Group 3 Project Proposal

Project Team

Andrew Chi, Eric McElhinny

Project Title: Flight Booking System

URL: http://classwork.engr.oregonstate.edu:16490/flight-admin

Executive Summary

The project underwent significant simplification by removing the Passengers, Reservations, and Invoices entities, streamlining the system to allow only one seat purchase at a time from the database's perspective. Relationships between Seats and Flights, as well as Seats and Travel Classes, were revised to ensure they are non-optional, preventing the addition of a seat without its corresponding flight and travel class. To maintain consistency across the database, the naming syntax for Airports was updated, and specific numbers were added to the overview to more accurately quantify the system's potential application. Some feedback, such as the suggestion to include all four relationship types, was deemed unnecessary since there are only three types (1:1, 1, M). Additionally, the recommendation to add a Flight ID to the airplane_travel_classes table was not followed due to the assumption that specific airplane types would have consistent capacities.

Further simplifications were made, reinforcing the relationships within the database to ensure alignment with the ERD. The schema was realigned visually to better represent these relationships, and adjustments were made to the DDL to align with insert statements and ticketing notes. The database's naming conventions for Airports were revised to maintain consistent attribute naming throughout. Feedback suggesting additional filters in the UI was acknowledged, though the existing filters were maintained for simplicity. Some design decisions were upheld, such as using TINYINT instead of BOOLEAN due to MySQL's automatic conversion and performance considerations.

To meet rubric requirements and facilitate database creation on OSU servers, schema creation statements were removed from the DDL, and the schema was realigned to better reflect relationships. The airplane_travel_classes table was updated with an INSERT statement to ensure the addition of 0 capacity entries for each travel class when a new airplane is added. The Seats data was updated to match the DDL, and the seats constraint was removed to be handled in the backend. An INSERT and DELETE UI was added for seats, though this

introduced risks by allowing the creation of seats outside the standard configuration, placing a burden on administrators. DELETE statements were implemented across all tables, with the Flights and Seats tables converted to ON DELETE CASCADE to ensure that when an airport is deleted, all associated flights and seats are also deleted. A flight select statement was added for use in the Seats flight filter.

Certain feedback was not implemented based on the system's design assumptions. For instance, the recommendation to add a Flight ID to the airplane_travel_classes table was not followed due to the assumption that airplane types would maintain consistent capacities. Additionally, feedback regarding transitive dependency in the Airports table was disagreed with, as the database lacks the necessary information to make such inferences. Overall, the project maintained its design approach, balancing feedback with the practical constraints and assumptions of the system.

Summary of Changes From Feedback

Actions:

Step 3
It was recommended to remove the schema creation statements in the DDL so that the database can be created on the OSU servers, which was done. This document was also re-ordered to meet the rubric requirements. It was noted that the optionality in the ERD was lost in the Schema, this was rectified. The airport ID's in the seats table were removed from the outline to match the schema. The DDL was adjusted to align with the notes on insert statements and tickets. The schema was adjusted visually to align the relationships. The Seats data was updated to match the DDL. The seat constraint was removed and will be handled in the backend. An INSERT and DELETE UI was added for seats. This is dangerous though because it allows for seats outside the standard configuration to be made and places a lot of onus on the administrators to be correct. It was noted by several commenters to add entries to M:M tables if a FK was added to its table. While the SQL queries did exist for this function, these comments helped us understand adding an INSERT statement to the airplane_travel_classes table, where adding a new airplane will trigger the addition of 0 capacity entries of each travel class for the new plane. Adding lines to seats will be a complicated effort in the back end that will require multiple queries to determine the number of seats of each travel class to be added per flight. DELETE statements exist for all tables and Flights and Seats tables were converted to ON DELETE CASCADE so that if an Airport is deleted all flights involving it are deleted as are their relevant seats.

Step 2
The most impactful action taken from the feedback was to reduce the complexity of the design by removing the Passengers, Reservations, and Invoices entities. This removes the billing portion of the system and simplifies the ticketing by reducing the system to allowing only one seat to be purchased at a time (from the database's perspective).

Additional, specific, numbers were added to the overview to more specifically quantify the potential system application.
The relationships between Seats and Flights and Seats and Travel Classes were revised to make these relationships non-optional for a seat. Therefore, a seat cannot be added without a flight and travel class.
The naming syntax was updated for Airports, so the attributes are named consistently throughout the database.

Step 1
The most impactful action taken from the feedback was to reduce the complexity of the design by removing the Passengers, Reservations, and Invoices entities. This removes the billing portion of the system and simplifies the ticketing by reducing the system to allowing only one seat to be purchased at a time (from the database's perspective).
Additional, specific, numbers were added to the overview to more specifically quantify the potential system application.
The relationships between Seats and Flights and Seats and Travel Classes were revised to make these relationships non-optional for a seat. Therefore, a seat cannot be added without a flight and travel class.
The naming syntax was updated for Airports, so the attributes are named consistently throughout the database.

Actions Not Taken:

Step 3
It was mentioned the UI was not working in 1 review and that inserts were not implemented. The other reviewers were unable to access the UI, there may have been a VPN issue for this reviewer. The seats UI did have a filter, but the comment encouraged us to add more filters.
Step 2
It was recommended that a Flight ID be added to the Airplane_Travel_Classes table in order to allow the same Airplane_Types to have different capacities. It is an assumption of this design that a specific Airplane_Type would have the same capacities. If it did not then it would be a variant of that type, a 737-800 vs 737-900 for example. If this assumption was broken, then there would be no point in having an Airplane_Types table. Based on this, the feedback was not followed. It was recommended that the primary key be specified in the outline relationships in addition to the foreign key. This was decided against as unnecessary because while technically relationships can use any unique column as a foreign key, it is not generally considered good coding practice, and we do not currently have any unique, non-key attributes in the schema. It was recommended that instances of the TINYINT datatype were changed to BOOLEAN. This was not accepted as TINYINT and BOOLEAN are synonymous in MySQL and the workbench automatically converts to TINYINT. It was decided to remain with snake case. Workbench does not seem to have the capability to tie relationships to specific attributes the way requested in the reviews. If viewing in the actual environment, the keys are highlighted with respect to the relationship. If desired the model file can be provided, but for this application we feel the representation is sufficient. We disagreed with the feedback that the Airports table was transitively dependent. While people may be able to determine the city from the country the

database does not have the information to make that decision, and it requires serious assumptions as well. Additionally, from a performance and safety perspective, we decided to keep Available in the Seats table. Querying against NULL is risky and slower than querying against a Boolean value.

Step 1
Several pieces of advice were not followed. It was suggested that the project should have all 4 unique relationship types. Given that there are only 3 relationship types (1:1, 1:M, and M:M) this advice was disregarded. Also, any changes to the three removed entities were moot as well, although they will be taken into consideration if they are ever added back in. There was a misconception that Flights and Seats should have an M:M relationship. This is not the case because a flight and sat on that flight are unique. When a new flight is charted, every seat gets a new ID. Therefore, the relationship between flight and seat is 1:M. When a flight ends, the seat and flight are archived and a new flight and seats are created. Finally, it was raised that a flight will need to be limited to a certain number of seats and, more specifically, a certain number of seats per travel class. This is correct and addressed by the constraint on the seats limiting the number of seats by the travel class and plane type

Upgrades:

Step 3
A flight's select statement was added to be used for the Seats flight filter.
Step 2
Based on the removal of passengers, reservations, and invoices entities, additional fields were added to the seats table. Available was added to indicate if the seat was reserved already. Passenger was added to allow the system to have a reference for who had reserved the seat.

Step 1
Based on the removal of passengers, reservations, and invoices entities, additional fields were added to the seats table. Available was added to indicate if the seat was reserved already. Passenger was added to allow the system to have a reference for who had reserved the seat.

Project Outline

Overview:

The main object of this project is to provide consumers with a straightforward way to look and book their flights. Users are able to look up seat tickets, receive information on ticket prices, and browse flight options available at various times on a given date using this database. It stores all the available flights and seats that are needed to plan a trip and will store, organize, and manage a database of multiple flights. The system is scalable to support regional airlines running 400 flights with 8000 seats at a time.

Database Outline

- **Airports:** records the location details of potentially serviced airports
    - Airport_ID: PK, INT, not null, AI
    - Airport_Name: VARCHAR(100), not null
    - Airport_City: VARCHAR(100), not null
    - Airport_Country: VARCHAR(100), not null
        - **Airport Relationships:**
            - A 1:M (Optional) between Airports and Flights is implemented, with Origin_Airport_ID as a FK in Flights.
            - A 1:M (Optional) between Airports and Flights is implemented, with Destination_Airport_ID as a FK in Flights.

- **Flights:** records the departure and arrival information for a specific flight
    - Flight_ID: PK, INT, not null, AI
    - Origin_Airport_ID: INT, FK
    - Destination_Airport_ID: INT, FK
    - Departure_Date_Time: DateTime, not null
    - Arrival_Date_Time: DateTime, not null
    - Airplane_Type_ID: INT, FK
        - **Flights Relationships:**
            - A M (Optional):1 between Flights and Airports is implemented with Origin_Airport_ID as a FK in Flight_Details.
            - A M (Optional):1 between Flights and Airports is implemented, with Destination_Airport as a FK in Flight_Details.
            - A M (Optional):1 between Flights and Airplane_Types is implemented with Airplane_Type_ID as a FK in Flights.
            - A 1:M (Optional) between Flights and Seats is implemented with Flight_ID as a FK in Seats.

- **Seats:** records the specific seat reserved for a reservation. A constraint is placed to limit the number of seats of a specific class on a specific flight to the number available on the flight's plane type
    - Seat_ID: PK, INT, not null, AI
    - Seat_Number: VARCHAR(100), not null
    - Travel_Class_ID: INT, FK
    - Flight_ID: INT, FK
    - Available: TINYINT, not null
    - Passenger_Name: VARCHAR(100)
        - **Seats Relationships:**
            - A M (Optional):1 (Optional) between Seats and Travel_Classes is implemented with Travel_Class_ID as a FK in Seats.

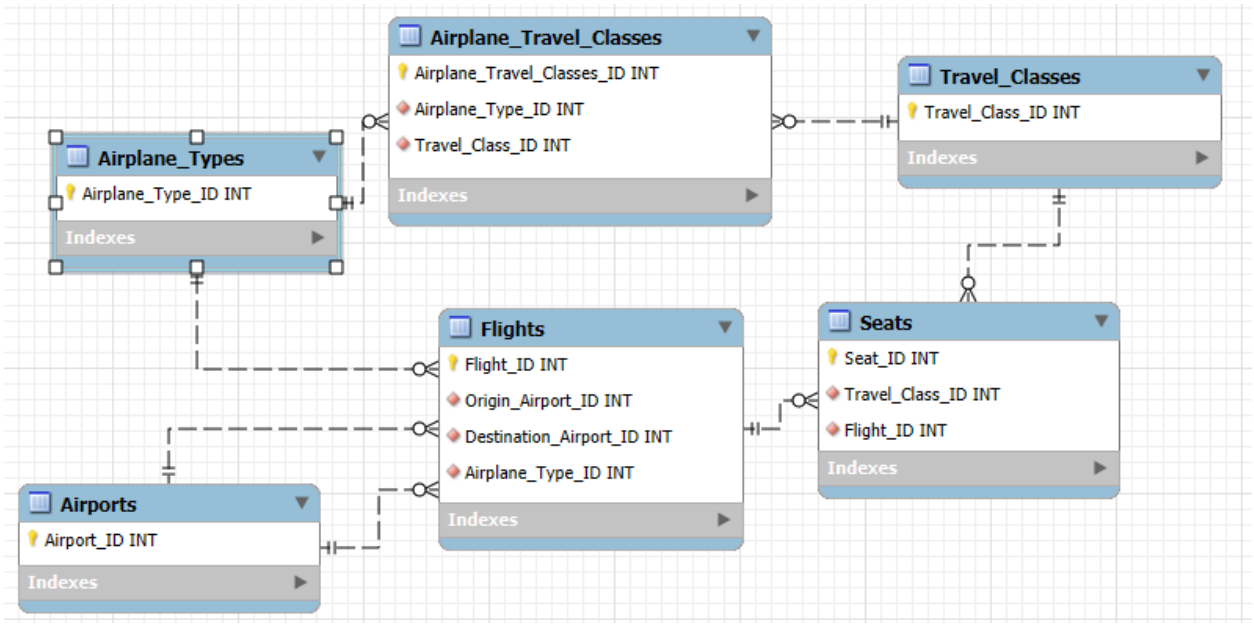
- A 1:M (Optional) between Flights and Seats is implemented with Flight_ID as a FK in Seats.

- **Travel_Classes:** differentiates seats based on comfort and cost
    - Travel_Class_ID: PK, INT, not null, AI
    - Travel_Class_Name: VARCHAR(100), not null
    - Travel_Class_Cost: Decimal(20,2), not null
        - **Travel_Class Relationships:**
            - A 1 (Optional) :M (Optional) between Travel_Classes and Seats is implemented with Travel_Class_ID as a FK in Seats.
            - A M:M between Travel_Classes and Airplane_Types is implemented with Airplane_Travel_Classes as a Junction Table
                - A 1:M (Optional) between Travel_Classes and Airplane_Travel_Classes is implemented with Travel_Classes_ID as a FK in Airplane_Travel_Classes.

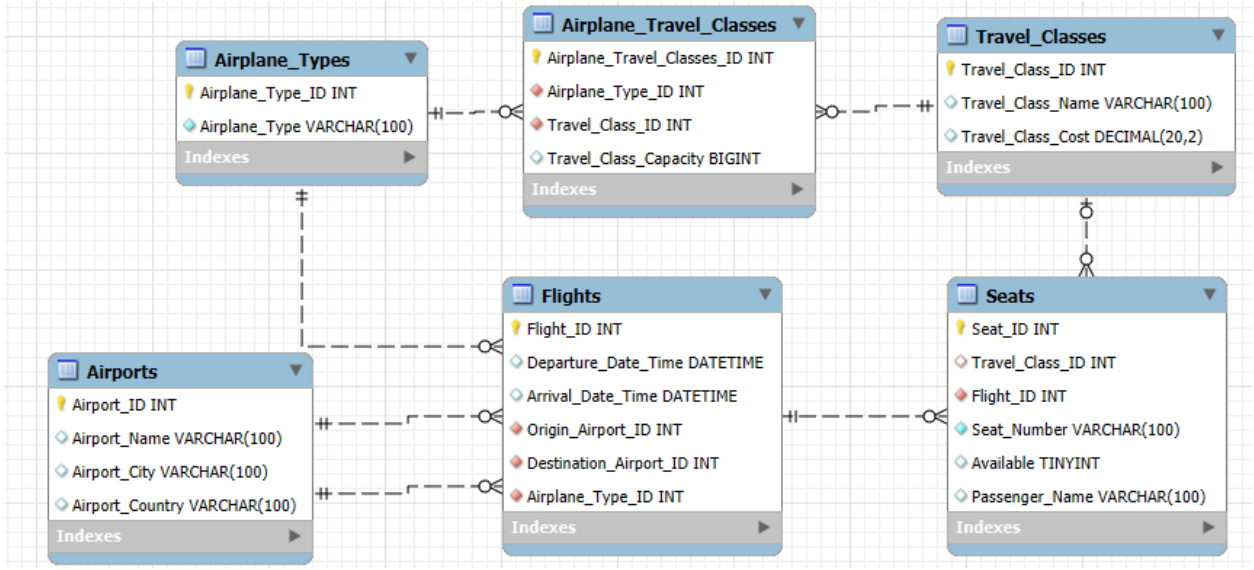
- **Airplane_Types:** records the types of airplanes in the fleet
    - Airplane_Type_ID: PK, INT, not null, AI
    - Aiplane_Type: VARCHAR(100), not null
        - **Airplane_Types Relationships:**
            - A 1:M (Optional) between Airplane_Types and Flights is implemented with Airplane_Type_ID as a FK in Flights.
            - A M:M between Travel_Class and Airplane_Types is implemented with Airplane_Travel_Classes as a Junction Table
                - A 1:M (Optional) between Airplane_Types and Airplane_Travel_Classes is implemented with Airplane_Type_ID as a FK in Airplane_Travel_Classes.

- **Airplane_Travel_Classes:** used to link the number of each seat class are present on each plane type
    - Airplane_Travel_Classes_ID: PK, INT, not null, AI
    - Airplane_Type_ID: INT, FK
    - Travel_Class_ID: INT, FK
    - Travel_Class_Capacity: BIGINT
        - **Airplane_Travel_Classes Relationships:**
            - A M (Optional):1 between Airplane_Travel_Classes and Travel_Classes is implemented with Travel_Class_ID as a FK in Airplane_Travel_Classes.
            - A M (Optional):1 between Airplane_Travel_Classes and Airplane_Types is implemented with Airplane_Types_ID as a FK in Airplane_Types.

Entity-Relationship Diagram



Schema



Data

Airports

Airport_ID	Airport_Name	Airport_City	Airport_Country
1	MDT	Harrisburg	United States
2	LGA	New York	United States
3	LHR	London	United Kingdom
4	HND	Tokyo	Japan
5	CAI	Cairo	Egypt

Airplane_Types

Airplane_Type_ID	Airplane_Type
1	Boeing 737
2	Airbus A320
3	Bombardier CRJ
4	Cessna 172
5	Boeing 747

Travel_Classes

Travel_Class_ID	Travel_Class_Name	Travel_Class_Cost
1	First	100
2	Business	50
3	Economy	10

Flights

Flight_ID	Departure_Date_Time	Arrival_Date_Time	Origin_Airport_ID	Destination_Airport_ID	Airplane_Type_ID
1	7/16/2024 8:00	7/16/2024 20:00	2	3	1
2	7/16/2024 9:00	7/16/2024 20:00	3	4	2
3	7/16/2024 10:00	7/16/2024 21:00	4	3	3
4	7/16/2024 11:00	7/16/2024 22:00	2	5	3
5	7/16/2024 12:00	7/17/2024 0:00	1	5	5

Seats

Seat_ID	Travel_Class_ID	Flight_ID	Seat_Number	Available	Passenger_Name
1	1	1	1	0	James Smith
2	2	1	2	1	NULL
3	3	1	3	1	NULL
4	1	2	1	1	NULL
5	1	2	2	1	NULL

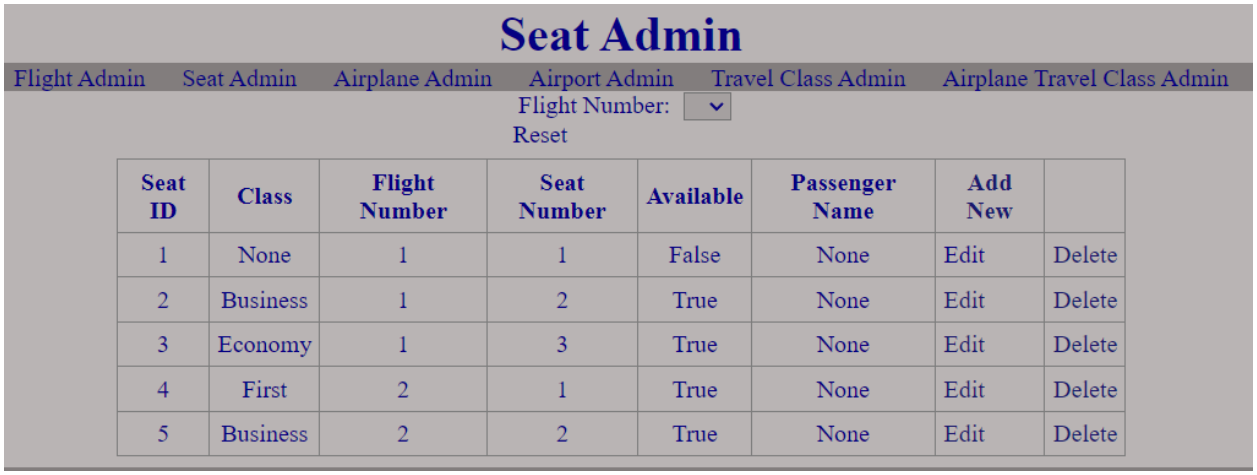
Airplane_Travel_Classes

Airplane_Travel_Classes_ID	Airplane_Type_ID	Travel_Class_ID	Travel_Class_Capacity
1	1	1	12
2	1	2	36
3	1	3	102
4	2	1	16
5	2	2	40

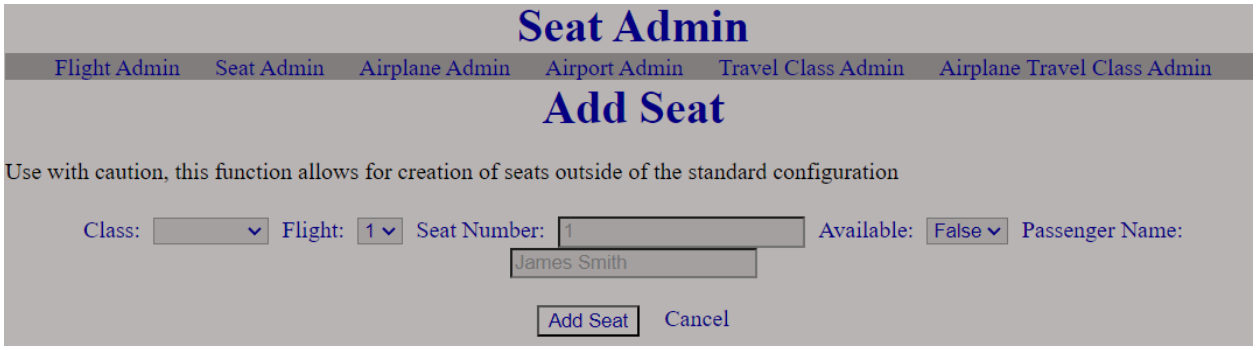
6	2	3	120
7	3	1	8
8	3	2	24
9	3	3	60
10	4	1	1
11	4	2	1
12	4	3	2
13	5	1	24
14	5	2	72
15	5	3	200

UI Screenshots

Read - Seats page

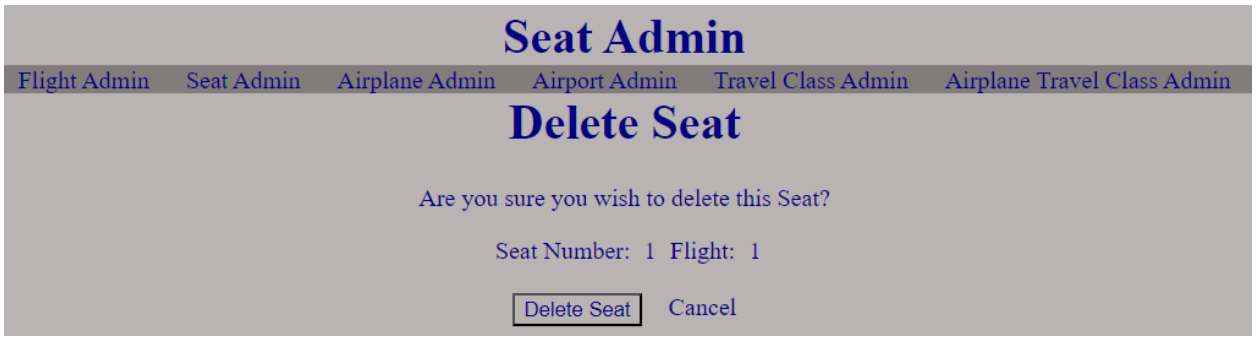


Create - Seats page



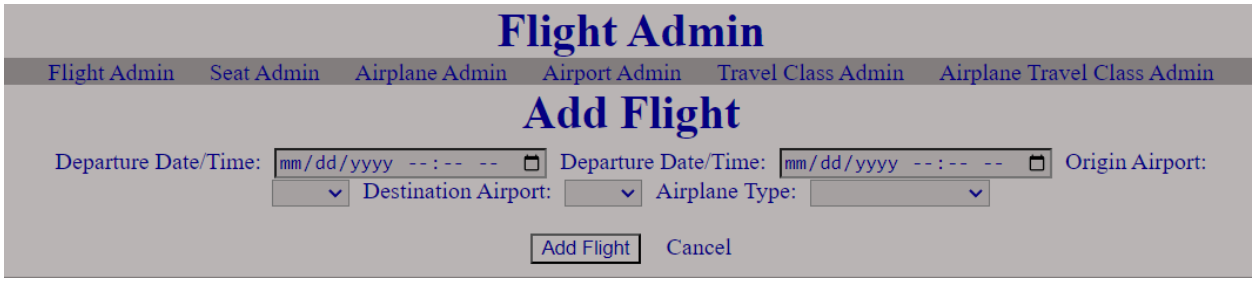
Update - Seats page

Delete - Seats page

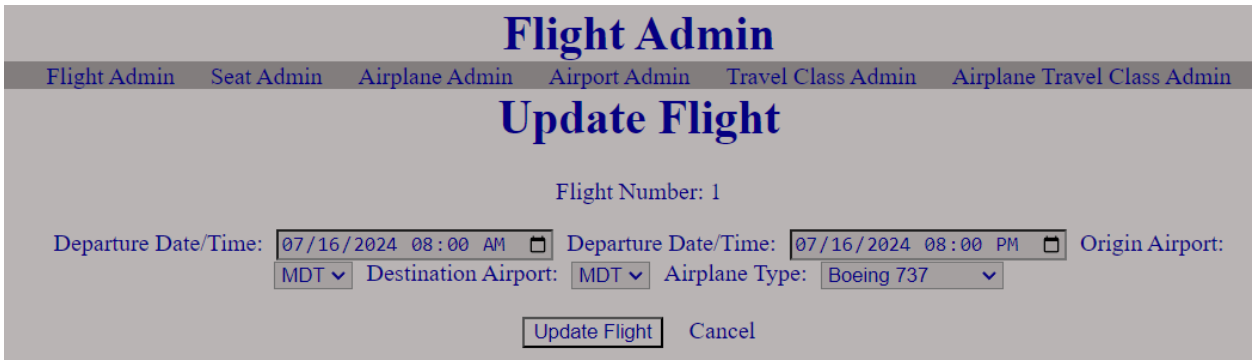


Read - Flights page

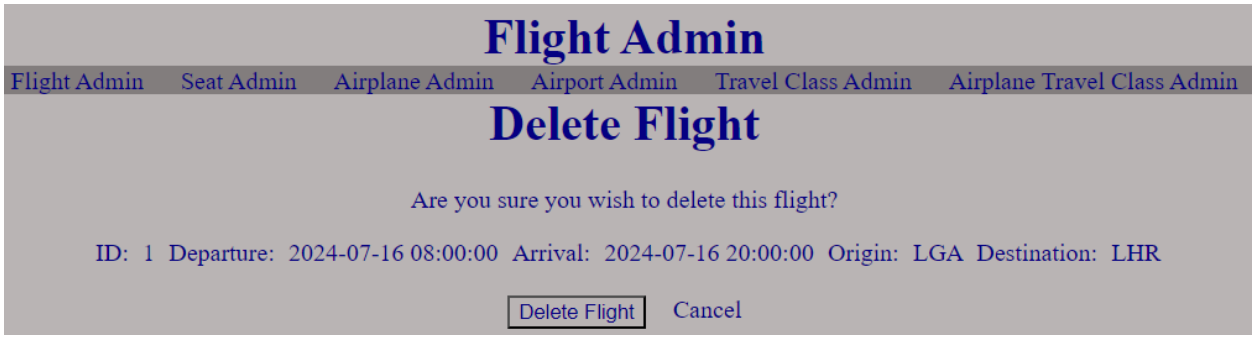
Create - Flights page



Update - Flights page

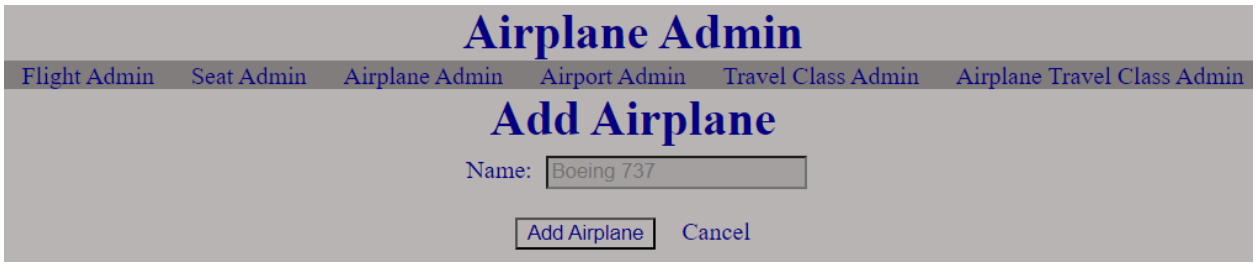


Delete - Flights page

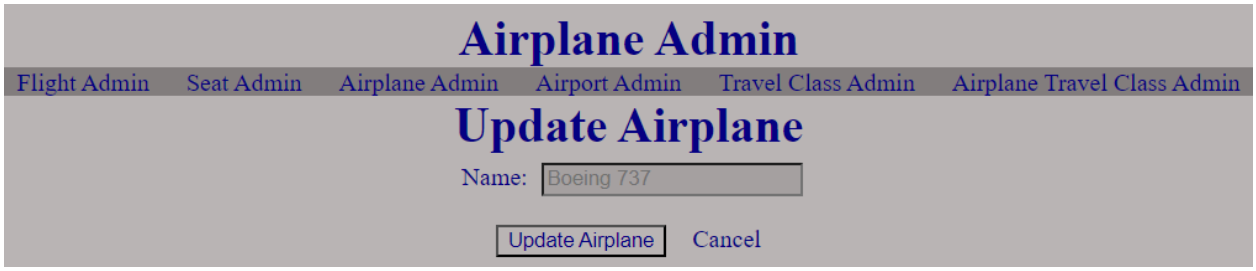


Read - Airplane Types page

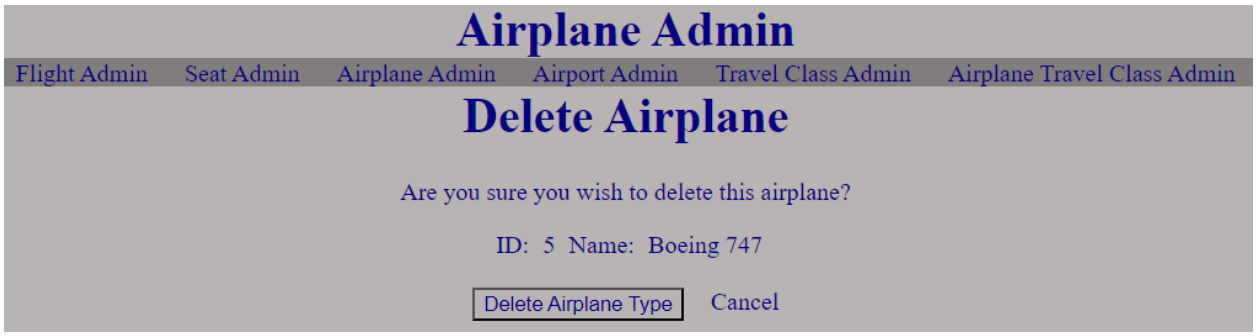
Create - Airplane Types page



Update - Airplane Types page

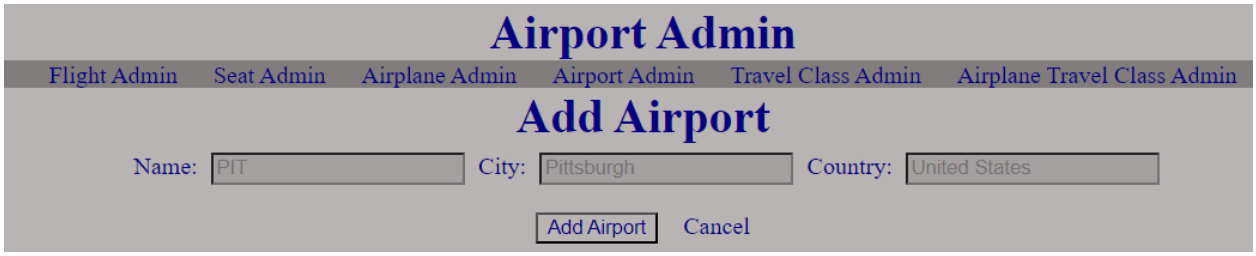


Delete - Airplane Types page

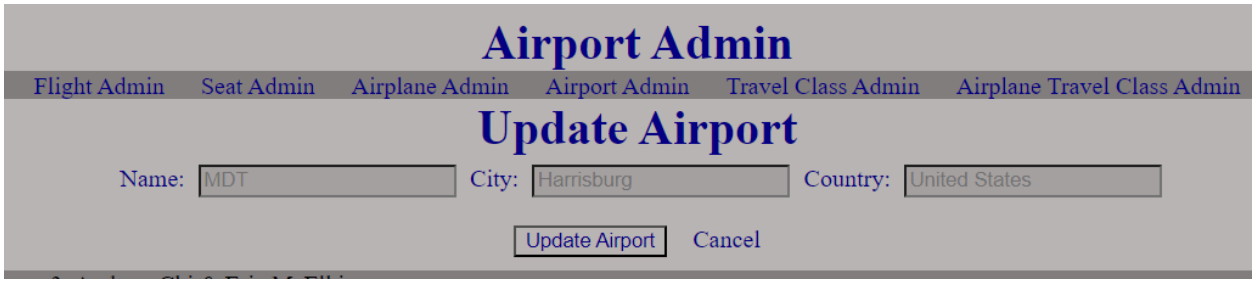


Read - Airports page

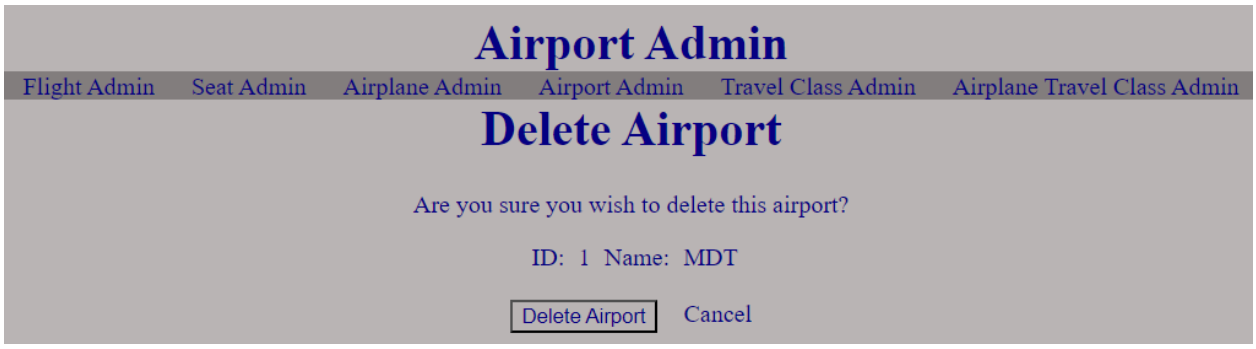
Create - Airports page



Update - Airports page



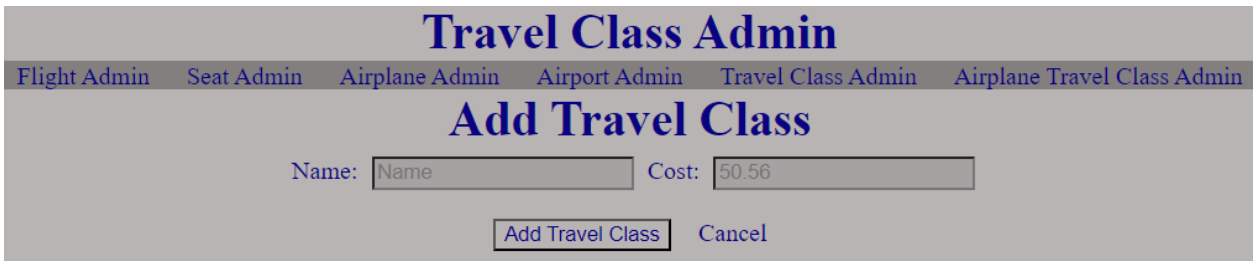
Delete - Airports page



Read - Travel Class page

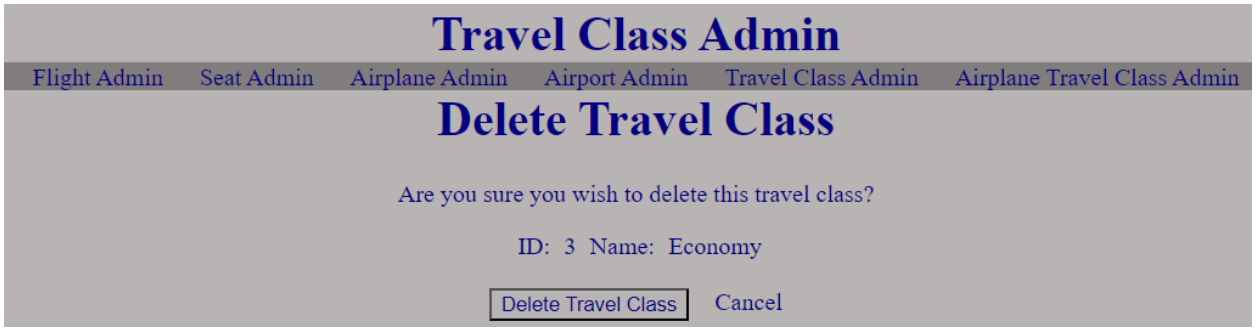
Travel Class ID	Travel Class Name	Travel Class Cost	Add New	
1	First	100.00	Edit	Delete
2	Business	50.00	Edit	Delete
3	Economy	10.00	Edit	Delete

Create - Travel Class page

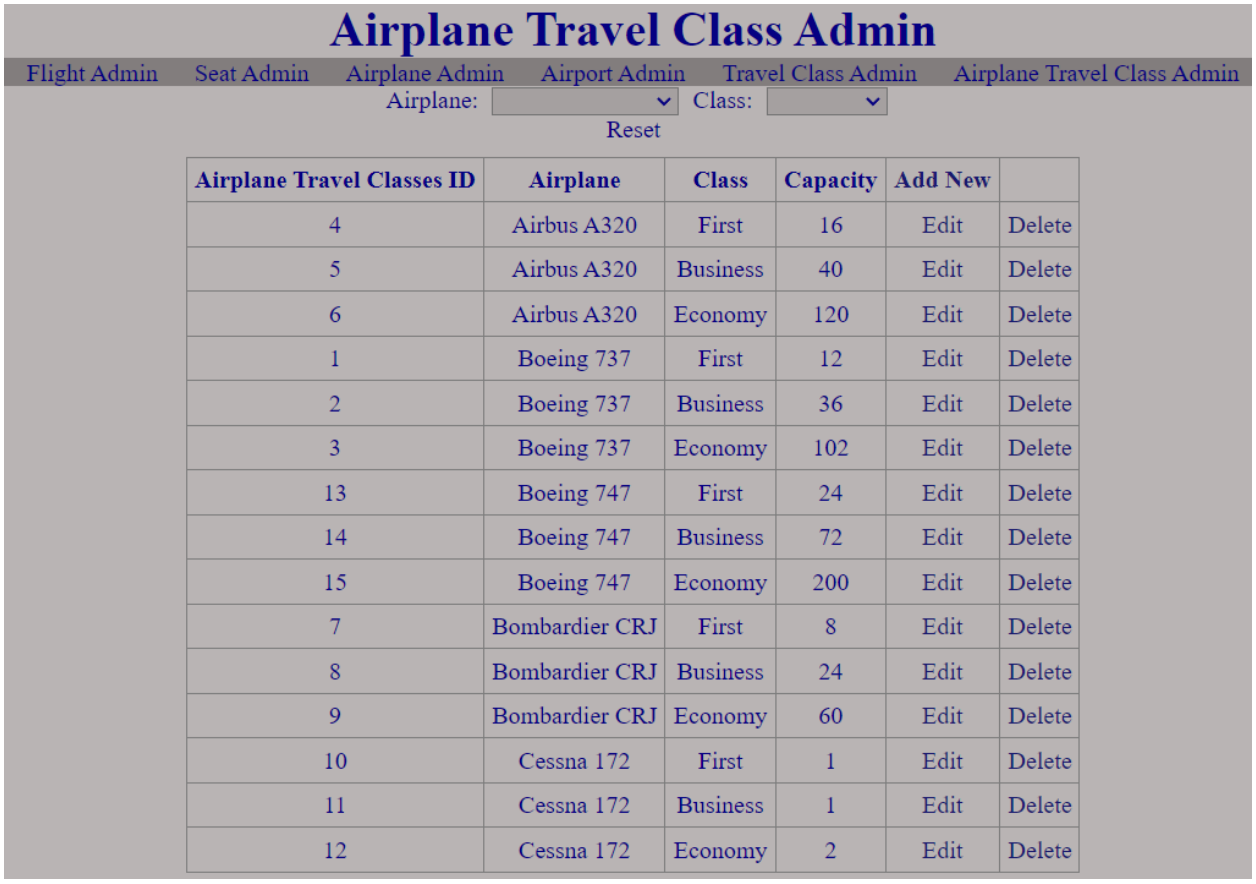


Update - Travel Class page

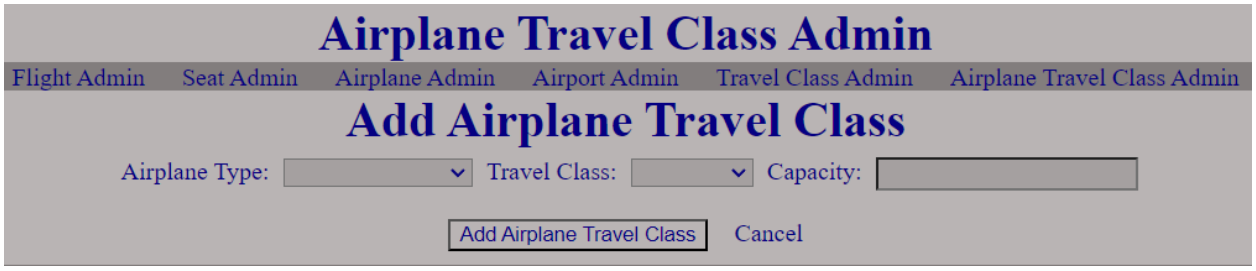
Delete - Travel Class page



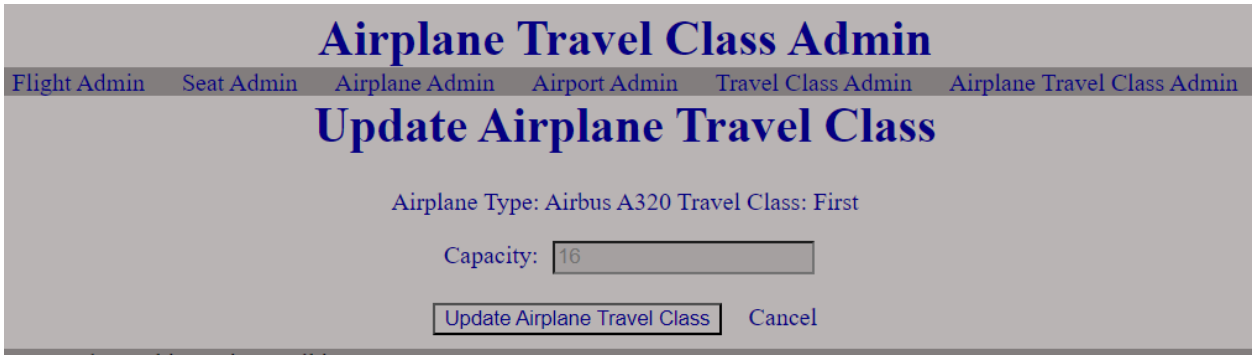
Read - Airplane - Travel Class page



Create - Airplane - Travel Class page



Update - Airplane - Travel Class page



Delete - Airplane - Travel Class page