ABBOTTABAD UNIVERSITY OFF SCIENCE AND TECHNOLOGY

PROJECT REPORT

FINAL PROJECT

Check Github CODE

https://github.com/madimalik3119/Airport-Scheduling-Project.git

SUBMITTED TO :	**SIR JAMAL ABDUL_AHAD**
SUBMITTED BY:	**HAMMAD SADAQAT**
SUBJECT:	**DSA PROECT REPORT**
ROLL NO :	**14849**
FINAL PROJCT :	**01**
SECTION:	**BSCS 3D**
DATE OF SUBMISSION:	**26/12/2024**

PROJECT NAME

Airport Flight Scheduler – Build a flight scheduling system using dynamic programming

"This project combines HTML, CSS, and JavaScript for the frontend, with Python as the backend, to create a dynamic web application. By integrating these technologies, the project achieves a harmonious blend of interactivity, styling, and server-side logic, yielding a robust and user-friendly web application."

Flight Scheduling Project Report

Project Overview

The Flight Scheduling project is a web-based application designed to manage flight schedules. The application allows users to view, add, edit, and delete flight schedules.

Technologies Used

- HTML5 for structuring content
- CSS3 for styling and layout
- JavaScript for adding interactivity
- Python as the backend technology (for server-side logic and database integration)

Code Structure

The project consists of the following files:

- index.html: The main HTML file containing the application's UI

- styles.css: The CSS file for styling and layout

- script.js: The JavaScript file for adding interactivity

- (App.py): The Python file for server-side logic and database integration

HTML

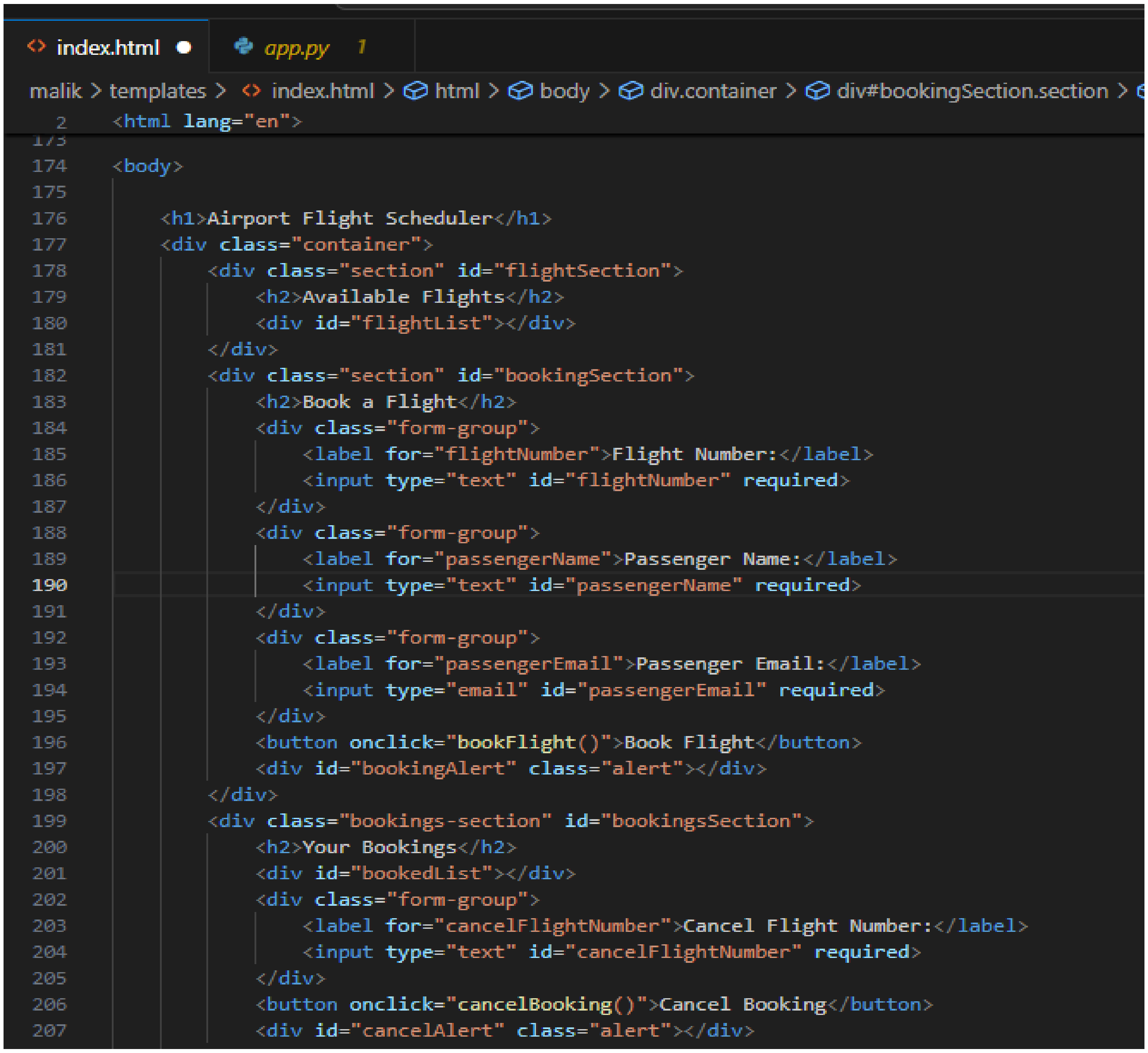
Internal CSS

Internal JAVA SCRIPT

PYTHON(beckend)

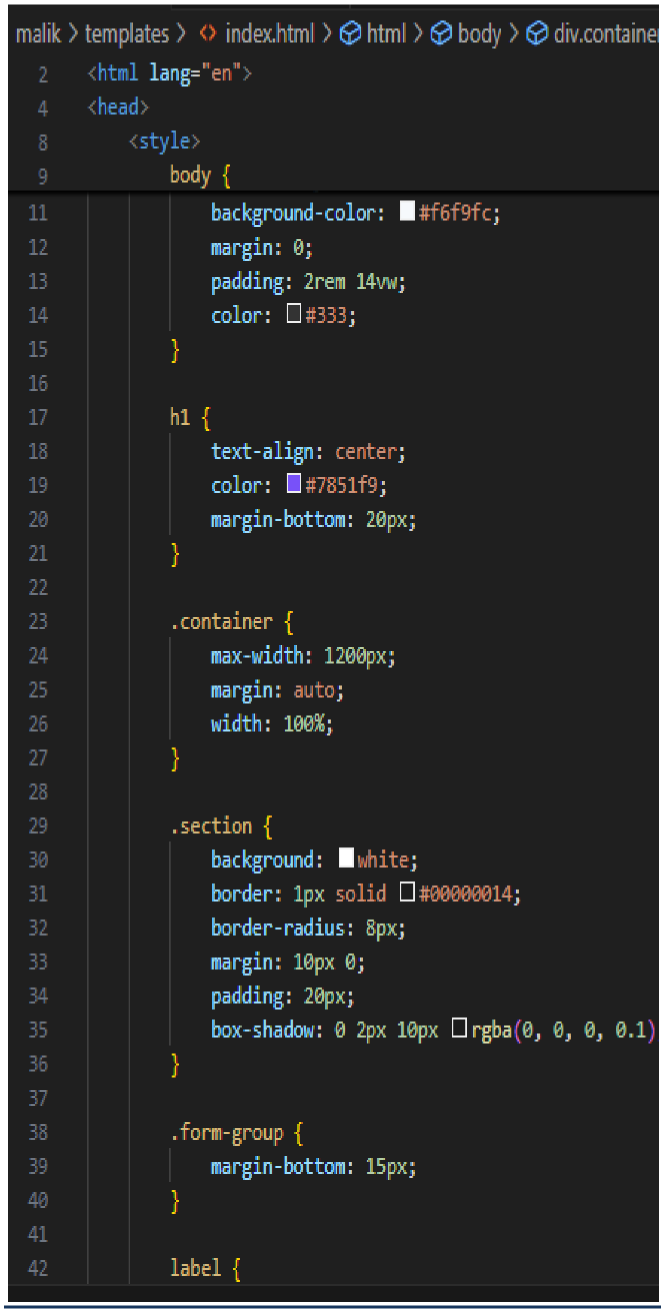
Let’s start coding

Using Html

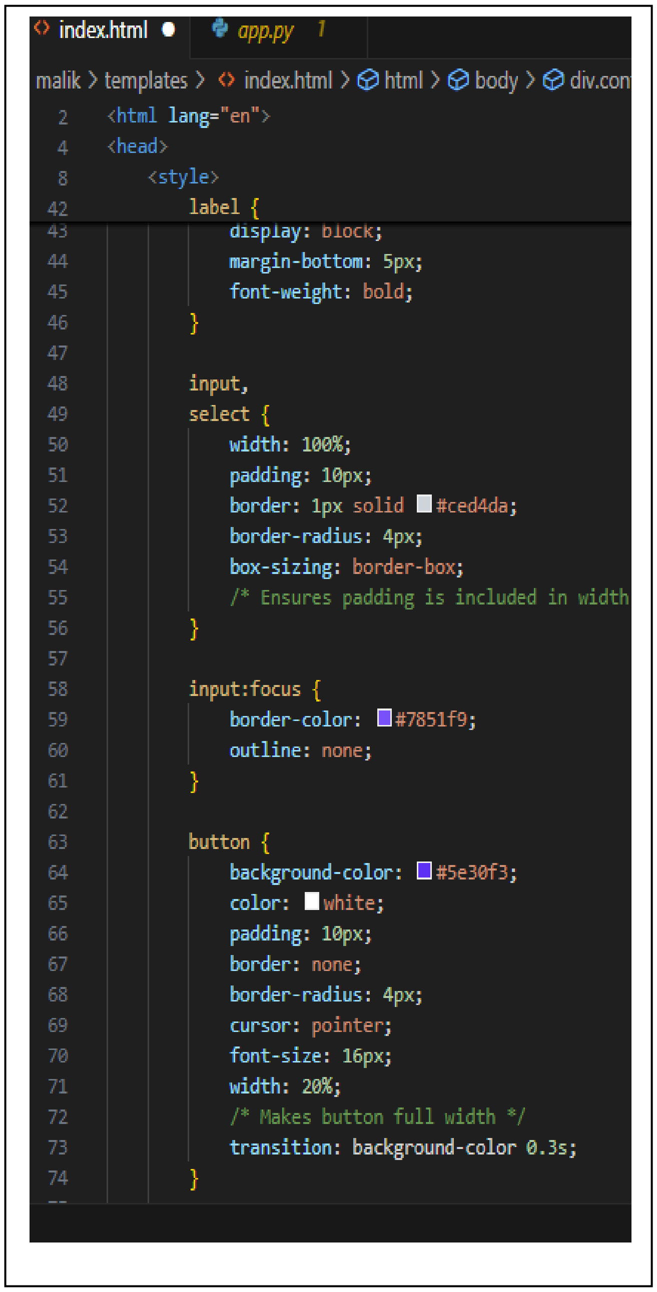

index.html
app.py 1
malik > templates > index.html > html > body > div.container > div#bookingSection.section >

```html
<html lang="en">

<body>

    <h1>Airport Flight Scheduler</h1>
    <div class="container">
        <div class="section" id="flightSection">
            <h2>Available Flights</h2>
            <div id="flightList"></div>
        </div>
        <div class="section" id="bookingSection">
            <h2>Book a Flight</h2>
            <div class="form-group">
                <label for="flightNumber">Flight Number:</label>
                <input type="text" id="flightNumber" required>
            </div>
            <div class="form-group">
                <label for="passengerName">Passenger Name:</label>
                <input type="text" id="passengerName" required>
            </div>
            <div class="form-group">
                <label for="passengerEmail">Passenger Email:</label>
                <input type="email" id="passengerEmail" required>
            </div>
            <button onclick="bookFlight()">Book Flight</button>
            <div id="bookingAlert" class="alert"></div>
        </div>
        <div class="bookings-section" id="bookingsSection">
            <h2>Your Bookings</h2>
            <div id="bookedList"></div>
            <div class="form-group">
                <label for="cancelFlightNumber">Cancel Flight Number:</label>
                <input type="text" id="cancelFlightNumber" required>
            </div>
            <button onclick="cancelBooking()">Cancel Booking</button>
            <div id="cancelAlert" class="alert"></div>
```

Internal CSS



```
<html lang="en">
<head>
    <style>
        body {
            background-color: #f6f9fc;
            margin: 0;
            padding: 2rem 14vw;
            color: #333;
        }

        h1 {
            text-align: center;
            color: #7851f9;
            margin-bottom: 20px;
        }

        .container {
            max-width: 1200px;
            margin: auto;
            width: 100%;
        }

        .section {
            background: white;
            border: 1px solid #00000014;
            border-radius: 8px;
            margin: 10px 0;
            padding: 20px;
            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1)
        }

        .form-group {
            margin-bottom: 15px;
        }

        label {
```



```
<html lang="en">
<head>
    <style>
        label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
        }

        input,
        select {
            width: 100%;
            padding: 10px;
            border: 1px solid #ced4da;
            border-radius: 4px;
            box-sizing: border-box;
            /* Ensures padding is included in width
        }

        input:focus {
            border-color: #7851f9;
            outline: none;
        }

        button {
            background-color: #5e30f3;
            color: white;
            padding: 10px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 16px;
            width: 20%;
            /* Makes button full width */
            transition: background-color 0.3s;
        }
```

Javascript code

Flight Data and Booking Management

- Defines an array of flight objects with properties like flight number, origin, destination, time, and status.

- Manages an array of booking objects, which contain flight number, passenger name, and passenger email.

Displaying Flights and Bookings

- The display Flights () function populates the #flight List div with a list of available flights, including their status and a button to check availability.

- The display Bookings () function populates the #booked List div with a list of booked flights, including passenger details.

Booking and Cancellation

- The book Flight () function books a flight by updating the flight status, adding a new booking object, and displaying a success message.

- The cancel Booking () function cancels a booking by resetting the flight status, removing the booking object, and displaying a success message.

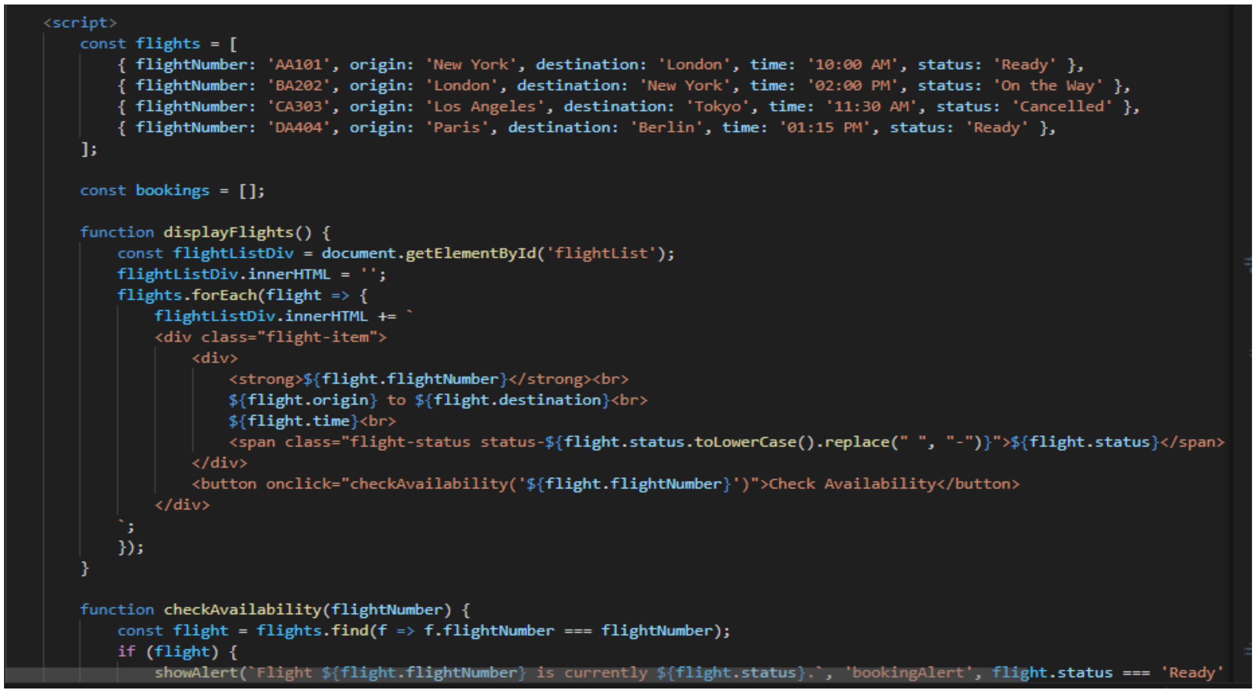
Alert and Input Management

- The show Alert () function displays alert messages with a specified type (success or error) and duration.

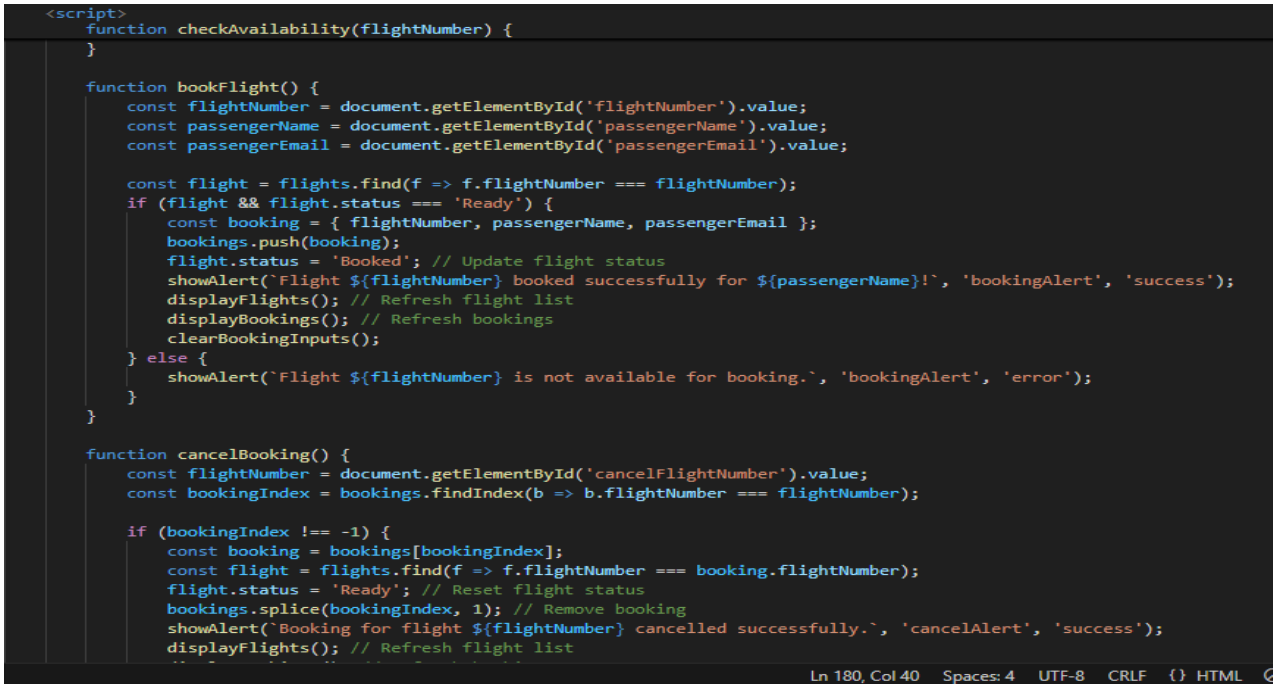
- The clearBookingInputs () and clearCancelInputs () functions clear the input fields after booking or cancellation.

Initialization

- The display Flights() function is called initially to display the list of available flights.



```
<script>
    const flights = [
        { flightNumber: 'AA101', origin: 'New York', destination: 'London', time: '10:00 AM', status: 'Ready' },
        { flightNumber: 'BA202', origin: 'London', destination: 'New York', time: '02:00 PM', status: 'On the Way' },
        { flightNumber: 'CA303', origin: 'Los Angeles', destination: 'Tokyo', time: '11:30 AM', status: 'Cancelled' },
        { flightNumber: 'DA404', origin: 'Paris', destination: 'Berlin', time: '01:15 PM', status: 'Ready' },
    ];

    const bookings = [];

    function displayFlights() {
        const flightListDiv = document.getElementById('flightList');
        flightListDiv.innerHTML = '';
        flights.forEach(flight => {
            flightListDiv.innerHTML += `
            <div class="flight-item">
                <div>
                    <strong>${flight.flightNumber}</strong><br>
                    ${flight.origin} to ${flight.destination}<br>
                    ${flight.time}<br>
                    <span class="flight-status status-${flight.status.toLowerCase().replace(" ", "-")}">${flight.status}</span>
                </div>
                <button onclick="checkAvailability('${flight.flightNumber}')">Check Availability</button>
            </div>
            `;
        });
    }

    function checkAvailability(flightNumber) {
        const flight = flights.find(f => f.flightNumber === flightNumber);
        if (flight) {
            showAlert(`Flight ${flight.flightNumber} is currently ${flight.status}.`, 'bookingAlert', flight.status === 'Ready'
```



```
<script>
    function checkAvailability(flightNumber) {
    }

    function bookFlight() {
        const flightNumber = document.getElementById('flightNumber').value;
        const passengerName = document.getElementById('passengerName').value;
        const passengerEmail = document.getElementById('passengerEmail').value;

        const flight = flights.find(f => f.flightNumber === flightNumber);
        if (flight && flight.status === 'Ready') {
            const booking = { flightNumber, passengerName, passengerEmail };
            bookings.push(booking);
            flight.status = 'Booked'; // Update flight status
            showAlert(`Flight ${flightNumber} booked successfully for ${passengerName}!`, 'bookingAlert', 'success');
            displayFlights(); // Refresh flight list
            displayBookings(); // Refresh bookings
            clearBookingInputs();
        } else {
            showAlert(`Flight ${flightNumber} is not available for booking.`, 'bookingAlert', 'error');
        }
    }

    function cancelBooking() {
        const flightNumber = document.getElementById('cancelFlightNumber').value;
        const bookingIndex = bookings.findIndex(b => b.flightNumber === flightNumber);

        if (bookingIndex !== -1) {
            const booking = bookings[bookingIndex];
            const flight = flights.find(f => f.flightNumber === booking.flightNumber);
            flight.status = 'Ready'; // Reset flight status
            bookings.splice(bookingIndex, 1); // Remove booking
            showAlert(`Booking for flight ${flightNumber} cancelled successfully.`, 'cancelAlert', 'success');
            displayFlights(); // Refresh flight list
```

Ln 180, Col 40   Spaces: 4   UTF-8   CRLF   {} HTML

PYTHON

Project Overview

The provided Python code is for a simple flight booking system using the Flask web framework. The system allows users to view available flights, book flights, and cancel bookings.

Backend Functionality

The code defines a Flask application with several routes:

1. / Route

The / route renders an index.html template, which is not provided in the code snippet.

2. /flights Route

The /flights route returns a JSON list of all available flights.

3. /book Route

The /book route books a flight by accepting a JSON payload with the flight number, passenger name, and passenger email. It updates the flight status to "Booked" and adds a new booking to the bookings list.

4. /cancel Route

The /cancel route cancels a booking by accepting a JSON payload with the flight number. It updates the flight status back to "Ready" and removes the corresponding booking from the bookings list.

Data Storage

The code uses two lists to store data:

1. flights List

The flights list stores information about each flight, including its number, origin, destination, time, and status.

2. bookings List

The bookings list stores information about each booking, including the flight number, passenger name, and passenger email.

Error Handling

The code returns JSON error messages with appropriate HTTP status codes (400) when:

- A flight is not available for booking.

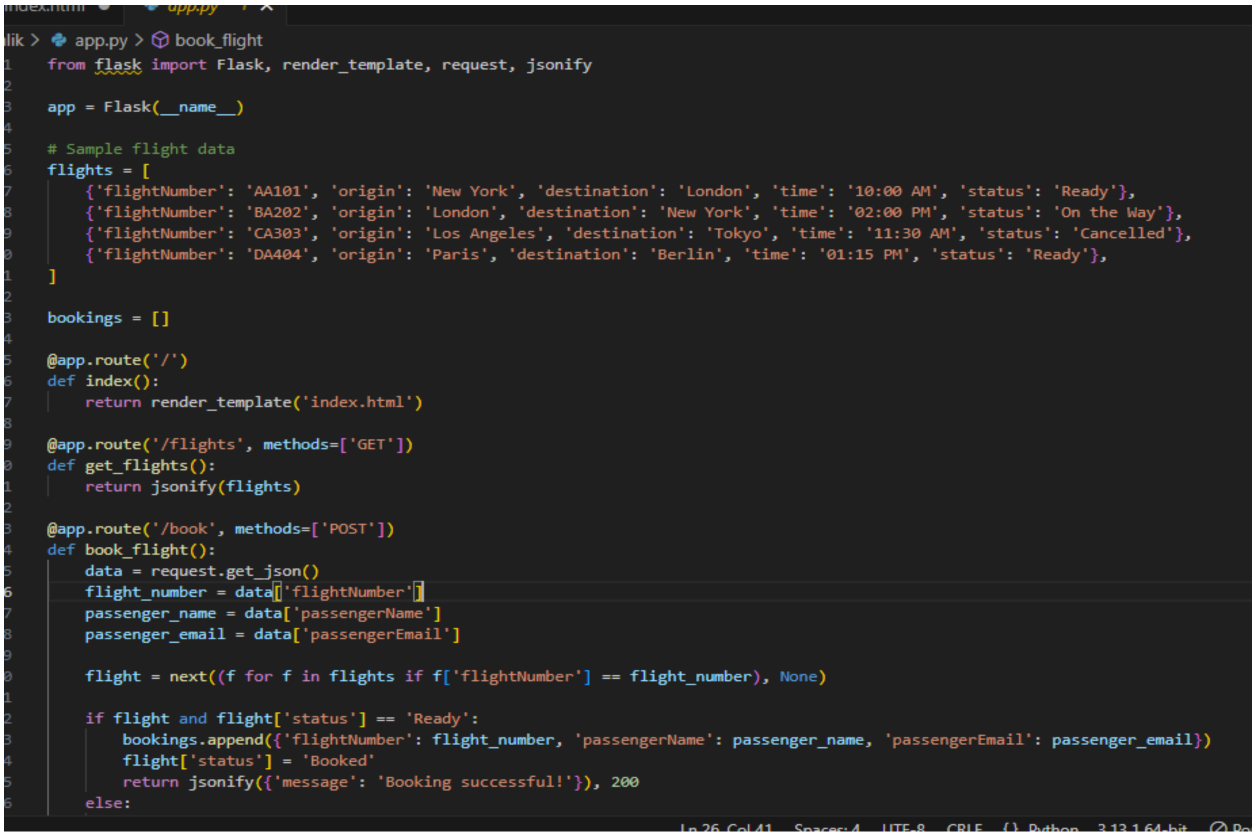
- A booking is not found for cancellation.

Security Considerations

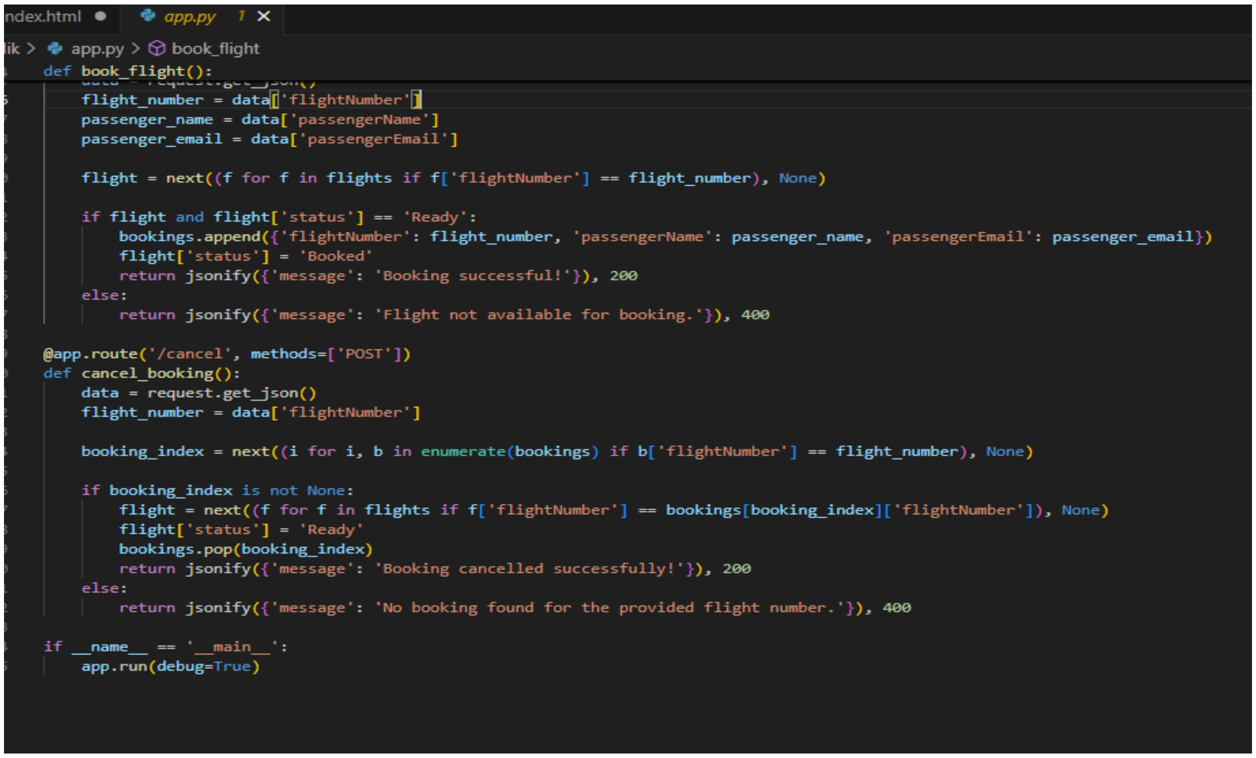
The code does not implement any authentication or authorization mechanisms, which is a significant security concern. In a real-world application, you should implement proper authentication and authorization to ensure only authorized users can access and modify data.

Conclusion

The provided Python code is a basic implementation of a flight booking system using Flask. While it demonstrates some essential backend functionality, it lacks proper security measures and data storage mechanisms. To build a robust and secure application, you should consider using a database to store data and implementing authentication and authorization mechanisms.



```python
from flask import Flask, render_template, request, jsonify

app = Flask(__name__)

# Sample flight data
flights = [
    {'flightNumber': 'AA101', 'origin': 'New York', 'destination': 'London', 'time': '10:00 AM', 'status': 'Ready'},
    {'flightNumber': 'BA202', 'origin': 'London', 'destination': 'New York', 'time': '02:00 PM', 'status': 'On the Way'},
    {'flightNumber': 'CA303', 'origin': 'Los Angeles', 'destination': 'Tokyo', 'time': '11:30 AM', 'status': 'Cancelled'},
    {'flightNumber': 'DA404', 'origin': 'Paris', 'destination': 'Berlin', 'time': '01:15 PM', 'status': 'Ready'},
]

bookings = []

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/flights', methods=['GET'])
def get_flights():
    return jsonify(flights)

@app.route('/book', methods=['POST'])
def book_flight():
    data = request.get_json()
    flight_number = data['flightNumber']
    passenger_name = data['passengerName']
    passenger_email = data['passengerEmail']

    flight = next((f for f in flights if f['flightNumber'] == flight_number), None)

    if flight and flight['status'] == 'Ready':
        bookings.append({'flightNumber': flight_number, 'passengerName': passenger_name, 'passengerEmail': passenger_email})
        flight['status'] = 'Booked'
        return jsonify({'message': 'Booking successful!'}), 200
    else:
```



```python
def book_flight():
    flight_number = data['flightNumber']
    passenger_name = data['passengerName']
    passenger_email = data['passengerEmail']

    flight = next((f for f in flights if f['flightNumber'] == flight_number), None)

    if flight and flight['status'] == 'Ready':
        bookings.append({'flightNumber': flight_number, 'passengerName': passenger_name, 'passengerEmail': passenger_email})
        flight['status'] = 'Booked'
        return jsonify({'message': 'Booking successful!'}), 200
    else:
        return jsonify({'message': 'Flight not available for booking.'}), 400

@app.route('/cancel', methods=['POST'])
def cancel_booking():
    data = request.get_json()
    flight_number = data['flightNumber']

    booking_index = next((i for i, b in enumerate(bookings) if b['flightNumber'] == flight_number), None)

    if booking_index is not None:
        flight = next((f for f in flights if f['flightNumber'] == bookings[booking_index]['flightNumber']), None)
        flight['status'] = 'Ready'
        bookings.pop(booking_index)
        return jsonify({'message': 'Booking cancelled successfully!'}), 200
    else:
        return jsonify({'message': 'No booking found for the provided flight number.'}), 400

if __name__ == '__main__':
    app.run(debug=True)
```