AeroManageX

Database Management System

MSCS 542L-256

Aerotech Titans

Marist College

School of Computer Science and Mathematics

Submitted To:

Dr. Reza Sadeghi

September 13, 2023

Table of Contents

Project Report of AeroManageX 4
1. Team Name 4
2. Aerotech Titans 4
Team Members 4
Description of Team Members 4
3. AeroManageX Objective 4
4. Review Related Works 5
5. The Merits of Our Project 5
6. GitHub Repository Address 5
7. External ER Diagrams 5
Manager Description 5
Manager Model 6
Booking Agent Description 6
8. Entity Relationship Diagram (ER Diagram) 7
Description 7
9. Enhanced Entity Relationship Diagram (EER) 8
Description 8
Diagram 10
10. Database Development 11
Table: Cost 11
Table: Airline 13
Table: Passenger 14
Table: Pilot 16
Table: Des_Airport 17
Table: Org_Airport 18
Table: Flight_Attendent 20
Table: Plane 21
Table: Booking 22
Table: Baggage 24
Table: Flight 26
Table: Flight_Attendent_has_Flight 28
11. Loading data and performance enhancements 29

Handling foreign key constraints 29
Importing data 29
Insertion optimization 30
Normalization Check 30
12. Application Development 30
Graphical User Interface Design 30
View's implementation 32
13. References 37

Project Report of AeroManageX

1. Team Name

2. Aerotech Titans

Team Members

1. Bashir Dahir bashir.dahir1@marist.edu (Team Head)
2. Nihar Lodaya nihar.lodaya1@marist.edu (Team Member)
3. Marguerite McGahay marguerite.mcgahay@marist.edu (Team Member)

Description of Team Members

1. Bashir Dahir

I'm a Computer Science student at Marist College, Beacon, New York, in my fifth year, with just two semesters to go before graduation. My academic journey has sharpened my skills in programming and data structures, but my true passion lies in database management systems. Currently, I'm eagerly gearing up for a project focused on airline management systems, where I plan to apply my expertise in database management to create efficient and robust solutions for the aviation industry. Beyond academics, I enjoy problem-solving and community engagement.

2. Nihar Lodaya

Hey there, I'm Nihar Lodaya, and I come from India. I've got a bachelor's degree in computer science from back home, and right now, I'm in the middle of my master's program in Computer Science at Marist College. I'm stoked about working with this bunch of awesome folks on our current project. What really got me excited about my teammates Bashir & Marguerite is how dedicated they are to making this project a success.

3. Marguerite McGahay

I grew up in Poughkeepsie and then attended the University of Delaware, where I graduated in 2021 with a BS in Mathematics and a Minor in Computer Science. While I was there, I was a TA for multiple computer science classes and was also on the Women's Rowing Team! I currently work at Marist as the Assistant Women's Rowing Coach. This is my first semester in the Computer Science – Software Development program, so I don't know many people, but I was excited when Bashir and Nihar extended an invitation for me to be a member of this group. We selected our team head, Bashir, since it was his initial idea to pursue this project, but I truly believe each member of this team has the capability to be able to step into that role if asked of us.

3. AeroManageX Objective

The primary objective of the AeroManageX project is to elevate your airline's existing database infrastructure to be able to better compete with the world's top airlines. Our company will help you condense the mass amount of information needed to be able to have thousands of planes arrive and depart each day. Our system operates to help your airline in multiple ways, including but not limited to managing flight bookings, optimizing the cost of flights, making sure planes are in the necessary airports based on different flights' place of departure and arrival, and efficiently assigning pilots to flights that make sense for your company and their schedules.

4. Review Related Works

There are many competitions for our company including Ramco Aviation, Airline Suite, and AvPro. Each of these companies has positive and negative aspects. To begin with, Ramco Aviation published on their website that they provide the best aviation software for the following three reasons: “Proven technology champions, usability focusses and in memory planning and optimization” (2). However, Ramco Aviation critics emphasize on their “Security and data control issues and difficulties of data migration” (*2*). Second, Airline Suite promotes that their system is easy to use, scalable and affordable. On the other hand, however, Airline Suite is notorious for having “no history of previous or editions and no import options for Microsoft Excel or file” (1). Finally, AvPro is famous for their “reliability reporting, budget forecasting, and inventory management” (4). On the contrary, AvPro lacks the availability of “instruction manuals and system stability” as there are system glitches (3).

5. The Merits of Our Project

Our company will provide airlines with more features than any other competitive company. We provide airlines with the ability to see and organize information which can be categorized into three principal operations:

Our Land Operations:

- Checked baggage: Each passenger has a bag that must be on their flight.
- Staff management: Such as gate-workers, baggage handlers, custodial staff, etc.
- Times of arrival and departure of flights
- Fleet inventory

Air Operations:

- Flight plan: Including non-stop flights and layovers.
- Assigning pilots and airline staff.

Billing Operations:

- Bookings: Secure database for credit card numbers.
- Cost of seats
- Checked luggage (including oversized): If a passenger checks a bag that is over 50lbs, they must pay an extra fee on top of the base cost.

An airline should choose AeroManageX to manage their data because we are security focused and can benefit any size airline, from companies with just a small fleet of planes to global “Aero-Titans"!

6. GitHub Repository Address

Here is our GitHub repository’s URL.

https://github.com/bashirad/MSCS-542L-256_AeroManageX_Aerotech-Titans.git

7. External ER Diagrams

Manager Description

In the External Entity Relationship Diagram for the Manager Model in our AeroManageX, the primary entities include Flight, Origin Airport, Destination Airport, Pilot, Flight Attendant, Plane, and Airline. Flight is the central entity, while Origin Airport and Destination Airport are directly connected to Flight, representing the departure and arrival locations. Pilots are associated with specific flights as they operate them, and Flight

Attendants work on board these flights. The Plane entity represents the aircraft used for the flights, and Airline serves as a parent entity encompassing various flights operated by the airline. This diagram illustrates how these entities are interrelated, providing a clear overview of the airline's operations.

Manager Model

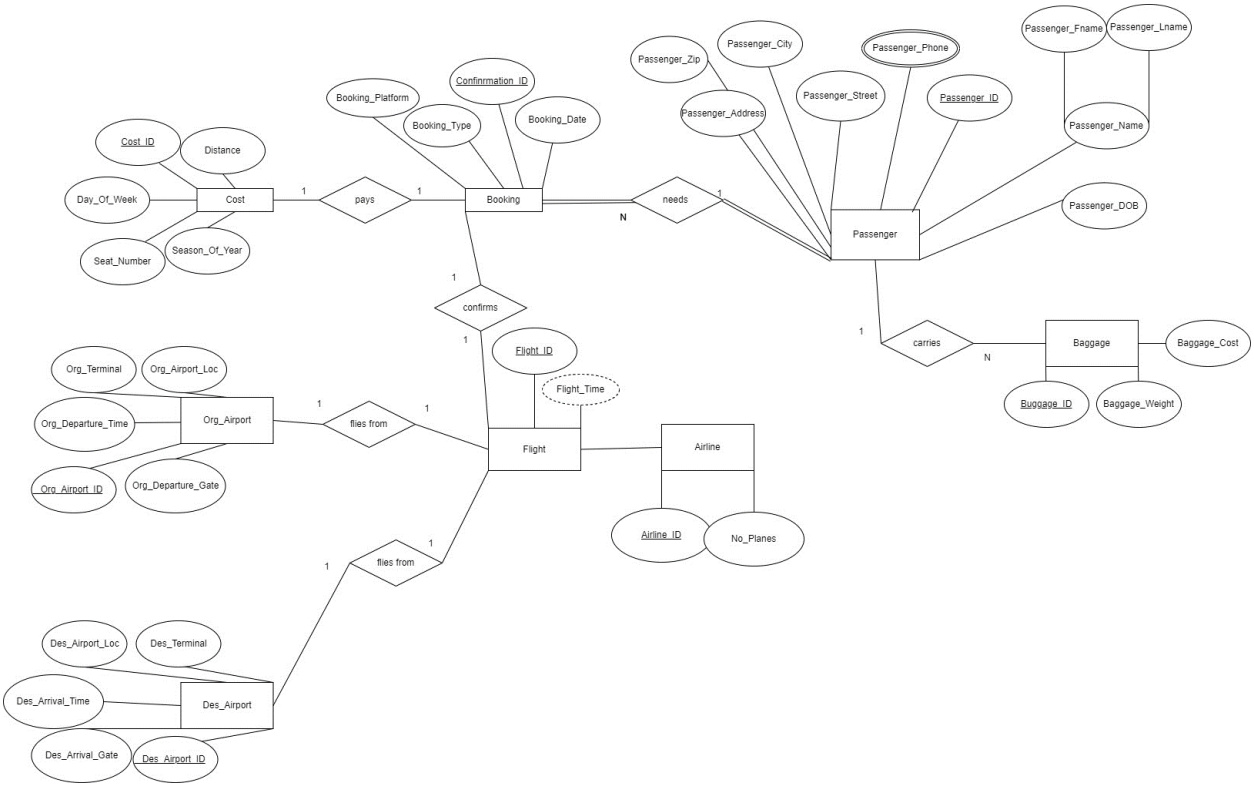


Figure 1

Booking Agent Description

In the External Entity Relationship Diagram for the Booking Agent Model in the AeroManageX, a web of interconnected entities and relationships emerges. Cost is linked to Booking, reflecting the financial transaction between booking agents and reservations. Booking necessitates Passenger, showing the connection between bookings and the passengers they are made for, while Passenger also carries Baggage, signifying the belongings associated with passengers. Bookings confirm Flights, illustrating the reservation process, and Flights are tied to both Origin and Destination Airports, representing the departure and arrival points of the journeys.

Booking Agent Model

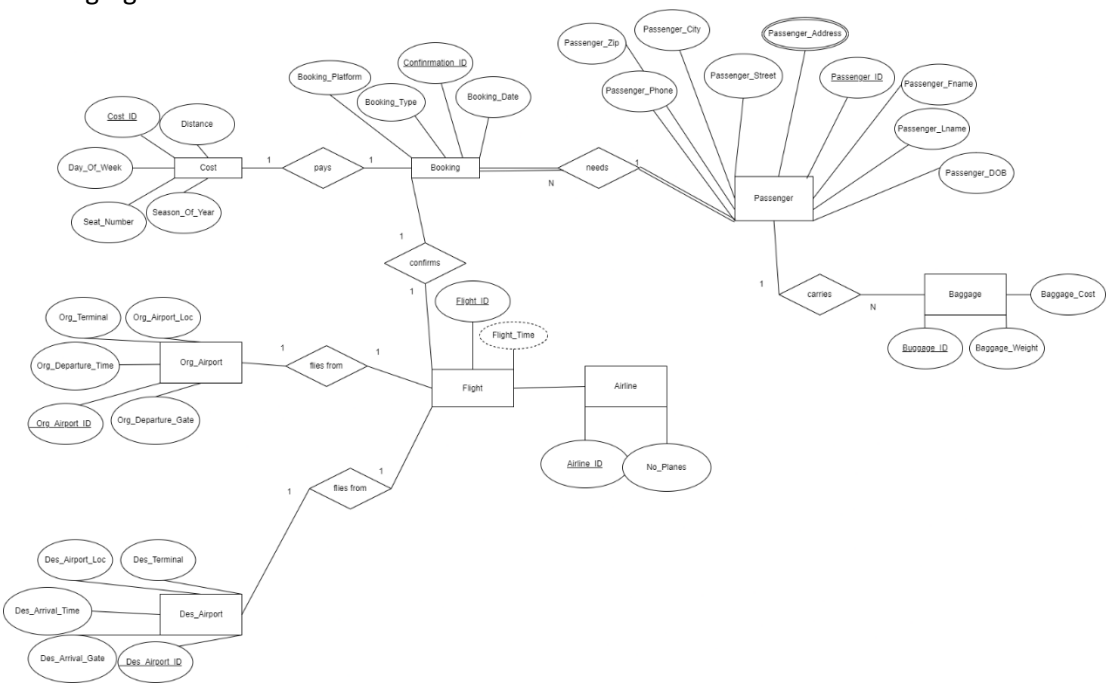


Figure 2

8. Entity Relationship Diagram (ER Diagram)

Description

Creating this project involves a thoughtful selection of entities, attributes, relationships, participations, and cardinalities to represent a simplified domain. In this case, the chosen entities include "Cost," "Baggage," "Booking," "Plane," "Flight," "Flight_Attendant," "Org_Airport," "Des_Airport," "Pilot," "Passenger," and "Airline." Each of these entities represents key elements in the domain of Airline travel. Attributes are then identified for each entity to capture relevant information; for instance, "Passenger" have attributes like Passenger_Fname, Passenger_Lname, Passenger_Address, etc. While "Cost" has attributes like Confirmation_ID, Distance, Season_Of_Year, etc. Relationships are established between entities to represent how they are connected, such as the relationship between "Booking" and "Passenger" to show that a passenger can make a booking.

Diagram

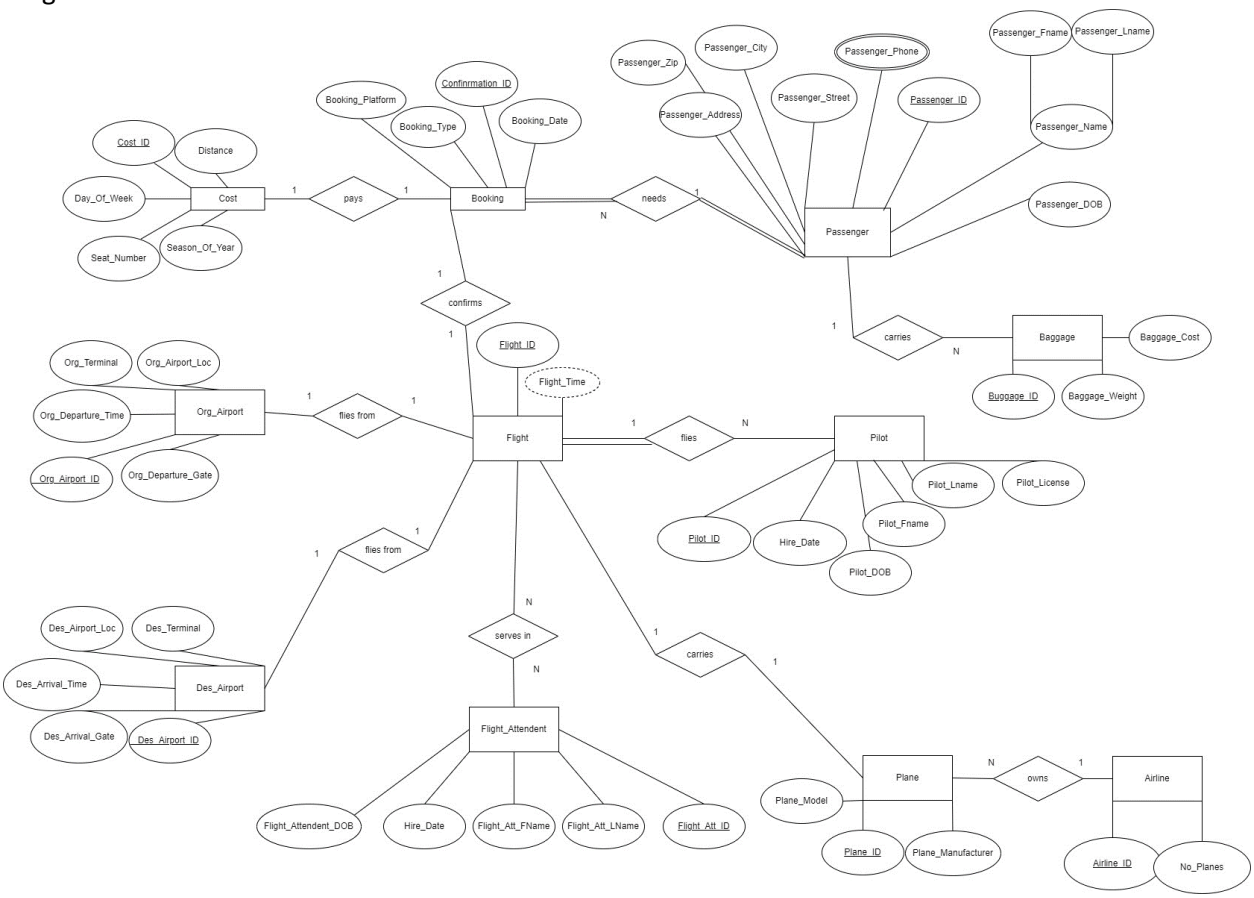


Figure 3

9. Enhanced Entity Relationship Diagram (EER)

Description

#	Entity	Key	Relationships
1	Flight	Flight_ID	Each flight has one origin airport, one destination airport, multiple bookings for their passengers, as well as a one plane, one pilot, and multiple flight attendants
2	Plane	Plane_ID	Each flight must have a plane, which owned by an airline
3	Airline	Airline_ID	Each airline has multiple planes in its fleet
4	Cost	Cost_ID	Cost is related to Booking via a 1-1 relationship.
5	Booking	Confirmation_ID	Each booking is related to a passenger, which results in a cost of

			booking. When a passenger books, they are then connected to a flight
6	Passenger	Passenger_ID	Each passenger can have one or more bags
7	Baggage	Baggage_ID	Each bag is related to one passenger
8	Org_Airport	Org_Airport_ID	Each flight must depart from one origin airport
9	Des_Airport	Des_Airport_ID	Each flight must arrive at one destination airport.
10	Pilot	Pilot_ID	Each flight must have one pilot. A pilot can have multiple flights in one day
11	Flight Attendant	Flight_Att_ID	Each flight can have multiple flight attendants and a flight attendant can have multiple flights in one day

Figure 4

Diagram

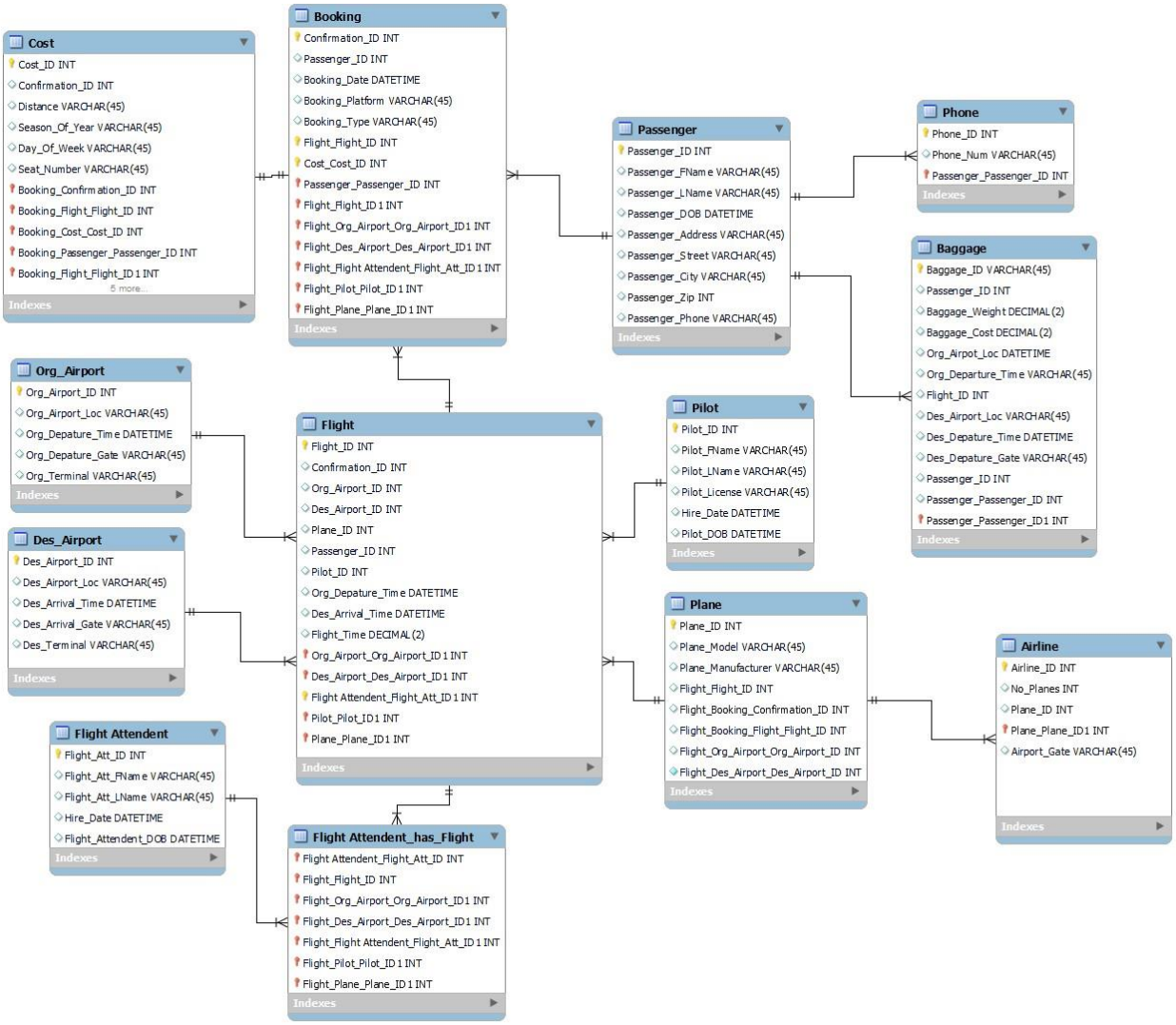


Figure 5

10. Database Development

Create database and tables.

```
DROP DATABASE IF EXISTS mydb;
CREATE DATABASE IF NOT EXISTS mydb DEFAULT CHARACTER SET utf8;
USE mydb;

-- GET RID OF ALL FOREIGN KEYS EXCEPT BOOKING'S CONFIRMATION_ID
-- Table mydb.Cost
DROP TABLE IF EXISTS mydb.Cost;
CREATE TABLE IF NOT EXISTS mydb.Cost (
  Cost_ID INT NOT NULL,
  Amount INT NOT NULL,
  Distance VARCHAR(45) NULL,
  Season_Of_Year VARCHAR(45) NULL,
  Day_Of_Week VARCHAR(45) NULL,
  Seat_Number VARCHAR(45) NULL,
  PRIMARY KEY (Cost_ID)
);
```

Table: Cost

Attribute	Data Type	Description
Cost_ID	INT	This attribute serves as the primary key for the "Cost" table, uniquely identifying each cost record. It is an integer (INT) data type.
Amount	INT	The "Amount" attribute represents the cost or price associated with a specific item or service. It is of integer (INT) data type and is required, meaning it cannot be null.
Distance	VARCHAR	This attribute stores textual information, likely related to the

		distance associated with the cost. It is of VARCHAR(45) data type, allowing up to 45 characters, and it is not marked as required (nullable).
Season_Of_Year	VARCHAR	The "Season_Of_Year" attribute stores information related to the season of the year associated with the cost. It is of VARCHAR(45) data type, allowing up to 45 characters, and it is not marked as required (nullable).
Day_Of_Week	VARCHAR	This attribute represents information related to the day of the week associated with the cost. It is of VARCHAR(45) data type, allowing up to 45 characters, and it is not marked as required (nullable).
Seat_Number	VARCHAR	The "Seat_Number" attribute stores information related to seat numbers associated with the cost. It is of VARCHAR(45) data type, allowing up to 45 characters, and it is not marked as required (nullable).

```
-- Table mydb.Airline
DROP TABLE IF EXISTS mydb.Airline;
CREATE TABLE IF NOT EXISTS mydb.Airline (
  Airline_ID INT NOT NULL,
  No_Planes INT NULL,
  Plane_ID INT NULL,
  Airport_Gate VARCHAR(45) NULL,
  Num_Employees INT,
  PRIMARY KEY (Airline_ID),
```

```
);
```

Table: Airline

Attribute	Data Type	Description
Airline_ID	INT	This attribute serves as the primary key for the "Airline" table, uniquely identifying each airline record. It is an integer (INT) data type and is required, meaning it cannot be null.
No_Planes	INT	The "No_Planes" attribute represents the number of planes associated with the airline. It is of integer (INT) data type and is not marked as required (nullable).
Plane_ID	INT	This attribute is used to store the ID of a specific plane associated with the airline. It is of integer (INT) data type and is not marked as required (nullable).
Airport_Gate	VARCHAR	The "Airport_Gate" attribute stores textual information related to the airport gate used by the airline. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Num_Employees	INT	This attribute represents the number of employees working for the airline. It is of integer (INT) data type and is required, meaning it cannot be null.

```
-- Table mydb.Passenger
DROP TABLE IF EXISTS mydb.Passenger;
CREATE TABLE IF NOT EXISTS mydb.Passenger (
```

```
    Passenger_ID INT NOT NULL,
    Passenger_FName VARCHAR(45) NULL,
    Passenger_LName VARCHAR(45) NULL,
    Passenger_DOB DATETIME NULL,
    Passenger_Address VARCHAR(45) NULL,
    Passenger_Street VARCHAR(45) NULL,
    Passenger_City VARCHAR(45) NULL,
    Passenger_Zip INT NULL,
    Passenger_Phone VARCHAR(45) NULL,
    PRIMARY KEY (Passenger_ID)
);
```

Table: Passenger

Attribute	Data Type	Description
Passenger_ID	INT	This attribute serves as the primary key for the "Passenger" table, uniquely identifying each passenger record. It is an integer (INT) data type and is required, meaning it cannot be null.
Passenger_FName	VARCHAR	The "Passenger_FName" attribute represents the first name of the passenger. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Passenger_LName	VARCHAR	This attribute is used to store the last name of the passenger. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Passenger_DOB	DATETIME	The "Passenger_DOB" attribute is used to store the date of birth of the passenger. It is of DATETIME data type and is not marked as required (nullable).

Passenger_Address	VARCHAR	This attribute stores the primary address of the passenger. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Passenger_Street	VARCHAR	The "Passenger_Street" attribute represents the street information associated with the passenger's address. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Passenger_City	VARCHAR	This attribute is used to store the city information associated with the passenger's address. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Passenger_Zip	INT	The "Passenger_Zip" attribute is used to store the zip code associated with the passenger's address. It is of integer (INT) data type and is not marked as required (nullable).
Passenger_Phone	VARCHAR	This attribute represents the phone number of the passenger. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).

```
-- Table mydb.Pilot
DROP TABLE IF EXISTS mydb.Pilot;
CREATE TABLE IF NOT EXISTS mydb.Pilot (
  Pilot_ID INT NOT NULL,
```

```
    Pilot_FName VARCHAR(45) NULL,
    Pilot_LName VARCHAR(45) NULL,
    Pilot_License VARCHAR(45) NULL,
    Hire_Date DATETIME NULL,
    Pilot_DOB DATETIME NULL,
    PRIMARY KEY (Pilot_ID)
);
```

Table: Pilot

Attribute	Data Type	Description
Pilot_ID	INT	This attribute serves as the primary key for the "Pilot" table, uniquely identifying each pilot record. It is an integer (INT) data type and is required, meaning it cannot be null.
Pilot_FName	VARCHAR	The "Pilot_FName" attribute represents the first name of the pilot. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Pilot_LName	VARCHAR	This attribute is used to store the last name of the pilot. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Pilot_License	VARCHAR	The "Pilot_License" attribute represents the license information associated with the pilot. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Hire_Date	DATETIME	This attribute is used to store the hire date of the pilot. It is of

		DATETIME data type and is not marked as required (nullable).
Pilot_DOB	DATETIME	The "Pilot_DOB" attribute is used to store the date of birth of the pilot. It is of DATETIME data type and is not marked as required (nullable).

```
-- Table mydb.Des_Airport
DROP TABLE IF EXISTS mydb.Des_Airport;
CREATE TABLE IF NOT EXISTS mydb.Des_Airport (
  Des_Airport_ID INT NOT NULL,
  Des_Airport_Loc VARCHAR(45) NULL,
  Des_Arrival_Time DATETIME NULL,
  Des_Arrival_Gate VARCHAR(45) NULL,
  Des_Terminal VARCHAR(45) NULL,
  PRIMARY KEY (Des_Airport_ID)
);
```

Table: Des_Airport

Attribute	Data Type	Description
Des_Airport_ID	INT	This attribute serves as the primary key for the "Des_Airport" table, uniquely identifying each destination airport record. It is an integer (INT) data type and is required, meaning it cannot be null.
Des_Airport_Loc	VARCHAR	The "Des_Airport_Loc" attribute represents the location of the destination airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).

Des_Arrival_Time	DATETIME	This attribute is used to store the arrival time at the destination airport. It is of DATETIME data type and is not marked as required (nullable).
Des_Arrival_Gate	VARCHAR	The "Des_Arrival_Gate" attribute represents the arrival gate at the destination airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Des_Terminal	VARCHAR	This attribute is used to store the terminal information at the destination airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).

```
-- Table mydb.Org_Airport
DROP TABLE IF EXISTS mydb.Org_Airport;
CREATE TABLE IF NOT EXISTS mydb.Org_Airport (
  Org_Airport_ID INT NOT NULL,
  Org_Airport_Loc VARCHAR(45) NULL,
  Org_Departure_Time DATETIME NULL,
  Org_Departure_Gate VARCHAR(45) NULL,
  Org_Terminal VARCHAR(45) NULL,
  PRIMARY KEY (Org_Airport_ID)
);
```

Table: Org_Airport

Attribute	Data Type	Description
Org_Airport_ID	INT	This attribute serves as the primary key for the "Org_Airport" table, uniquely

		identifying each originating airport record. It is an integer (INT) data type and is required, meaning it cannot be null.
Org_Airport_Loc	VARCHAR	The "Org_Airport_Loc" attribute represents the location of the originating airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Org_Departure_Time	DATETIME	This attribute is used to store the departure time from the originating airport. It is of DATETIME data type and is not marked as required (nullable).
Org_Departure_Gate	VARCHAR	The "Org_Departure_Gate" attribute represents the departure gate at the originating airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Org_Terminal	VARCHAR	This attribute is used to store the terminal information at the originating airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).

```
-- Table mydb.Flight Attendent
DROP TABLE IF EXISTS mydb.Flight_Attendent;
CREATE TABLE IF NOT EXISTS mydb.Flight_Attendent (
  Flight_Att_ID INT NOT NULL,
  Flight_Att_FName VARCHAR(45) NULL,
  Flight_Att_LName VARCHAR(45) NULL,
  Hire_Date DATETIME NULL,
```

```
    Flight_Attendent_DOB DATETIME NULL,
    PRIMARY KEY (Flight_Att_ID)
);
```

Table: Flight_Attendent

Attribute	Data Type	Description
Flight_Att_ID	INT	This attribute serves as the primary key for the "Flight_Attendent" table, uniquely identifying each flight attendant's record. It is an integer (INT) data type and is required, meaning it cannot be null.
Flight_Att_FName	VARCHAR	The "Flight_Att_FName" attribute represents the first name of the flight attendant. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Flight_Att_LName	VARCHAR	This attribute represents the last name of the flight attendant. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Hire_Date	DATETIME	The "Hire_Date" attribute is used to store the date when the flight attendant was hired. It is of DATETIME data type and is not marked as required (nullable).
Flight_Attendant_DOB	DATETIME	This attribute represents the date of birth of the flight attendant. It is of DATETIME data type and is not marked as required (nullable).

```
-- Table mydb.Plane
DROP TABLE IF EXISTS mydb.Plane;
CREATE TABLE IF NOT EXISTS mydb.Plane (
  Plane_ID INT NOT NULL,
  Plane_Model VARCHAR(45) NULL,
  Plane_Manufacturer VARCHAR(45) NULL,
  Plane_Size VARCHAR(20),
  Plane_Engine_Power VARCHAR(20),
  PRIMARY KEY (Plane_ID),
);
```

Table: Plane

Attribute	Data Type	Description
Plane_ID	INT	This attribute serves as the primary key for the "Plane" table, uniquely identifying each plane's record. It is an integer (INT) data type and is required, meaning it cannot be null.
Plane_Model	VARCHAR	The "Plane_Model" attribute represents the model of the plane. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Plane_Manufacturer	VARCHAR	This attribute represents the manufacturer of the plane. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Plane_Size	VARCHAR	The "Plane_Size" attribute describes the size or category of the plane. It is of VARCHAR(20) data type and is not marked as required (nullable).

Plane_Engine_Power	VARCHAR	This attribute represents the engine power of the plane. It is of VARCHAR(20) data type and is not marked as required (nullable).

```
-- Table mydb.Booking
DROP TABLE IF EXISTS mydb.Booking;
CREATE TABLE IF NOT EXISTS mydb.Booking (
  Confirmation_ID INT NOT NULL,
  Booking_Date DATETIME NULL,
  Booking_Platform VARCHAR(45) NULL,
  Booking_Type VARCHAR(45) NULL,
  PRIMARY KEY (Confirmation_ID),
  FOREIGN KEY (Passenger_ID) REFERENCES Passenger (Passenger_ID),
  );
```

Table: Booking

Attribute	Data Type	Description
Confirmation_ID	INT	This attribute serves as the primary key for the "Booking" table, uniquely identifying each booking's record. It is an integer (INT) data type and is required, meaning it cannot be null.
Booking_Date	DATETIME	The "Booking_Date" attribute represents the date and time when the booking was made. It is of DATETIME data type and is not marked as required (nullable).
Booking_Platform	VARCHAR	This attribute describes the platform or source through which the booking was made. It is of VARCHAR(45) data type,

		allowing up to 45 characters, and is not marked as required (nullable).
Booking_Type	VARCHAR	The "Booking_Type" attribute specifies the type of booking made, such as "economy," "business," etc. It is of VARCHAR(45) data type and is not marked as required (nullable).
Passenger_ID	INT	This foreign key is used to establish a relationship with the "Passenger" table, specifically referencing the "Passenger_ID" attribute in the "Booking" table. It represents the passenger associated with the booking.

```
-- Table mydb.Baggage
DROP TABLE IF EXISTS mydb.Baggage;
CREATE TABLE IF NOT EXISTS mydb.Baggage (
  Baggage_ID VARCHAR(45) NOT NULL,
  Passenger_ID INT NULL,
  Baggage_Weight DECIMAL(2) NULL,
  Baggage_Cost DECIMAL(2) NULL,
  Org_Airport_Loc DATETIME NULL,
  Org_Departure_Time VARCHAR(45) NULL,
  Flight_ID INT NULL,
  Des_Airport_Loc VARCHAR(45) NULL,
  Des_Departure_Time DATETIME NULL,
  Des_Departure_Gate VARCHAR(45) NULL,
  PRIMARY KEY (Baggage_ID),
  FOREIGN KEY (Passenger_ID) REFERENCES Passenger (Passenger_ID)
);
```

Table: Baggage

Attribute	Data Type	Description
Baggage_ID	VARCHAR	This attribute serves as the primary key for the "Baggage" table, uniquely identifying each piece of baggage. It is of VARCHAR(45) data type, allowing up to 45 characters, and it is required, meaning it cannot be null.
Passenger_ID	INT	The "Passenger_ID" attribute represents the passenger associated with the baggage. It is an integer (INT) data type and is not marked as required (nullable). This attribute is linked to the "Passenger" table through a foreign key relationship.
Baggage_Weight	DECIMAL	This attribute indicates the weight of the baggage, represented as a decimal value with two decimal places. It is not marked as required (nullable).
Baggage_Cost	DECIMAL	The "Baggage_Cost" attribute specifies the cost associated with the baggage. It is represented as a decimal value with two decimal places and is not marked as required (nullable).
Org_Airport_Loc	DATETIME	This attribute represents the location of the originating airport. It is of DATETIME data type and is not marked as required (nullable). The data type choice may be inappropriate for this attribute, and it might need to be adjusted based on the actual data it represents.
Org_Departure_Time	VARCHAR	The "Org_Departure_Time" attribute denotes the departure

		time of the flight from the originating airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Flight_ID	INT	This attribute represents the ID of the flight associated with the baggage. It is an integer (INT) data type and is not marked as required (nullable).
Des_Airport_Loc	VARCHAR	The "Des_Airport_Loc" attribute specifies the location of the destination airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).
Des_Departure_Time	DATETIME	This attribute represents the departure time of the flight from the destination airport. It is of DATETIME data type and is not marked as required (nullable).
Des_Departure_Gate	VARCHAR	The "Des_Departure_Gate" attribute indicates the departure gate at the destination airport. It is of VARCHAR(45) data type, allowing up to 45 characters, and is not marked as required (nullable).

```
-- Table mydb.Flight
DROP TABLE IF EXISTS mydb.Flight;
CREATE TABLE IF NOT EXISTS mydb.Flight (
  Flight_ID INT NOT NULL,
  Org_Departure_Time DATETIME NULL,
  Des_Arrival_Time DATETIME NULL,
```

```
  Flight_Time DECIMAL(2) NULL,

  PRIMARY KEY (Flight_ID),

  FOREIGN KEY (Passenger_ID) REFERENCES Passenger(Passenger_ID),

  FOREIGN KEY (Confirmation_ID) REFERENCES
Booking(Confirmation_ID),

  FOREIGN KEY (Org_Airport_ID) REFERENCES Org_Airport
(Org_Airport_ID),

  FOREIGN KEY (Des_Airport_ID) REFERENCES Des_Airport
(Des_Airport_ID),

  FOREIGN KEY (Plane_ID) REFERENCES Plane (Plane_ID),

  FOREIGN KEY (Pilot_ID) REFERENCES Pilot (Pilot_ID),

  FOREIGN KEY (Flight_Att_ID) REFERENCES Flight_Attendent
(Flight_Att_ID)

);
```

Table: Flight

Attribute	Data Type	Description
Flight_ID	INT	This attribute serves as the primary key for the "Flight" table, uniquely identifying each flight. It is of INT data type, and it is required, meaning it cannot be null.
Org_Departure_Time	DATETIME	The "Org_Departure_Time" attribute represents the departure time of the flight from the originating airport. It is of DATETIME data type and is not marked as required (nullable).
Des_Arrival_Time	DATETIME	This attribute denotes the arrival time of the flight at the destination airport. It is of DATETIME data type and is not marked as required (nullable).
Flight_Time	DECIMAL	The "Flight_Time" attribute specifies the duration of the flight in decimal format with two decimal places. It is not marked as required (nullable).

Passenger_ID	INT	This foreign key establishes a relationship with the "Passenger" table, specifically referencing the "Passenger_ID" attribute in the "Flight" table. It associates flights with the corresponding passenger.
Confirmation_ID	INT	This foreign key relates to the "Confirmation_ID" attribute in the "Booking" table, allowing flights to be associated with specific bookings.
Org_Airport_ID	INT	This foreign key links to the "Org_Airport_ID" attribute in the "Org_Airport" table, indicating the originating airport of the flight.
Des_Airport_ID	INT	This foreign key references the "Des_Airport_ID" attribute in the "Des_Airport" table, specifying the destination airport of the flight.
Plane_ID	INT	This foreign key is associated with the "Plane_ID" attribute in the "Plane" table, allowing flights to be linked to specific planes.
Pilot_ID	INT	This foreign key relates to the "Pilot_ID" attribute in the "Pilot" table, indicating the pilot responsible for the flight.
Flight_Att_ID	INT	This foreign key establishes a relationship with the "Flight_Attendent" table, specifically referencing the "Flight_Att_ID" attribute in the "Flight" table. It associates flight attendants with specific flights.

```
-- Table mydb.Flight_Attendent_has_Flight
DROP TABLE IF EXISTS mydb.Flight_Attendent_has_Flight;
```

```
CREATE TABLE IF NOT EXISTS mydb.Flight_Attendent_has_Flight (

  PRIMARY KEY (Flight_Att_ID, Flight_ID, Org_Airport_ID,
Des_Airport_ID, Pilot_ID, Plane_ID),

  FOREIGN KEY (Flight_Att_ID) REFERENCES Flight_Attendent
(Flight_Att_ID),

  FOREIGN KEY (Flight_ID) REFERENCES Flight (Flight_ID),

  FOREIGN KEY (Org_Airport_ID) REFERENCES Org_Airport
(Org_Airport_ID),

  FOREIGN KEY (Des_Airport_ID) REFERENCES Des_Airport
(Des_Airport_ID),

  FOREIGN KEY (Pilot_ID) REFERENCES Pilot (Pilot_ID),

  FOREIGN KEY (Plane_ID) REFERENCES Plane (Plane_ID)

);
```

Table: Flight_Attendent_has_Flight

Attribute	Data Type	Description
Flight_Att_ID	INT	This attribute is part of the primary key and establishes a relationship with the "Flight_Attendent" table. It uniquely identifies the flight attendant involved in the flight.
Flight_ID	INT	This attribute is part of the primary key and references the "Flight" table. It uniquely identifies the flight in which the flight attendant is assigned.
Org_Airport_ID	INT	This foreign key is associated with the "Org_Airport" table, indicating the originating airport of the flight.
Des_Airport_ID	INT	This foreign key references the "Des_Airport" table, specifying the destination airport of the flight.
Pilot_ID	INT	This foreign key establishes a relationship with the "Pilot" table, indicating the pilot responsible for the flight.
Plane_ID	INT	This foreign key is associated with the "Plane" table, allowing

		the association of flight attendants with specific planes used for flights.

11. Loading data and performance enhancements

Handling foreign key constraints

We used the following steps:

```
SET FOREIGN_KEY_CHECKS=0;

-- insert into Cost

INSERT INTO mydb.Cost (Cost_ID, Amount, Distance, Season_Of_Year,
Day_Of_Week, Seat_Number)

VALUES

      (1, 50, 100, 'Summer', 'Monday', 'A1'),

      (2, 60, 150, 'Spring', 'Wednesday', 'B3'),

      (3, 45, 180, 'Fall', 'Friday', 'C7'),

      (4, 55, 75, 'Winter', 'Tuesday', 'D2'),

      (5, 70, 750, 'Summer', 'Thursday', 'E5'),

      (6, 65, 1000, 'Spring', 'Monday', 'F9'),

      (7, 40, 325, 'Fall', 'Sunday', 'G4'),

      (8, 75, 650, 'Winter', 'Saturday', 'H6'),

      (9, 58, 3400, 'Summer', 'Tuesday', 'I8'),

      (10, 68, 140, 'Spring', 'Thursday', 'J10');

SET FOREIGN_KEY_CHECKS=1;
```

We set the foreign key checks to zero temporarily, so we could perform the insert operation without MySQL raising errors about the foreign key violations. Then we set it back to one to minimize the chance of data inconsistency in the future.

Importing data

Here is sample of data insertion:

```
-- Booking

INSERT INTO mydb.Booking (Confirmation_ID, Booking_Date,
Booking_Platform, Booking_Type, Passenger_ID)

      VALUES

      (1, '2023-10-31 10:00:00', 'Website', 'One-way', 101),

      (2, '2023-11-01 14:30:00', 'Mobile App', 'Round-trip', 102),
```

```
(3, '2023-11-02 12:45:00', 'Website', 'One-way', 103),
(4, '2023-11-03 08:15:00', 'Mobile App', 'Round-trip', 104),
(5, '2023-11-04 16:20:00', 'Website', 'One-way', 105),
(6, '2023-11-05 11:30:00', 'Mobile App', 'Round-trip', 106),
(7, '2023-11-06 09:10:00', 'Website', 'One-way', 107),
(8, '2023-11-07 15:55:00', 'Mobile App', 'Round-trip', 108),
(9, '2023-11-08 13:25:00', 'Website', 'One-way', 109),
(10, '2023-11-09 17:40:00', 'Mobile App', 'Round-trip', 110);
```

Insertion optimization

Timing (as measured by the server):
Execution time: 0:00:0.00031280
Table lock wait time: 0:00:0.00000300

Normalization Check

All tables are in 1NF, 2NF and 3NF

12. Application Development

Graphical User Interface Design

Here's a brief outline of the capabilities provided in each page and the connections between them:

Start:

This is the initial page, likely serving as the entry point for the application. It doesn't have specific functionality but serves as a starting point for users.

Login Page:

Users can input their username and password to access the application. It serves as an authentication barrier for the application, ensuring only authorized users can proceed.

Main Menu:

After successful login, users are directed to the main menu. It likely provides options for various actions, including adding or removing flights, making bookings, and more.

Add Flight:

This page allows authorized users to input information for adding a new flight to the system. This is a feature for administrators or flight operators to update flight details.

Administration Set Password:

This page might be used by administrators to set or change passwords for authorized users or other administrators. It's a security and user management feature.

Remove Flight:

Allows administrators to remove flights from the system. It's a feature for managing flight data.

Make Booking:

Provides a form or interface for users to make flight bookings. Likely used by passengers or customers.

Search Flight:

Users can search for available flights based on specific criteria, like destination, date, or flight number. It assists users in finding suitable flights.

Delete Booking:

Allows users to cancel or delete their existing flight bookings. Provides a means for passengers to manage their bookings.

Change Password:

Enables users to change their own passwords for security reasons. Part of user account management.

Display Flight Details:

Likely shows detailed information about a specific flight, such as departure and arrival times, seat availability, and pricing. Helps users get more information about a particular flight.

Authorized?

It appears to be a validation step, where the application checks if the user is authorized. If authorization is successful, the user is directed to the "Access Granted" page. Otherwise, the user might encounter further login attempts.

Enter Username & Password:

This is part of the login process where users input their credentials for authentication. If successful, they are directed to the "Authorized?" page.

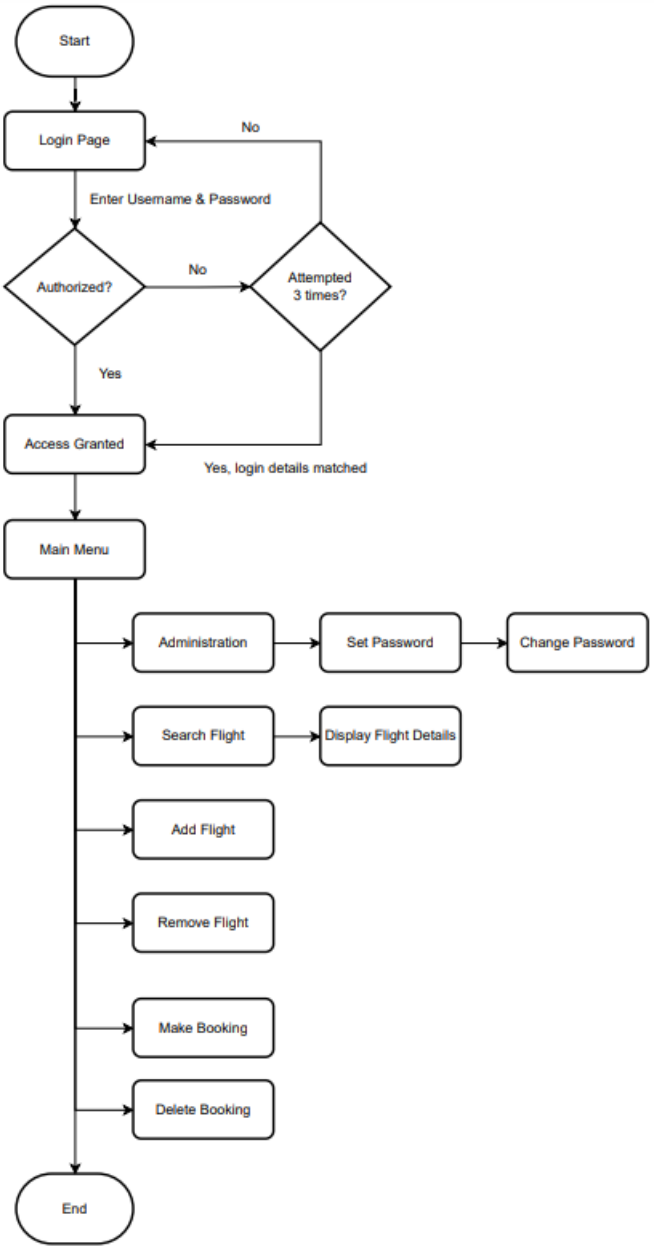
Access Granted:

This page signifies that the user has been successfully authenticated and authorized. It serves as a gateway to the Main Menu or other sections of the application.

Yes / No Attempted:

These options might be presented to users when authentication fails. "Yes" could indicate a retry, while "No" might lead to exiting the application or other appropriate actions.

Flowchart



View's implementation

Administrator View (Admin_View):

This view provides information relevant to administrators, such as passenger details and flight times for bookings. It selects data from the Booking, Passenger, and Flight tables.

```
-- Administrator view
CREATE VIEW Admin_View AS
SELECT
    Booking.Confirmation_ID,
    Passenger.Passenger_FName,
    Passenger.Passenger_LName,
    Flight.Org_Departure_Time,
    Flight.Des_Arrival_Time
FROM
    Booking
JOIN
    Passenger ON Booking.Passenger_ID = Passenger.Passenger_ID
JOIN
    Flight ON Booking.Confirmation_ID = Flight.Confirmation_ID;

select * from Admin_View;
```

Add Booking View (Add_Booking_View):

This view is likely used for adding bookings, providing information about the booking, passenger, flight, and associated costs. It selects data from the Booking, Passenger, Flight, and Cost tables.

```
-- Add Booking view
CREATE VIEW Add_Booking_View AS
SELECT
    Booking.Confirmation_ID,
    Passenger.Passenger_FName,
    Passenger.Passenger_LName,
    Flight.Org_Departure_Time AS Departure_Time,
```

```
    Flight.Des_Arrival_Time AS Arrival_Time,
    Cost.Amount AS Booking_Cost,
    Cost.Distance,
    Cost.Season_Of_Year,
    Cost.Day_Of_Week
FROM
    Booking
JOIN
    Passenger ON Booking.Passenger_ID = Passenger.Passenger_ID
JOIN
    Flight ON Booking.Confirmation_ID = Flight.Confirmation_ID
JOIN
    Cost ON Booking.Confirmation_ID = Cost.Cost_ID;
```

Delete Booking View (Delete_Booking_View):

This view is designed for deleting bookings and displays details about the booking, passenger, flight, and booking cost. It selects data from the Booking, Passenger, Flight, and Cost tables.

```
-- Delete Booking view
CREATE VIEW Delete_Booking_View AS
SELECT
    Booking.Confirmation_ID,
    Booking.Booking_Date,
    Booking.Booking_Platform,
    Booking.Booking_Type,
    Passenger.Passenger_ID,
    Passenger.Passenger_FName,
    Passenger.Passenger_LName,
    Flight.Flight_ID,
    Flight.Org_Departure_Time,
    Flight.Des_Arrival_Time,
```

```
    Cost.Amount AS Booking_Cost
FROM
    Booking
JOIN
    Passenger ON Booking.Passenger_ID = Passenger.Passenger_ID
JOIN
    Flight ON Booking.Confirmation_ID = Flight.Confirmation_ID
JOIN
    Cost ON Booking.Confirmation_ID = Cost.Cost_ID;
```

Add Flight View (Add_Flight_View):

This view offers information about flights, including details about the flight, pilot, plane, airports, and flight attendants. It selects data from the Flight table, as well as several other related tables via LEFT JOIN operations.

```
    -- Add Flight view
CREATE VIEW Add_Flight_View AS
SELECT
    Flight.Flight_ID,
    Flight.Org_Departure_Time,
    Flight.Des_Arrival_Time,
    Flight.Flight_Time,
    Pilot.Pilot_ID,
    Pilot.Pilot_FName,
    Pilot.Pilot_LName,
    Plane.Plane_ID,
    Plane.Plane_Model,
    Plane.Plane_Manufacturer,
    Org_Airport.Org_Airport_ID AS Org_Airport_ID,
    Org_Airport.Org_Airport_Loc AS Org_Airport_Location,
    Des_Airport.Des_Airport_ID AS Des_Airport_ID,
```

```
    Des_Airport.Des_Airport_Loc AS Des_Airport_Location,
    Flight_Attendent.Flight_Att_ID AS Flight_Attendant_ID,
    Flight_Attendent.Flight_Att_FName AS Flight_Attendant_FName,
    Flight_Attendent.Flight_Att_LName AS Flight_Attendant_LName
FROM
    Flight
LEFT JOIN
    Pilot ON Flight.Pilot_ID = Pilot.Pilot_ID
LEFT JOIN
    Plane ON Flight.Plane_ID = Plane.Plane_ID
LEFT JOIN
    Org_Airport ON Flight.Org_Airport_ID = Org_Airport.Org_Airport_ID
LEFT JOIN
    Des_Airport ON Flight.Des_Airport_ID = Des_Airport.Des_Airport_ID
LEFT JOIN
    Flight_Attendent ON Flight.Flight_Att_ID =
Flight_Attendent.Flight_Att_ID;
```

Remove Flight View (Remove_Flight_View):

This view is likely used to remove flights and provides information about flights, pilots, planes, and airport locations. It selects data from the Flight table, as well as other related tables via LEFT JOIN operations.

```
-- Remove Flight view
CREATE VIEW Remove_Flight_View AS
SELECT
    Flight.Flight_ID,
    Flight.Org_Departure_Time,
    Flight.Des_Arrival_Time,
    Pilot.Pilot_ID,
    Pilot.Pilot_FName,
    Pilot.Pilot_LName,
```

```
    Plane.Plane_ID,
    Plane.Plane_Model,
    Org_Airport.Org_Airport_ID AS Org_Airport_ID,
    Org_Airport.Org_Airport_Loc AS Org_Airport_Location,
    Des_Airport.Des_Airport_ID AS Des_Airport_ID,
    Des_Airport.Des_Airport_Loc AS Des_Airport_Location
FROM
    Flight
LEFT JOIN
    Pilot ON Flight.Pilot_ID = Pilot.Pilot_ID
LEFT JOIN
    Plane ON Flight.Plane_ID = Plane.Plane_ID
LEFT JOIN
    Org_Airport ON Flight.Org_Airport_ID = Org_Airport.Org_Airport_ID
LEFT JOIN
    Des_Airport ON Flight.Des_Airport_ID = Des_Airport.Des_Airport_ID;
```

13. References

1. Airline Suite. (n.d.). *Airline Suite Reviews - Pros & Cons, Ratings & More*. GetApp. https://www.getapp.com/operations-management-software/a/airline-suite/reviews/
2. *Aviation Maintenance Operations & Aircraft Management Software: RAMCO systems*. Aviation Maintenance Operations & Aircraft Management Software | Ramco Systems. (n.d.). https://www.ramco.com/products/aviation-software/?utm_source=capterra&utm_medium=capterra_ppc&utm_campaign=capterra_aviation&utm_channel=capterra
3. AvPro. (n.d.-a). *Aircraft Maintenance Software: Wellington FL: AvPro software*. avpro. https://www.avprosoftware.com/
4. AvPro. (n.d.). *Fly Safer. work smarter.* C.A.L.M. Systems | Airline Suite | Aviation Management Software. https://info.gartnerdigitalmarkets.com/calm-systems-gdm-lp?utm_source=capterra

5. Trustradius. (2021, June 16). *Pros and cons of Ramco ERP 2023*. TrustRadius. https://www.trustradius.com/products/ramco-erp/reviews?qs=pros-and-cons#reviews