Online Aeroplane BOOKING system

RAGHURAJ PRATAP SINGH---23SCSE1180562
MUKUND RAI---
ATUL KUMAR DUBEY----
PRAKHAR PRATAP SINGH-----

How to Book an Airplane Ticket Online?

CHOOSE A BOOKING PLATFORM

START BY SELECTING A PLATFORM TO BOOK YOUR FLIGHT. YOU HAVE TWO MAIN OPTIONS:

1. **AIRLINE WEBSITES**: *VISIT THE OFFICIAL WEBSITE OF THE AIRLINE YOU PREFER, LIKE AMERICAN AIRLINES, DELTA, AIR INDIA OR EMIRATES.*
2. **TRAVEL BOOKING WEBSITES**: *USE THIRD-PARTY TRAVEL PLATFORMS LIKE EXPEDIA, KAYAK, OR GOOGLE FLIGHTS TO COMPARE PRICES ACROSS AIRLINES AND FIND THE BEST DEALS.*

Enter Your Travel Information

- *Once on your chosen website or app, enter your travel details:*
- **Departure and Destination Airports**: *Select your starting airport and your destination.*
- **Travel Dates**: *Pick your departure date and, if booking a round trip, your return date.*
- **Number of Passengers**: *Choose the number of travelers and specify adults, children, or infants.*
- **Class Preference** *(Optional): If available, choose a cabin class (Economy, Premium Economy, Business, First Clas*

BROWSE AND SELECT A FLIGHT

- *The site will show you various flight options based on your search. Here's what to consider:*
- **Price**: *Check if the price fits your budget.*
- **Flight Duration**: *Direct flights are faster but may cost more.*
- **Layovers**: *If there are layovers, note the location and duration. Some prefer direct flights, while others are fine with layovers for a lower fare.*
- **Departure and Arrival Times**: *Choose timings that fit your schedule.*

challenges

- **Issues in Existing Booking Systems**

1. **Long Queues and Waiting Times**
    1. *Traditional booking methods require users to wait in physical lines or deal with time-consuming processes.*
2. **Limited Accessibility and Outdated Interfaces**
    1. *Many existing systems are not mobile-friendly or user-friendly, making navigation cumbersome.*
3. **Lack of Real-Time Updates**
    1. *Information on seat availability and pricing is often outdated, leading to booking errors or delays.*
4. **Security Concerns in Payment and Data Management**
    1. *Insufficient protection of sensitive user data can lead to fraud and unauthorized access.*

Our goal

- *Simplify the flight booking process with an intuitive interface.*
- *Provide secure and seamless payment options.*
- *Offer real-time updates on flight availability and pricing.*
- *Enable users to manage bookings and cancellations easily.*
- *Include an admin panel for managing flight details.*

A system flow

- *A diagram showing:User Interaction: Search flights, book tickets, make payments.*
- *Backend Processes: APIs, payment gateway, and database.*
- *Database Layers: Storing user, flight, and booking details*

Key feature

1. **Flight Search and Filter**: *Search flights by destination, date, airline, etc.*
2. **Seat Availability**: *Real-time updates on available seats.*
3. **Booking and Payment**: *Secure payment integration with receipt generation.*
4. **User Account Management**: *Login, registration, and booking history.*
5. **Admin Panel**: *Manage flight schedules, pricing, and user bookings.*

TECH STACK
HTML PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Airplane Booking</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            background: #f4f4f9;
        }
        .container {
            max-width: 600px;
            margin: 50px auto;
            padding: 20px;
            background: #fff;
            text-align: center;
            border-radius: 8px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
```

HTML CODE

```
font-size: 24px;
            color: #333;
        }
        input, select, button {
            width: 100%;
            margin: 10px 0;
            padding: 10px;
            font-size: 16px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            background: #007bff;
            color: white;
            border: none;
            cursor: pointer;
        }
        button:hover {
            background: #0056b3;
        }
        .hidden {
            display: none;
```

HTML CODE

```
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Airplane Booking</h1>
        <div id="login-section">
            <h2>Login</h2>
            <input id="email" type="email" placeholder="Email">
            <input id="password" type="password" placeholder="Password">
            <button onclick="login()">Login</button>
        </div>
        <div id="booking-section" class="hidden">
            <h2>Book a Flight</h2>
            <input id="destination" type="text" placeholder="Destination">
            <input id="date" type="date">
            <select id="class">
                <option value="economy">Economy</option>
                <option value="business">Business</option>
                <option value="first">First Class</option>
            </select>
            <h3>Payment Details</h3>
```

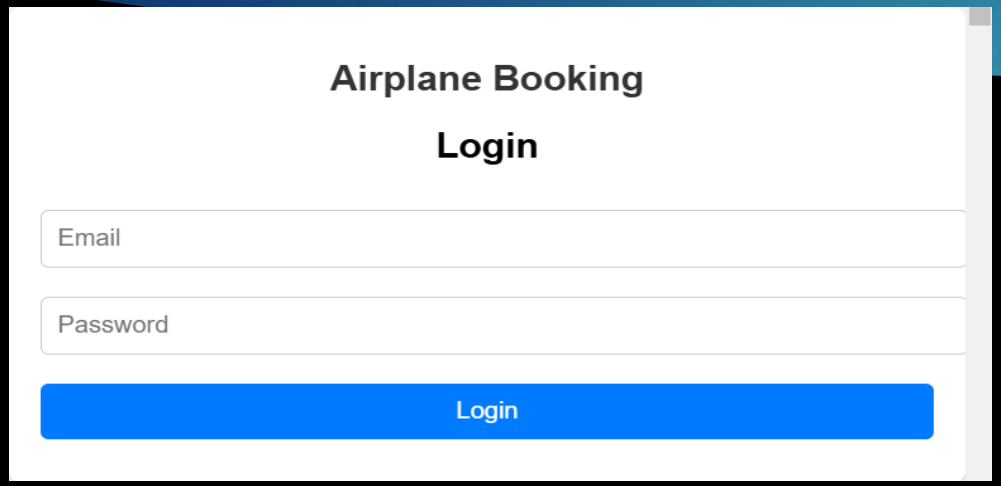
HTML CODE

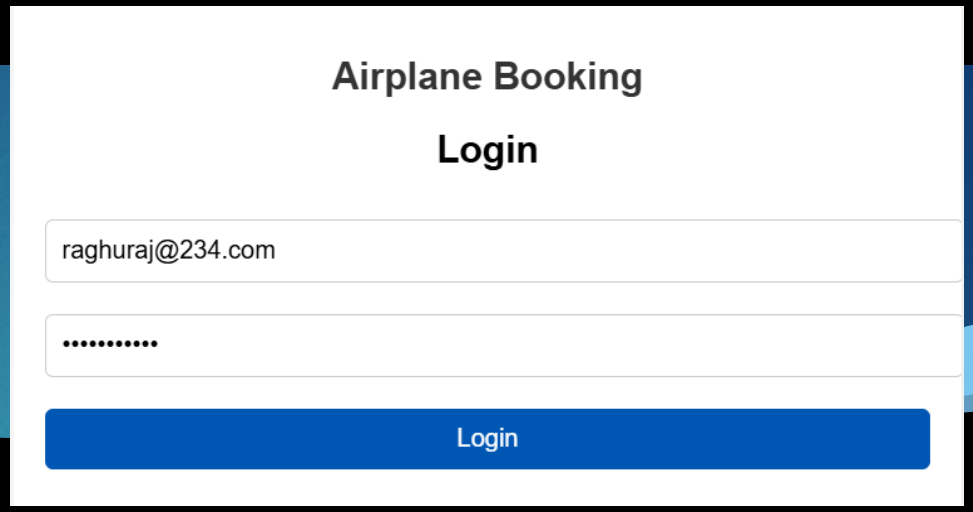
```
<input id="card-name" type="text" placeholder="Card Name">
        <input id="card-number" type="text" placeholder="Card Number">
        <input id="expiry" type="text" placeholder="Expiry (MM/YY)">
        <input id="cvv" type="text" placeholder="CVV">
        <button onclick="book()">Book & Pay</button>
      </div>
    </div>

    <script>
      const validUser = { email: "raghuraj@234.com", password: "password123" };
      function login() {
        const email = document.getElementById("email").value;
        const password = document.getElementById("password").value;
        if (email === validUser.email && password === validUser.password) {
          alert("Login Successfull");
          document.getElementById("login-section").classList.add("hidden");
          document.getElementById("booking-section").classList.remove("hidden");
        } else {
          alert("Invalid Credentials");
        }
      }
      function book() {
```

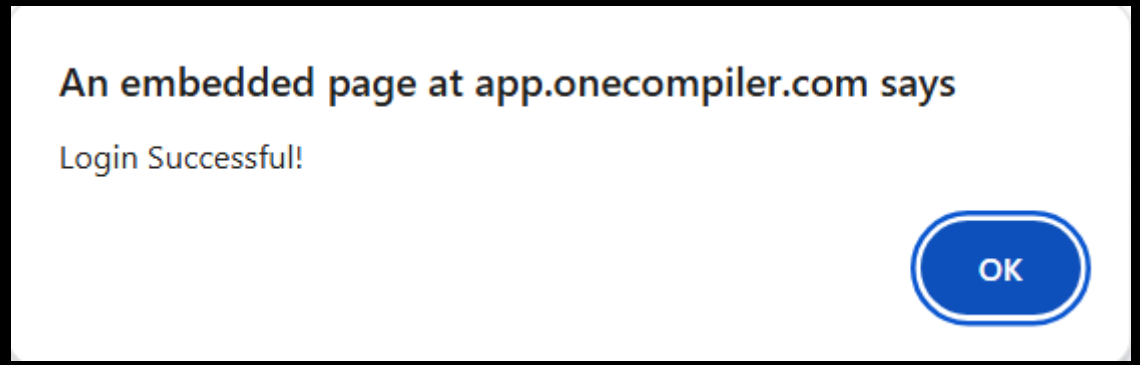
HTML CODE

```
const destination = document.getElementById("destination").value;
        const date = document.getElementById("date").value;
        const flightClass = document.getElementById("class").value;
        const cardName = document.getElementById("card-name").value;
        const cardNumber = document.getElementById("card-number").value;
        const expiry = document.getElementById("expiry").value;
        const cvv = document.getElementById("cvv").value;
        if (destination && date && flightClass && cardName && cardNumber && expiry && cvv) {
            alert(`Flight booked successfully!\nDestination: ${destination}\nDate: ${date}\nClass: ${flightClass}\nPayment by: ${cardName}`);
        } else {
            alert("Please fill all fields.");
        }
    }
    </script>
</body>
</html>
```

OUTPUT

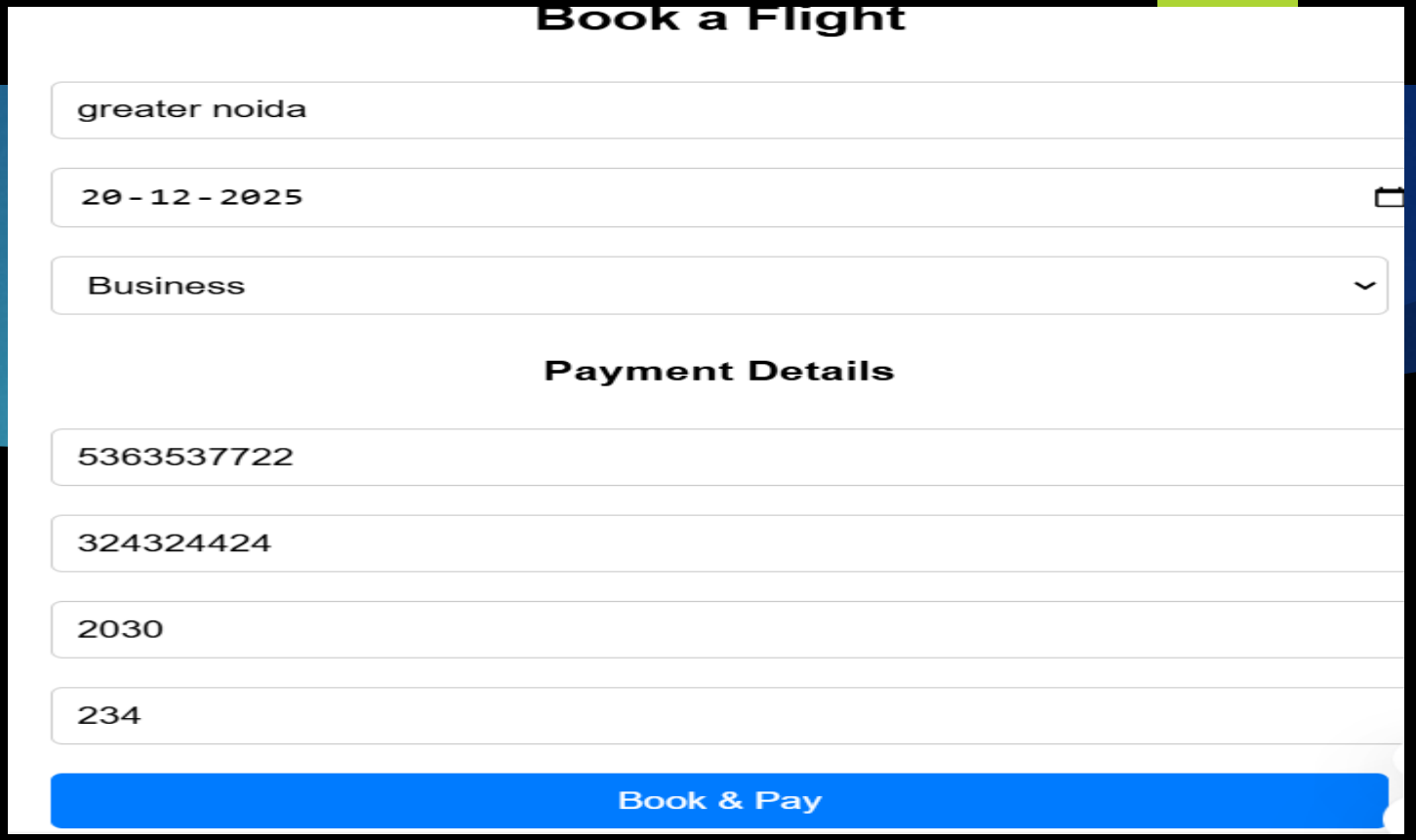

Airplane Booking
Login
Email
Password
Login

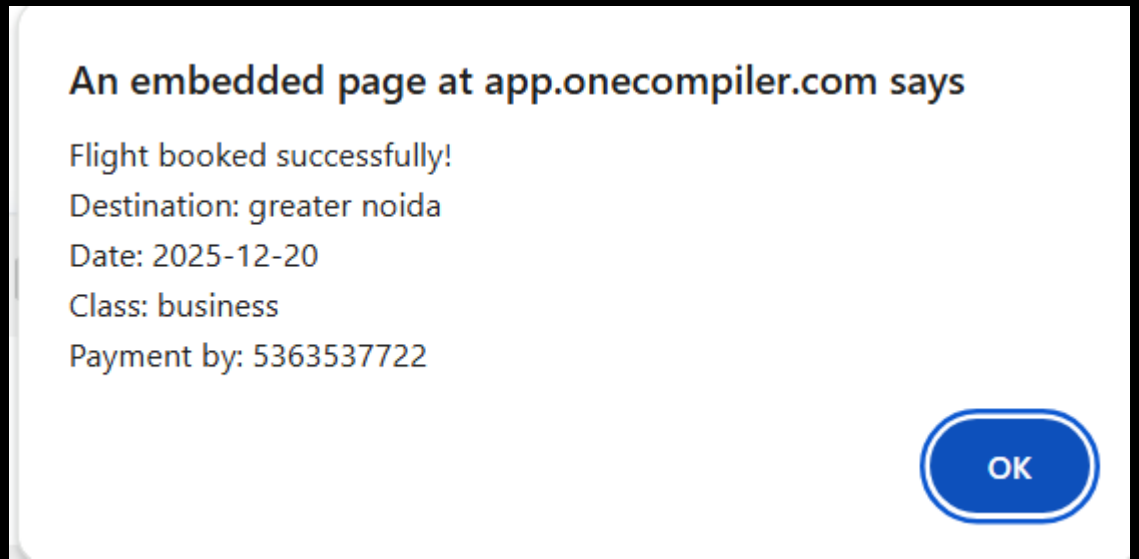

Airplane Booking
Login
raghuraj@234.com
Login


An embedded page at app.onecompiler.com says
Login Successful!
OK

OUTPUT


Airplane Booking
Book a Flight
Destination
dd - mm - yyyy
Economy
Payment Details
Card Name
Card Number
Expiry (MM/YY)


Book a Flight
greater noida
20 - 12 - 2025
Business
Payment Details
5363537722
324324424
2030
234
Book & Pay


An embedded page at app.onecompiler.com says
Flight booked successfully!
Destination: greater noida
Date: 2025-12-20
Class: business
Payment by: 5363537722
OK

JAVA CODE FOR FLIGHT BOOING

```
import java.util.*;
 class Flight {
    private String flightNumber;
    private String source;
    private String destination;
    private double price;
    private int availableSeats;

    public Flight(String flightNumber, String source, String destination, double price, int availableSeats) {
        this.flightNumber = flightNumber;
        this.source = source;
        this.destination = destination;
        this.price = price;
        this.availableSeats = availableSeats;
    }

    public String getFlightNumber() {
        return flightNumber;
    }

    public String getSource() {
        return source;
```

JAVA CODE

```
    }

    public String getDestination() {
        return destination;
    }

    public double getPrice() {
        return price;
    }

    public int getAvailableSeats() {
        return availableSeats;
    }

    public boolean bookSeat() {
        if (availableSeats > 0) {
            availableSeats--;
            return true;
        } else {
            return false;
        }
    }
```

JAVA CODE

```
}

class Booking {
    private String customerName;
    private Flight flight;
    private int numberOfSeats;

    public Booking(String customerName, Flight flight, int numberOfSeats) {
        this.customerName = customerName;
        this.flight = flight;
        this.numberOfSeats = numberOfSeats;
    }

    public void displayBookingDetails() {
        System.out.println("Booking Confirmation:");
        System.out.println("Customer Name: " + customerName);
        System.out.println("Flight Number: " + flight.getFlightNumber());
        System.out.println("From: " + flight.getSource());
        System.out.println("To: " + flight.getDestination());
        System.out.println("Seats Booked: " + numberOfSeats);
        System.out.println("Total Price: " + (numberOfSeats * flight.getPrice()));
    }
```

JAVA CODE

```
}

public class Main {
    private static List<Flight> flights = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        // Sample Flights
        flights.add(new Flight("AI101", "New York", "London", 500.0, 50));
        flights.add(new Flight("AI102", "London", "Paris", 300.0, 60));
        flights.add(new Flight("AI103", "Paris", "Tokyo", 800.0, 40));

        while (true) {
            System.out.println("\nAirplane Booking System");
            System.out.println("1. View Available Flights");
            System.out.println("2. Book a Flight");
            System.out.println("3. Exit");
            System.out.print("Enter choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();  // Consume newline

            switch (choice) {
```

JAVA CODE

```
            case 1:
                    viewAvailableFlights();
                    break;
                case 2:
                    bookAFlight();
                    break;
                case 3:
                    System.out.println("Exiting system...");
                    return;
                default:
                    System.out.println("Invalid choice, please try again.");
            }
        }
    }

    public static void viewAvailableFlights() {
        System.out.println("\nAvailable Flights:");
        for (Flight flight : flights) {
            System.out.println("Flight Number: " + flight.getFlightNumber());
            System.out.println("From: " + flight.getSource());
            System.out.println("To: " + flight.getDestination());
            System.out.println("Price: " + flight.getPrice());
```

JAVA CODE

```
            System.out.println("Available Seats: " + flight.getAvailableSeats());
                System.out.println("-------------------------");
            }
        }

        public static void bookAFlight() {
            System.out.print("Enter Customer Name: ");
            String customerName = scanner.nextLine();

            System.out.print("Enter Flight Number to book: ");
            String flightNumber = scanner.nextLine();

            Flight selectedFlight = null;
            for (Flight flight : flights) {
                if (flight.getFlightNumber().equals(flightNumber)) {
                    selectedFlight = flight;
                    break;
                }
            }

            if (selectedFlight != null && selectedFlight.getAvailableSeats() > 0) {
                System.out.print("Enter number of seats to book: ");
```

JAVA CODE

```
int seatsToBook = scanner.nextInt();
            if (seatsToBook <= selectedFlight.getAvailableSeats()) {
                selectedFlight.bookSeat();
                Booking booking = new Booking(customerName, selectedFlight, seatsToBook);
                booking.displayBookingDetails();
            } else {
                System.out.println("Not enough seats available.");
            }
        } else {
            System.out.println("Flight not found or no available seats.");
        }
    }
}
```

OUTPUT

Main.java

Share Run

```java
import java.util.*;
class Flight {
    private String flightNumber;
    private String source;
    private String destination;
    private double price;
    private int availableSeats;

    public Flight(String flightNumber, String source, String destination,
        double price, int availableSeats) {
        this.flightNumber = flightNumber;
        this.source = source;
        this.destination = destination;
        this.price = price;
        this.availableSeats = availableSeats;
    }

    public String getFlightNumber() {
        return flightNumber;
    }

    public String getSource() {
        return source;
    }

```

Output

```
Airplane Booking System
1. View Available Flights
2. Book a Flight
3. Exit
Enter choice:
```

Main.java

Share Run

```java
import java.util.*;
class Flight {
    private String flightNumber;
    private String source;
    private String destination;
    private double price;
    private int availableSeats;

    public Flight(String flightNumber, String source, String destination,
        double price, int availableSeats) {
        this.flightNumber = flightNumber;
        this.source = source;
        this.destination = destination;
        this.price = price;
        this.availableSeats = availableSeats;
    }

    public String getFlightNumber() {
        return flightNumber;
    }

    public String getSource() {
        return source;
    }

```

Output

```
Airplane Booking System
1. View Available Flights
2. Book a Flight
3. Exit
Enter choice: 1
```

OUTPUT

Output Clear

```
To: London
Price: 500.0
Available Seats: 50
--------------------------
Flight Number: AI102
From: London
To: Paris
Price: 300.0
Available Seats: 60
--------------------------
Flight Number: AI103
From: Paris
To: Tokyo
Price: 800.0
Available Seats: 40
--------------------------

Airplane Booking System
1. View Available Flights
2. Book a Flight
3. Exit
Enter choice: 2
Enter Customer Name: ATUL AND MUKUND
Enter Flight Number to book: AI102
```

```
Airplane Booking System
1. View Available Flights
2. Book a Flight
3. Exit
Enter choice: 2
Enter Customer Name: ATUL AND MUKUND
Enter Flight Number to book: AI102
Enter number of seats to book: 4
Booking Confirmation:
Customer Name: ATUL AND MUKUND
Flight Number: AI102
From: London
To: Paris
Seats Booked: 4
Total Price: 1200.0
```

1. Users Table
Stores information about customers and administrators.

DATABASE OF PROJECT
user database

Field Name	Data Type	Description
user_id	INT	Unique identifier for the user (Primary Key)
first_name	VARCHAR	Customer's first name
last_name	VARCHAR	Customer's last name
email	VARCHAR	User's email address
password	VARCHAR	User's password (hashed)
phone_number	VARCHAR	User's phone number
role	VARCHAR	Role of the user (admin, customer)

2. Flights Table
Stores information about available flights.

DATABASE

Field Name	Data Type	Description
flight_id	INT	Unique identifier for the flight (Primary Key)
flight_number	VARCHAR	Flight number (e.g., AA123)
airline	VARCHAR	Airline name
departure_city	VARCHAR	Departure city
arrival_city	VARCHAR	Arrival city
departure_time	DATETIME	Flight departure time
arrival_time	DATETIME	Flight arrival time
seats_available	INT	Number of available seats

Field Name	Data Type	Description
payment_id	INT	Unique identifier for the payment (Primary Key)
booking_id	INT	Associated booking ID (Foreign Key)
payment_date	DATETIME	Payment date
payment_method	VARCHAR	Payment method (Credit card, PayPal, etc.)
amount_paid	DECIMAL	Amount paid for the booking
payment_status	VARCHAR	Status of the payment (paid, pending)

5. Airports Table
Stores details about airports.

DATABASE

Field Name	Data Type	Description
airport_id	INT	Unique identifier for the airport (Primary Key)
airport_name	VARCHAR	Name of the airport
city	VARCHAR	City where the airport is located
country	VARCHAR	Country where the airport is located

Field Name	Data Type	Description
seat_id	INT	Unique identifier for the seat (Primary Key)
flight_id	INT	Flight ID the seat is associated with (Foreign Key)
seat_number	VARCHAR	Seat number (e.g., 1A, 12B)
seat_class	VARCHAR	Seat class (Economy, Business, First)
is_reserved	BOOLEAN	Indicates if the seat is reserved

What We Faced & How We Solved It

- **Challenges**:
- Handling large datasets for real-time seat availability.
- Ensuring payment security and PCI compliance.
- Designing an intuitive and responsive UI.
- **Solutions**:
- Optimized database queries and indexing.
- Integrated industry-standard payment APIs.
- Conducted user testing for UI improvements.

conclusion

- Summarize the project's value to users and administrators.
- Highlight the alignment of the system with project goals.
- Emphasize how the project addresses real-world challenges.