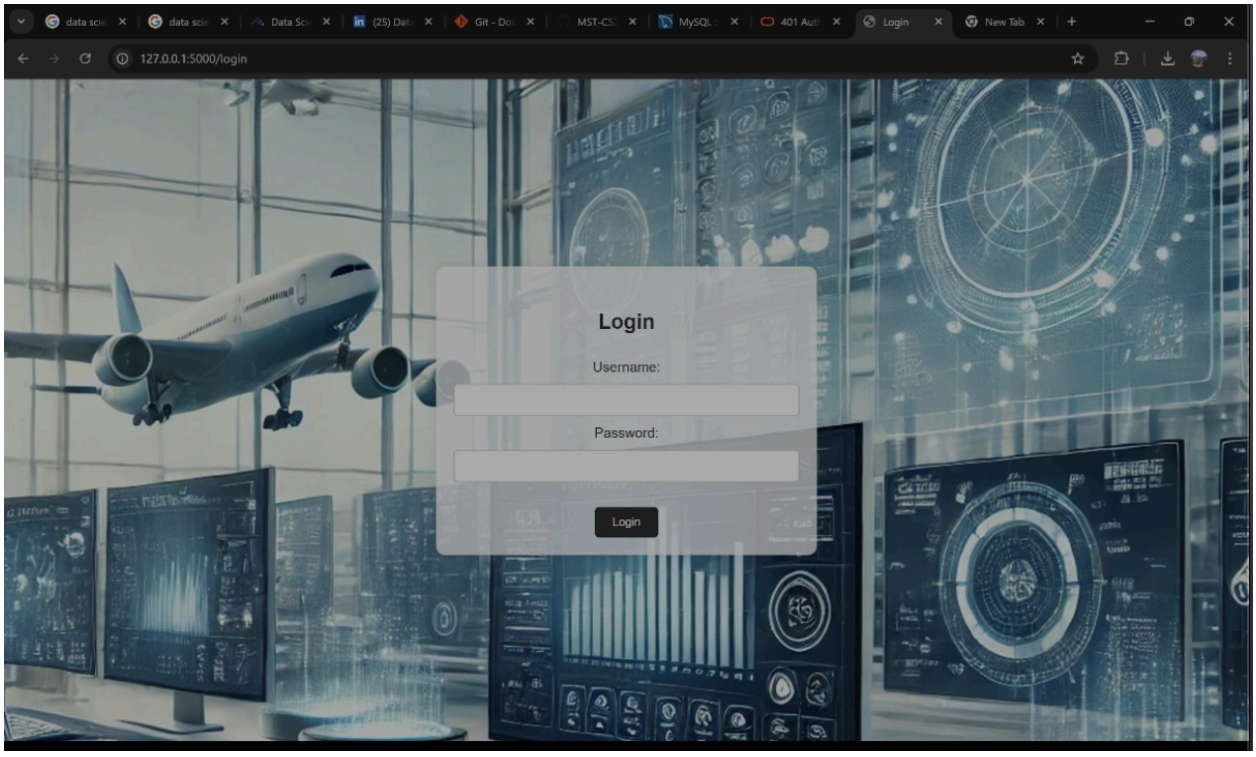
Dependencies:

- Python 3.11+
- Flask
- Mysql-connector-python

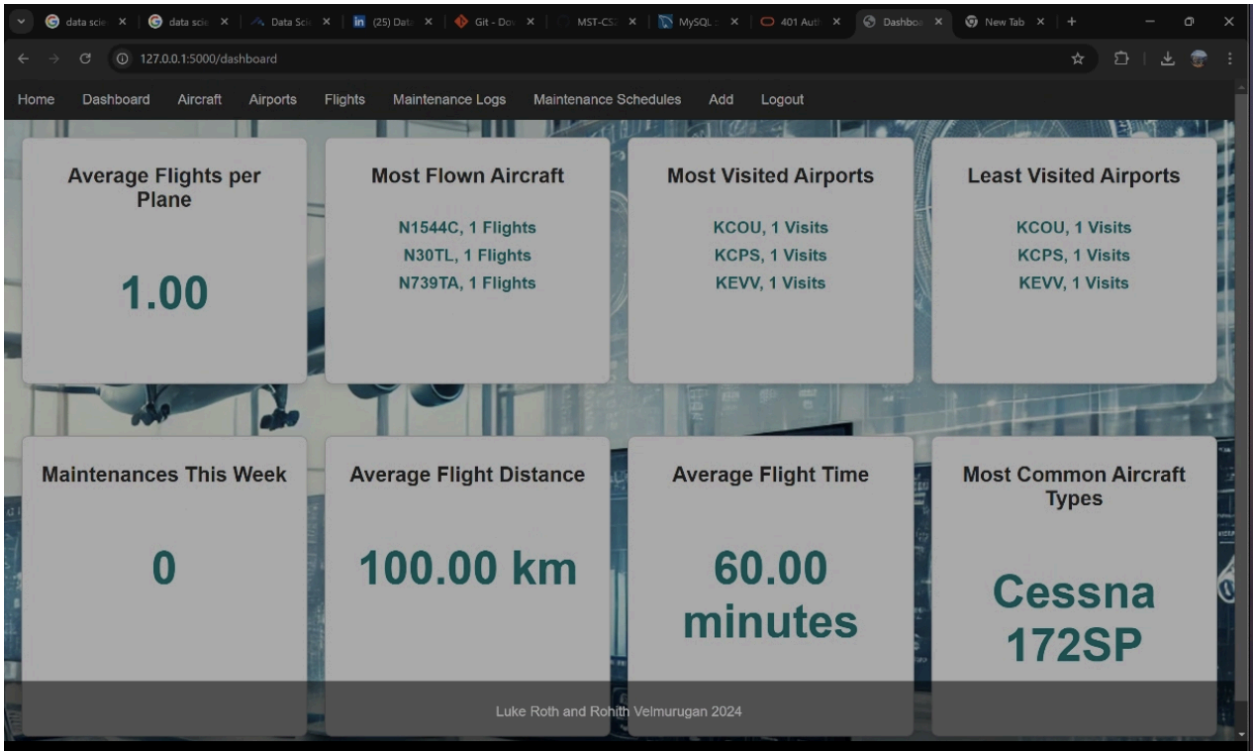
User Manual:



1. **Login: You initially begin logged in. If you press the Logout button, enter your credentials to access the system. Default credentials are:**
   Username: admin and Password: CS2300

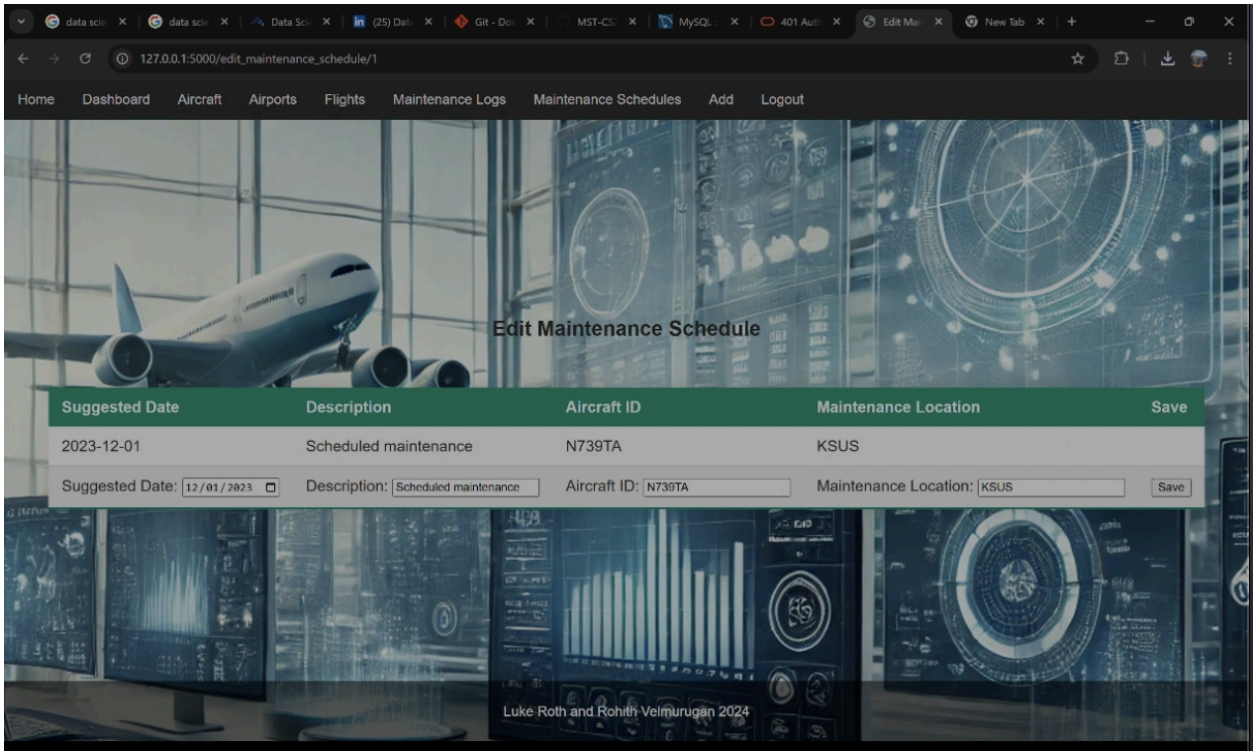


2. **Dashboard: Click on the Dashboard tab to view key metrics and statistics.**

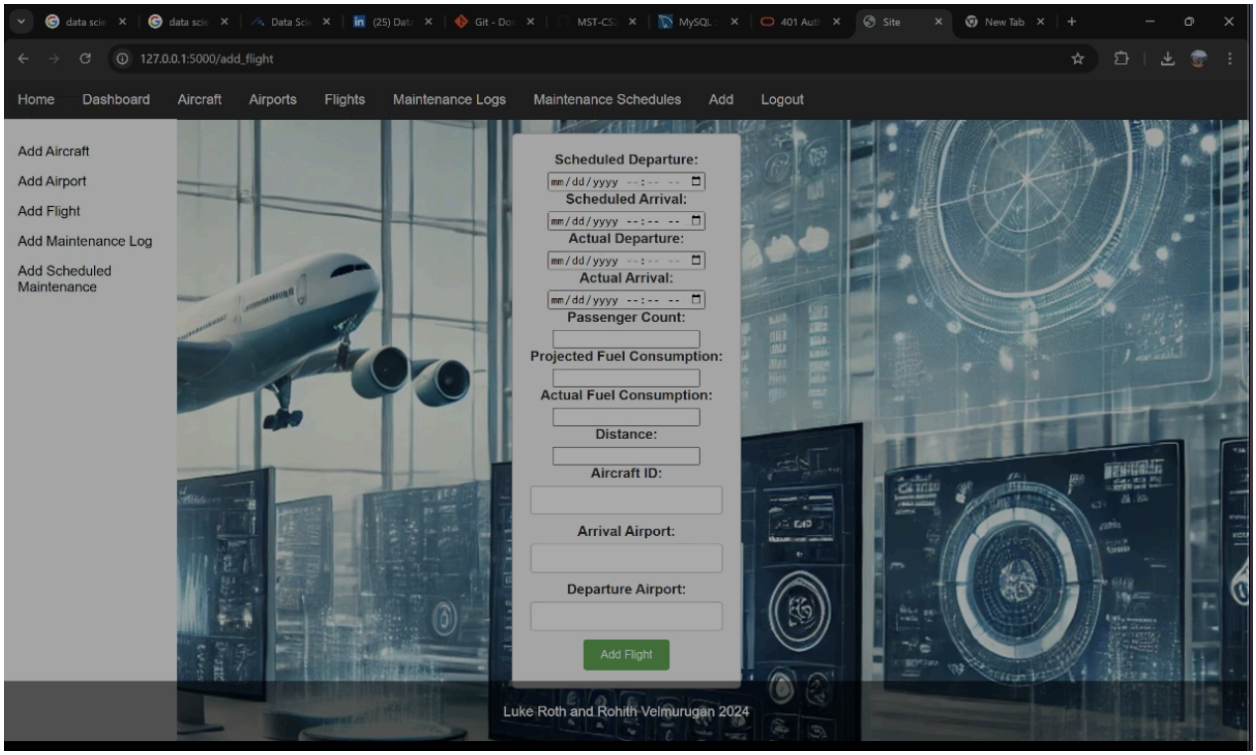


3. **Manage Data:**

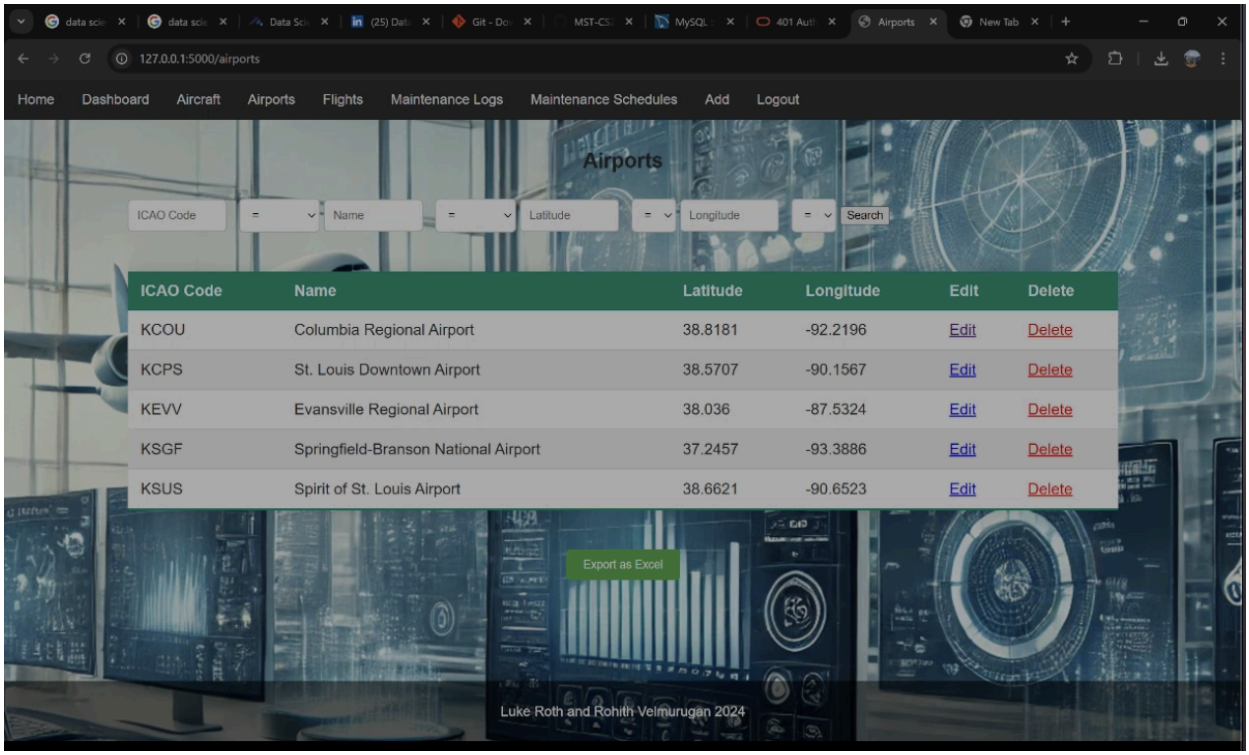
   Aircraft: Navigate to the Aircraft tab to view, edit, or delete aircraft records.
   Airports: Navigate to the Airports tab to view, edit, or delete airport records.
   Flights: Navigate to the Flights tab to view, edit, or delete flight records.
   Maintenance Logs: Navigate to the Maintenance Logs tab to view, edit, or delete maintenance log records.
   Maintenance Schedules: Navigate to the Maintenance Schedules tab to view, edit, or delete maintenance schedule records.



4. **Add Data: Navigate to the Add tab, then select an option from the left sidebar to add an entry in the respective table.**



5. **Export Data: Use the export buttons available on each data management page to download records as CSV files.**



6. **Reports: Generate performance reports for specific aircraft by clicking View on any aircraft in the Aircraft Tab.**