

ONLINE FLIGHT TICKET BOOKING WEBSITE MINI PROJECT

Submitted in partial fulfillment of the requirement for the award of

BACHELOR OF COMPUTER SCIENCE

Submitted By

D GOKUL-222201153

N VIGNESH-222201176

V VINOTH KUMAR-222201178

V BHARATHI KANNAN-222201182

Under the guidance of

Prof. S.NITHYA.,

UG DEPARTMENT OF COMPUTER SCIENCE



DHARMAMURTHI RAO BAHADUR CALAVALA CUNNAN CHETTY'S

HINDU COLLEGE [SHIFT-II]

“LINGUISTIC (TELUGU) MINORITY STATUS

CONFERRED BY THE GOVT.OF.TAMIL NADU”

Re-Accredited by NAAC –Affiliated to the University of Madras

Dharmamurthi Nagar, Pattabiram ,Chennai-600072

APRIL-2025

BONAFIDE CERTIFICATE

Certified that this report titled “**ONLINE FLIGHT TICKET BOOKING WEBSITE**” is a bonafide record of the project work done by **V VINOTH KUMAR (Reg No:-222201178)**. Under our supervision and guidance, towards partial fulfillment of the requirement for award of the Degree of B.SC Computer Science of **Dharmamurthi Rao Bahadur Calavala Cunnan Chetty's Hindu College.**

Signature of the HOD

Dr.V.Sasikala.,

Department of Computer Science

Signature of the Internal Guide

Prof.S.Nithya.,

Assistant Professor and

Department of Computer Science

Internal Examiner

External Examiner

Date:

DECLARATION

I am **V VINOTH KUMAR (Reg.No:-222201178)**. Hereby declare that the Mini project work title **“ONLINE FLIGHT TICKET BOOKING WEBSITE FOR AIRLINES”** submitted to the UG Department of Computer Science, D.R.B.C.C.HinduCollege ,Pattabiram , Chennai, in partial fulfillment of the requirement for the award of Bachelor of Computer Science is a record of original and independent work done by from December 2024 to April 2025 under the supervision and guidance of **Prof.S.Nithya**.

Place:

Date:

Signature of the candidate

TABLE OF CONTENTS

S.NO	TOPIC	PAGE NO.
1	ACKNOWLEDGEMENT	
2	ABSTRACT	
3	INTRODUCTION	
4	EXISTING SYSTEM	
5	SYSTEM ANALYSIS	
6	SOFTWARE SPECIFICATION	
7	DATA FLOW DIAGRAM	
8	UML DIAGRAM	
9	SOFTWARE AND HARDWARE CONFIGURATION	
10	SOURCE CODE	
11	SCREEN LAYOUTS AND DATABASE	
12	OVERVIEW	
13	CONCLUSION	
14	REFERENCE	

Preface

With the fast development of computer technology, the software projects are growing and complexity. Software experts have recently sought to develop a more systematic and formal approach in the design, development and implementation of their software. This new approach has become necessary because the traditional methods of system development often yielded software characterized by late diversity, costliness, unreliability, and non-maintainability and nonuse ability. In this new age of computing everything has been computerized, so how can we become solate and untouched from this environment. That's why keeping this thing in mind and an opportunity or probably a creativity to do such a task different and unique from others, we thought a way to develop this software. This project has been developed in aim to aid and computerize an flight booking. Whik keeping in mind the user will find an easy and friendlier user interface to perform his task. The software has been made so user friendly that any person can use it easily without having any computer experience.

1.ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and taking our Endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed guide, for providing us with the right guidance and advice at the crucial junctures and for showing me the right way. We also take this opportunity to express a deep sense of gratitude to our class coordinators, for their cordial support, valuable suggestions and guidance. We extend our sincere thanks to our respected Head of the department **Dr.V.Sasikala** Here, for allowing us to use the facilities available. We would like to thank the other faculty members also, at this occasion. Last but not the least, we would like to thank our friends and family for the support and encouragement they have given us during the course of our work

2. ABSTRACT

Flight Booking System For Indian Airlines is a system for booking flight. This project will show how a admin and user works in a web page. This project objective is to allow everyone to book a flight from they location to save time. This project database will maintain the details of the users and they feedback too. Users can print they ticket using this webpage itself, the printing option is inbuilt. Airline reservation System is a computerized system used to store and retrieve information and conduct transactions related to air travel. The project is aimed at exposing the relevance and importance of Airline Reservation Systems. It is projected towards enhancing the relationship between customers and airline agencies through the use of ARSs, and thereby making it convenient for the customers to book the flights as when they require such that they can utilize thissoftware to make reservations. After search the system display list of available flights and allows customer to choose a particular flight. Then the system checks for the availability of seats on the flight. If the seats are available then the system allows the passenger to book a seat. Otherwise it asks the user to choose another flight To book a flight the system asks the customer to enter his details such as name, address, city, state, and credit card number and contact number. Then it checks the validity of card and book the flight and update the airline database and user database. The system also allows the customer to cancel his/her reservation, if any problem occurs. The main purpose of this software is to reduce the manual errors involved in the airline reservation process and make it convenient for the customers to book the flights as when they require such that they can utilize this software to make reservations, modify reservations or cancel a particular reservation.

3. INTRODUCTION

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- Online Booking Website.
- A search the flight you want to book.
- If the flight is available then it will response.
- Users.
- An Admin login page where admin can add books, videos or page sources
- Open link for Learning Websites

PROJECT DESCRIPTION

Flight Booking Software is a complete flight booking quotation system which automates flight booking process to help book flight online for particular seats available from various flights and increase revenue. Airline Reservation System is complete airline quotation booking system helps airlines to distribute flight tickets online across multiple channels and optimize reservations for upcoming flights. Airline reservation system helps travel companies to integrate all flight-related data into online portal to provide customer with wide variety of flight options with competitive flight fares. It has a number of features that will allow users to book online flight tickets. This web application as well as the website's concept is all clear. It's the same as real-life scenarios and well-implemented on it. passengers can search for the flight details. For this, the passenger has to select departure and arrival location with dates, class, and number of passengers. As a result, the system filters out available flight schedules with detailed information.

Available Features:

- Client-Side Interaction
- Admin Panel
- Passenger Registration
- Booking Airline Tickets
- Simple Payment System
- Search for Flights
- View E-Ticket
- Cancel Tickets
- Print Tickets
- Check Flight Status
- View Total Amount
- List Today's Flights
- Manage Flight's Departure - Arrival
- Mark Flight Issues
- Flight Management Airlines Management

4.Existing System

The existing system is that the passenger must fill up the data manually and must submit it to the reservation counter. It may take a lot of time to process it and to book the flight. Therefore, there is a wastage of time. Since the data is entered manually, the probability of error or mistakes is high.

Need and Scope of Computer System.

Need:

- Computer enhance easy & simple means of storing information. The space
- occupied for storing of information is reduced
- Computer helps in fast retrieval of information we can search information in a
- short of time
- It is paperless system
- We can generate report on demand
- It will give faster modification
- Improve the client satisfaction & employee satisfaction.
- Enhance the stakeholder integration

Scope:

- The project has a wide scope, as it is not intended to a particular organization.
- This project is going to develop generic software which can be applied by any business organization. More over it provides facility to its users. Also, the software is young to provide a huge amount of summary.

5. SYSTEM ANALYSIS

Systems analysis is the study of sets of interacting entities, including computer systems analysis. Thus field is closely related to requirement analysis or operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker). It identifies a better course of action and make a better decision than he might otherwise have made. The development of a computer-based information system includes a systems analysis phase which produces or enhances data model which itself is a precursor to creating or enhancing a database. There are a number of different approaches to system analysis. When a computer-based information system is developed, systems analysis would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- They are few subtopic are:

1.GENERAL

2. DRAWBACKS OF EXISTING SYSTEM

3. PROPOSED SYSTEM

4. FEASIBILITY STUDY

1.GENERAL:

- Systems analysis researchers apply methodology to the analysis of systems involved to form an database for checking their results. System analysis is used in every field where there is a work of developing something. Analysis can also be defined as a series of components that perform organic function together
- The existing system is that the passenger must fill up the data manually and must submit it to the reservation counter. It may take a lot of time to process it and to book the flight.

- Therefore there is wastage of time. Since the data is entered manually, the probability of error or mistakes is high.

2. DRAWBACKS OF EXISTING SYSTEM:

Entering Record Entry of each record is done manually each time the record is done manual each time the record is maintained on paper and it maximizes the maintenance of additional files. Searching the record Due to absence of unique identification of a flight, the searching of record takes much time and increases the time wastage. Deleting the Record in the current system the concept of deleting record is tedious. Modification of Record Alf any modification is required it is done directly on the documents being preserved in correspondence to account information.

3. PROPOSED SYSTEM:

To avoid the limitation of current system it's necessary to design and develop a new system which have the following benefit and the existing system. Everything is automated which reduce the risk factor.

ADVANTAGES OF PROPOSED SYSTEM

- Customer services can not only be satisfied but also enhanced to the extent that one can obtain or cancel a reservation from any given time.
- Managing and maintaining data becomes easier and cost effective due to very high amount and reliability of storage space available in the proposed system.
- The proposed system due to computerized is much faster in reservation process, cancellation process and transactions.
- Transfer of information from various branches would become easier and faster.

4. FEASIBILITY STUDY

Feasibility study is a report directed management. It evaluates the impact of the proposed changes in the area(s) in question. The report is a formal document for management, brief enough and sufficiently, non technical to be understandable, yet detailed enough to provide the basis for system design. Technical feasibility centers around the existing system (hardware, software, etc) into what it can sort the proposed addition.

we think need not be feasible .It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

Technical Feasibility:

We can strongly says that it is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization here we are utilizing the resources which are available already.

Economical Feasibility:

Development of this application is highly economically feasible .The organization needed not spend much money for the development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. If we are doing so, we can attain the maximum usability of the corresponding resources .Even after the development, the organization will not be in condition to invest more in the organization .Therefore, the system is economically feasible.

6. SOFTWARE SPECIFICATION

FRONTEND

The frontend of the reservation system will be developed using a combination of HTML, CSS, JavaScript. HTML will provide the structure of the web pages, while CSS will provide the styling and layout. JavaScript will provide the interactivity and functionality. My SQLITE database to retrieve and store data. Django will be used as a front-end framework to enhance the overall design and responsiveness of the web pages, providing a consistent and modern look and feel across all devices, make reservations, and process payments securely. Creating a frontend reservation system for an airline ticket booking involves building a clean, intuitive, and responsive interface where users can search for flights, select dates, and make bookings

The frontend is the user interface of the Flight Reservation System, responsible for interacting with users and providing a seamless booking experience.

Frontend Features:

1. Search Flights: Search bar for flights by departure, arrival, and dates.
2. Flight Results: Display search results with flight details.
3. Booking Form: Form to input passenger details.
4. Payment Gateway: Secure payment processing.
5. Booking Confirmation: Display booking confirmation.

Frontend Technologies:

1. HTML: Structure and content.
2. CSS: Styling and layout.
3. JavaScript: Interactivity and dynamic updates.

Frontend Components:

1. Header: Navigation menu.
2. Footer: Contact information and social media links.
3. Search Bar: Input field for search queries.
4. Flight Results Table: Display search results.
5. Booking Form: Input fields for passenger details.

Benefits:

1. Improved User Experience: Intuitive interface for searching and booking flights.
2. Increased Conversion Rates: Streamlined booking process.
3. Enhanced Security: Secure payment processing.

Best Practices:

1. Responsive Design: Ensure compatibility with various devices.
2. Accessibility: Follow accessibility guidelines.
3. Testing: Thoroughly test frontend functionality.

HTML,

HTML (Hypertext Markup Language) is a standard markup language used for creating web pages and web applications. It provides a structured way to format content and display it in a web browser. HTML uses a set of tags and attributes to define the structure and presentation of web content. HTML is commonly used to create flight booking websites where users can search for flights, view schedules, and book tickets.

The HTML structure for a flight ticket booking system consists of the following key elements:

1. Header:

The header section includes:

- Navigation Menu: Links to search flights, book flights, and contact us.
- Logo: Flight ticket booking system logo.
- Search Bar: Input field for searching flights.

2. Search Flights:

The search flights section includes:

- Departure: Input field for departure city or airport.
- Arrival: Input field for arrival city or airport.
- Date: Input field for departure date.
- Passenger Count: Input field for number of passengers.
- Search Button: Button to submit search query.

3. Flight Results:

The flight results section includes:

- Flight Table: Table displaying search results with the following columns:
 - Flight Number: Unique flight identifier.
 - Departure: Departure city or airport.
 - Arrival: Arrival city or airport.
 - Date: Departure date.
 - Price: Flight price.
- Filter Options: Options to filter search results by price, departure time, or arrival time.

4. Booking Form:

The booking form section includes:

- Passenger Details: Input fields for passenger name, email, phone number, and address.
- Payment Information: Input fields for payment method, card number, and expiration date.
- Booking Confirmation: Button to confirm booking.

5. Footer:

The footer section includes:

- Contact Information: Address, phone number, and email.
- Social Media Links: Links to social media platforms.
- Terms and Conditions: Link to terms and conditions page.

Additional Elements:

- Error Messages: Displayed when user input is invalid or booking fails.
- Loading Animation: Displayed while booking is processing.
- Booking Confirmation Message: Displayed after successful booking.

CSS,

CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of web pages and web applications. It allows developers to separate the presentation of a web page from its content, making it easier to create and maintain consistent styles across multiple pages and sites.

The CSS for a flight ticket booking system includes styling for:

1. Layout: Structuring the search flights, flight results, and booking form sections.
2. Typography: Choosing fonts, font sizes, and line heights for headings, paragraphs, and form labels.
3. Colors: Selecting a palette of colors for the website's theme, including background colors, text colors, and accent colors.
4. Interactive Elements: Styling buttons, links, and form fields to provide visual feedback and enhance user experience.
5. Responsive Design: Ensuring the website adapts to different screen sizes and devices.
6. Spacing and Padding: Adding margins, padding, and borders to create a balanced and visually appealing layout.

CSS Properties and Values:

1. Layout: display, flex, grid, position.
2. Typography: font-family, font-size, line-height, color.
3. Colors: background-color, color, border-color.
4. Interactive Elements: hover, active, focus.
5. Responsive Design: @media, max-width, min-width.
6. Spacing and Padding: margin, padding, border.

JS,

JavaScript (often abbreviated as JS) is a high-level programming language used for creating interactive and dynamic web pages and web applications. It is a client-side scripting language, meaning it is executed in a user's web browser rather than on a server.

The JavaScript for a flight ticket booking system includes scripting for:

1. Dynamic Interactions: Handling user input, clicks, and hover effects to provide a responsive and interactive interface.
2. Form Validation: Verifying user input data to ensure accuracy and completeness.
3. Data Manipulation: Processing and updating data in real-time, such as flight availability and pricing.
4. API Integration: Connecting with external APIs to retrieve flight information, perform payment processing, and update booking status.
5. Error Handling: Catching and displaying errors to ensure a smooth user experience.
6. DOM Manipulation: Updating the Document Object Model (DOM) to reflect changes in the user interface.

JavaScript Features:

1. Event Listeners: Attaching event listeners to HTML elements to respond to user interactions.
2. AJAX Requests: Sending asynchronous requests to retrieve data from APIs.
3. JSON Data Handling: Parsing and manipulating JSON data from APIs.
4. Form Submission Handling: Handling form submissions and validating user input.
5. Animation and Effects: Adding visual effects to enhance user experience.

BACKEND

The backend for a flight ticket booking system includes a server-side architecture that:

1. Manages Flight Data: Stores and updates flight information, such as schedules, prices, and availability.
2. Handles Booking Requests: Processes booking requests from users, including validation and confirmation.
3. Processes Payments: Integrates with payment gateways to securely process transactions.
4. Integrates with External APIs: Connects with airlines, payment providers, and other third-party services.
5. Provides Real-time Updates: Ensures that users receive up-to-date information on flight availability and booking status.
6. Ensures Security: Implements measures to protect user data and prevent unauthorized access.
7. Supports Scalability: Designed to handle increased traffic and user activity.

Backend Technologies:

1. Programming Languages: python.
2. Frameworks: Django.
3. Databases: MySQLITE.
4. APIs: RESTful APIs, GraphQL.

Backend Components:

1. Flight Management System: Manages flight data and schedules.
2. Booking Engine: Handles booking requests and validation.
3. API Gateway: Manages API requests and responses.
4. Database: Stores flight data, booking information, and user data.

Django:

Django is a high-level Python web framework that enables rapid development of secure, maintainable, and scalable websites. It provides an architecture, templates, and APIs to build web applications quickly.

Django, a high-level Python web framework, can be used to build a scalable and efficient backend for a flight ticket booking system, providing features such as:

1. Object-Relational Mapping (ORM): Django's ORM system abstracts the underlying database, making it easy to interact with flight data.
2. Templates: Django's templating engine allows for easy rendering of flight search results, booking forms, and confirmation pages.
3. Authentication: Django provides built-in authentication and authorization features to secure user data and booking information.
4. Admin Interface: Django's admin interface allows for easy management of flights, bookings, and user accounts.
5. Scalability: Django's modular design and support for load balancing make it suitable for high-traffic flight booking systems.

Django Apps for Flight Ticket Booking System:

1. Flight App: Handles flight data, schedules, and availability.
2. Booking App: Manages booking requests, validation, and confirmation.
3. Payment App: Integrates with payment gateways for secure transactions.
4. User App: Handles user authentication, authorization, and profile management.

Django Models for Flight Ticket Booking System:

1. Flight Model: Represents a flight with attributes such as departure, arrival, date, and price.
2. Booking Model: Represents a booking with attributes such as flight ID, passenger name, and booking status.
3. User Model: Represents a user with attributes such as username, email, and password.

MYSQLITE:

MySQLITE is a popular open-source relational database management system that is commonly used for web applications. It provides a wide range of features and functions that make it easy to develop and manage databases.

MySQL is a relational database management system (RDBMS) used to manage data in a structured manner. It is an open-source database system that is widely used in web development.

Database are used in the software requirements are sqlite.3 version this version used to store the multiple file in the single file as the single folder .sqlite 3 version help us to store the database easily and the database are secured.

MySQLite, a lightweight and self-contained relational database management system, can be used in a flight ticket booking system project to:

1. Store Flight Data: Store information about flights, such as departure and arrival cities, dates, and prices.
2. Manage Bookings: Store booking information, such as passenger names, flight numbers, and booking status.
3. Store User Information: Store user data, such as usernames, passwords, and contact information.
4. Provide Data Retrieval: Retrieve data efficiently to support flight search, booking, and cancellation operations.
5. Ensure Data Integrity: Enforce data consistency and integrity through constraints and transactions.

MySQLite Features Suitable for Flight Ticket Booking System:

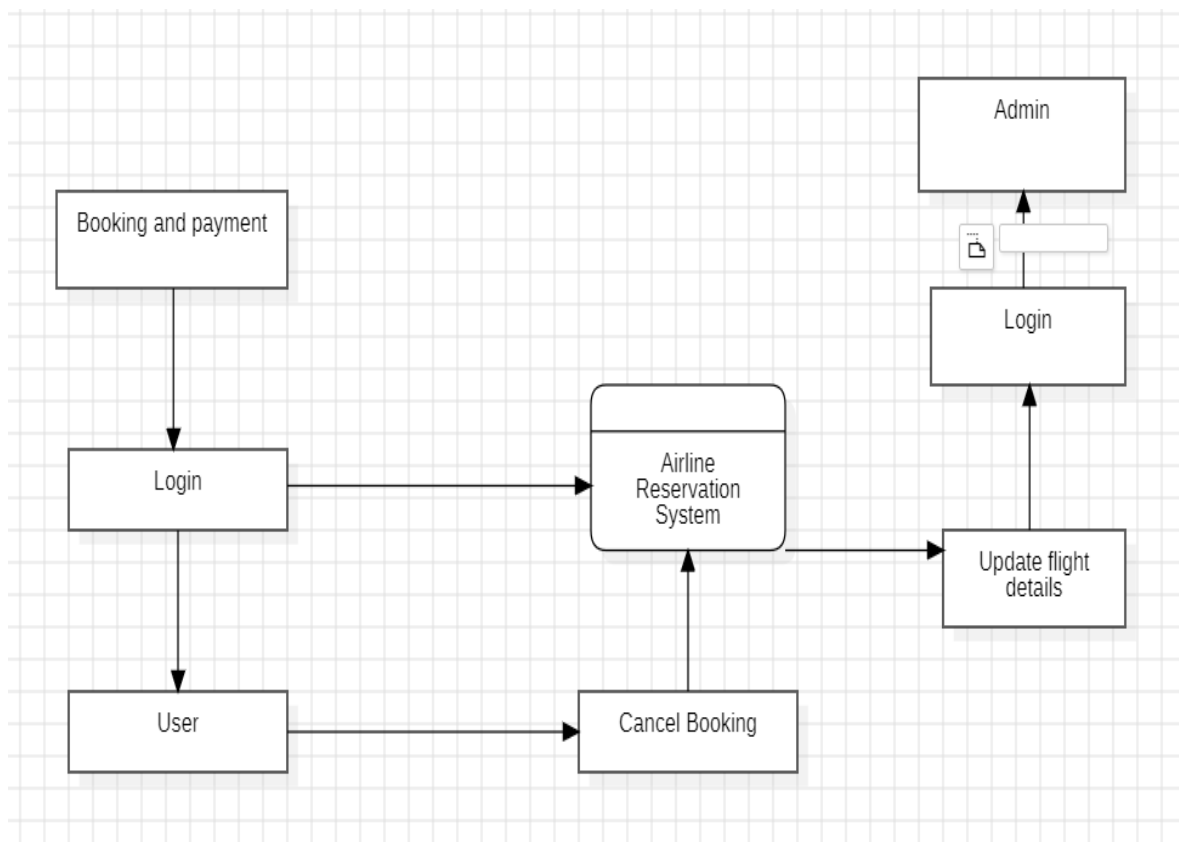
1. Serverless Architecture: SQLite does not require a separate server process, making it ideal for small-scale applications.
2. Self-Contained: SQLite databases are stored in a single file, making it easy to deploy and manage.
3. Lightweight: SQLite has a small footprint, requiring minimal system resources.
4. SQL Support: SQLite supports standard SQL syntax, making it easy to interact with the database.

7.Data Flow Diagram

Data flow diagrams (DFDs) visually map your process or system, so you can uncover opportunities to improve efficiency and performance.

The first level DFD (1st Level) of Airline Reservation System focus on more deep level details of internal processing. Level 1 DFD includes all the major sub processes that makes the entire ticket reservation system. It also identifies data store of flight schedule that contains all information about travel route, destination and fare. This information taken from several master data used is in the entire system where it needed such as search for seat on any particular route and date, ticket confirmation a final ticket printing etc.

The important process to be carried out are:



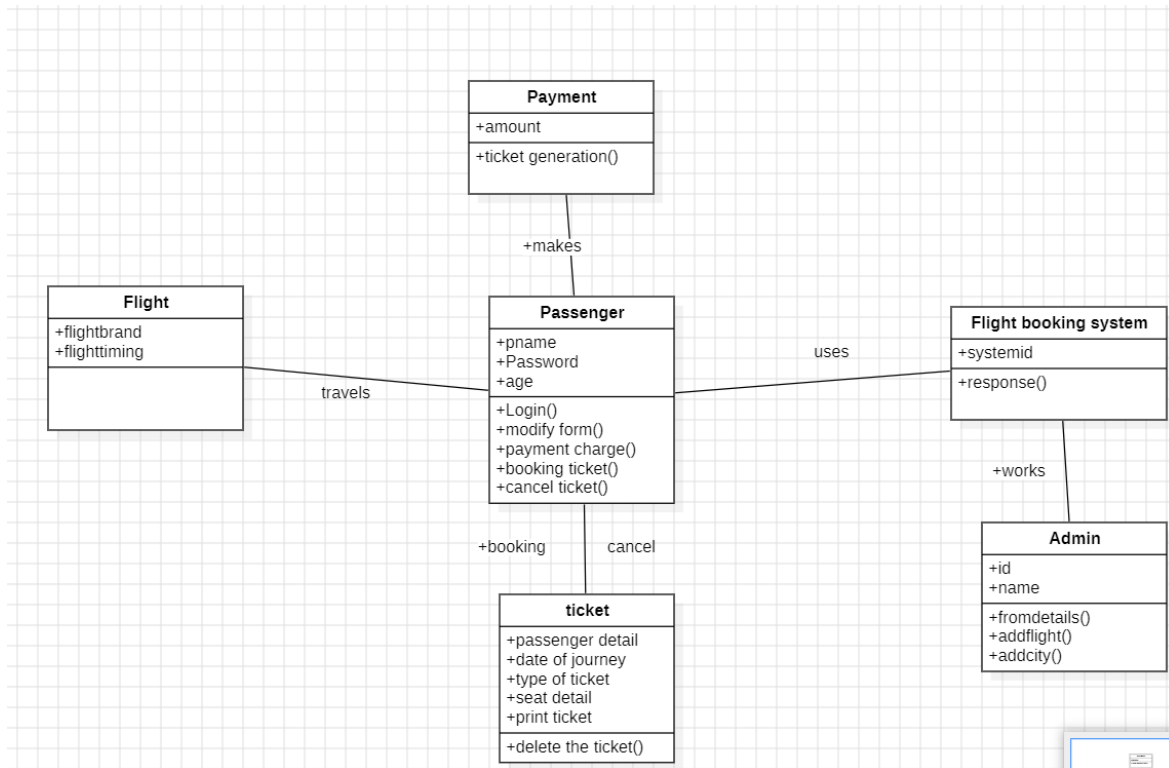
1. User login
2. Manage Flight Details
3. Customer Registration
4. Reservation Process
5. Payment & Ticket Print

The level 1 DFD for Flight Reservation System represents a broad overview but go into much depth than context level diagram. You can get more clear picture of the entire working in Ist level DFD.

8.UML DIAGRAMS

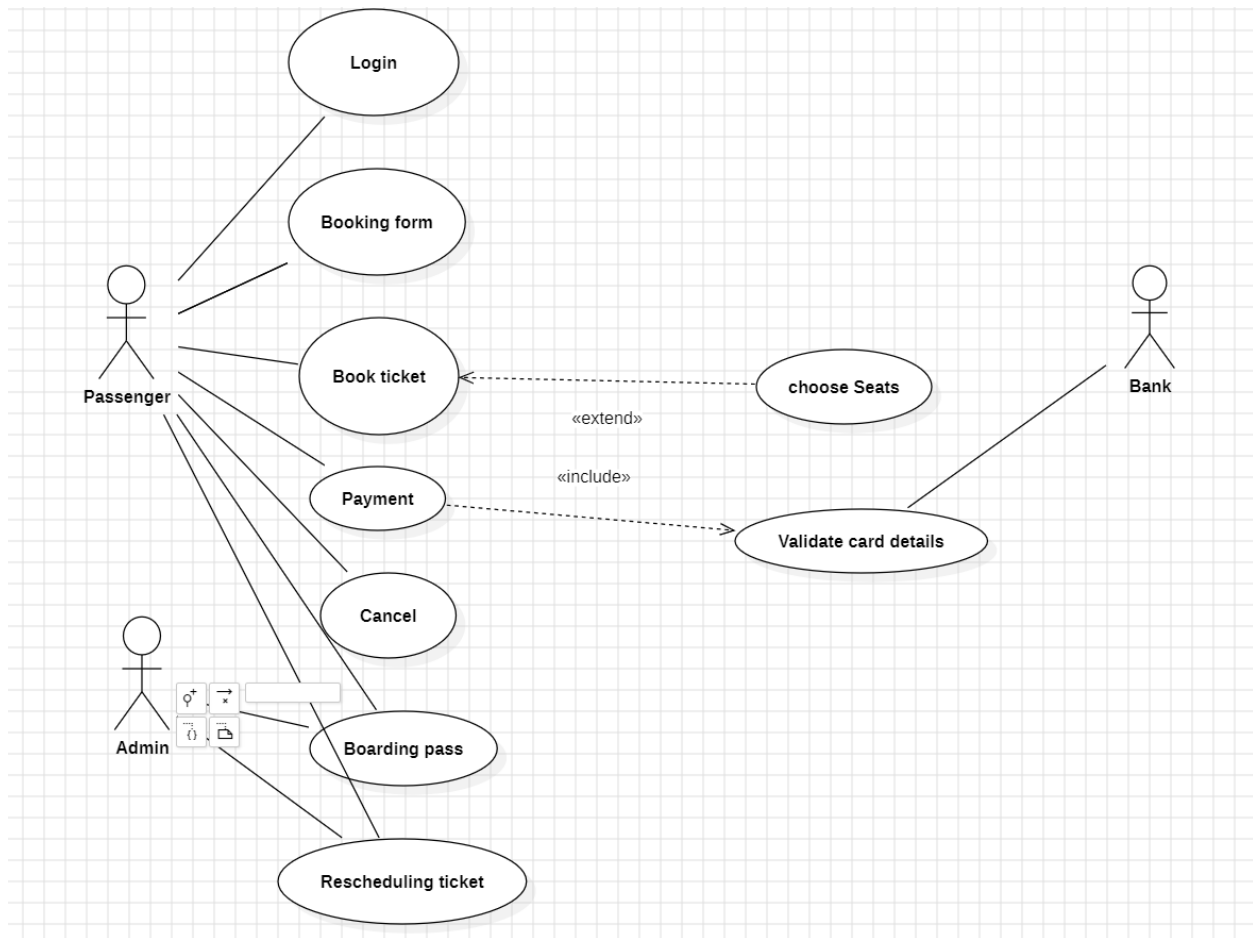
CLASS DIAGRAM

Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design.



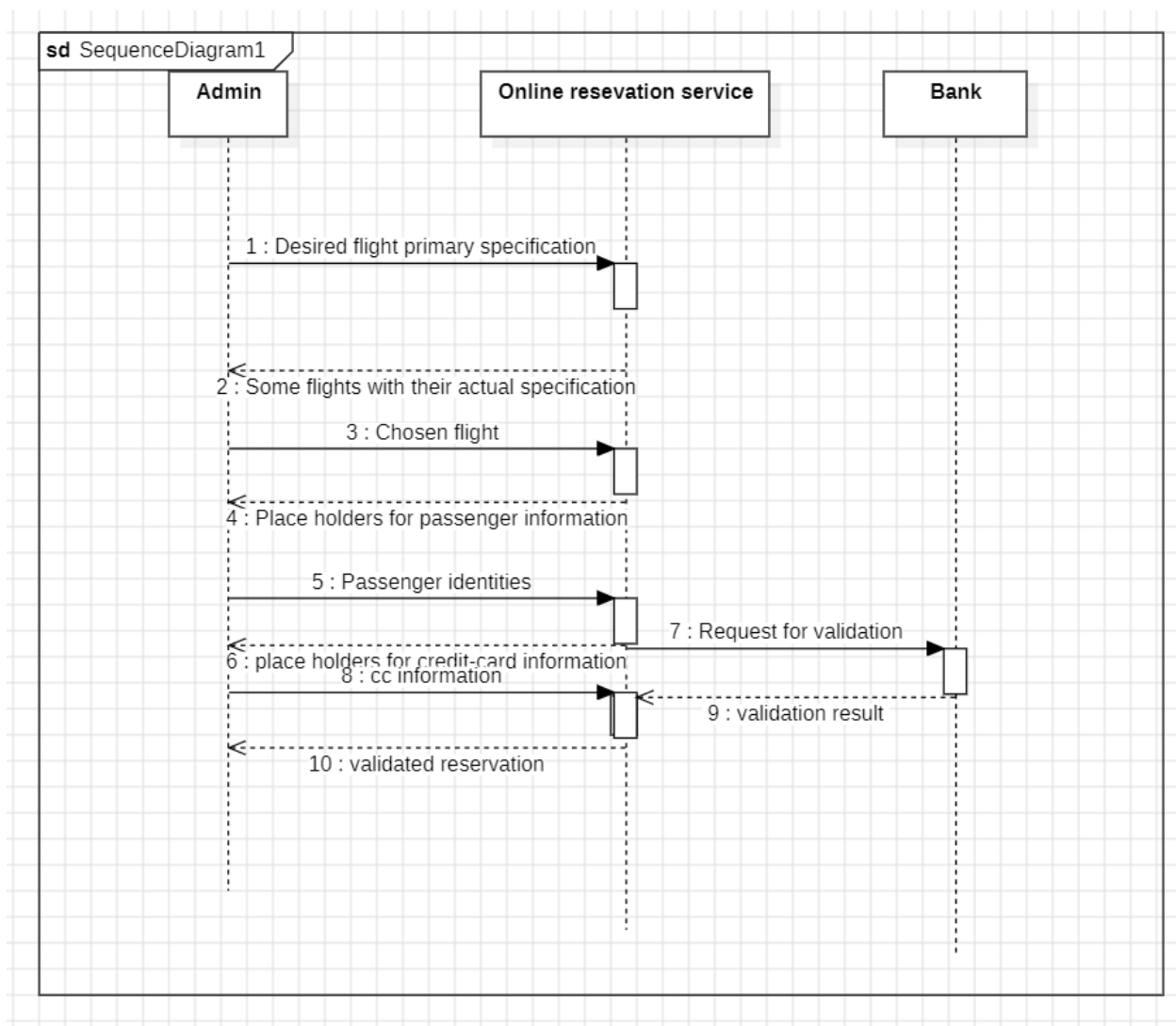
USE CASE DIAGRAM

A use case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system.



SEQUENCE DIAGRAM

The sequence diagram is a good diagram to use to document a system's requirements and to flush out a system's design. The reason the sequence diagram is so useful is because it shows the interaction logic between the objects in the system in the time order that the interactions take place.



9.SOFTWARE CONFIGURATION AND HARDWARE CONFIGURATION

SOFTWARE REQUIREMENTS

- Software or Resource Version Required
- A PHP engine Version 5. Included in XAMPP-Windows.
- A web server
- A database server
- Apache HTTP Server 2.2 is recommended. Included in XAMPP Windows.
- MySQL Server 5.0 is recommended. Included in XAMPP Windows.
- A PHP debugger (optional) and PHP version (Recommended): 5.6.7.4
- XDebug 2.0 or later.

HARDWARE REQUIREMENTS

- Processor: Intel Core i5 or equivalent
- RAM: 4GB or higher
- Display: 1080p or higher resolution

10.SOURCECODE

HOME PAGE.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  { % load static % }
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Airlines</title>
<link rel="stylesheet" href="{ % static 'css/style.css' % }">
<style>
body{
font-family:Arial,sans-serif;
width:100%;
height:100%;
background-image:url("{ % static 'images/flight.jpg' % }");
background-repeat:no-repeat;
}
</style>
</head>
<body>
<nav>
<ul style="display:flex;justify-content:space-between">
<li><a href="{ % url 'aboutUs' % }">About Us</a></li>
{ % if user.is_authenticated % }
<li><a href="{ % url 'logout' % }" onclick="return confirm('Are you sure you want to log
out?');">Logout</a></li>
{ % endif % }
```



```

</ul>
</nav>
<div class="logo"><center><imgsrc="{% static 'images/logo.png' %}" alt="Airlines
Logo"></center></div>
<main>
<section>
<center>
{% if not user.is_authenticated %}
<a href="{% url 'login' %}">Log-in or Sign-up</a>
{% else %}
<h1>Fly Airlines</h1>
<p>Your Journey, Our Priority</p>
<button onclick="redirectToBooking()">Book Now</button>
{% endif %}
</center>
</section>
<section class="about-us">
<h2>About Airlines</h2>
<p>Airlines is a premier airline committed to providing exceptional air travel experiences. With a
focus on passenger comfort, safety, and punctuality, we strive to exceed expectations. Our
dedicated team is passionate about delivering seamless journeys, whether you're a frequent flyer
or a first-time traveler. Explore our diverse network of destinations and experience the difference
of flying Airlines </p>
</section>
</main>
<footer>
<p>
<span>&copy; 2024 Airlines. All Rights Reserved.</span>
<span>
<a href="tel:+913705954448">Phone: +91 370 5954448</a> |
<a href="mailto:info@airlines.com">Email: info@airlines.com</a>

```

```
</span>
</p>
</footer
<script>
functionredirectToBooking() {
  {% if user.is_authenticated %}
  window.location.href = "{% url 'user_home' %}";
  {% else %}
  alert("Please login to book a flight!")
  window.location.href = "{% url 'home' %}"
  {% endif %}
}
</script>
</body>
</html>
```

Base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
{ % load static % }
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Airlines</title>
<link rel="stylesheet" href="{ % static 'css/style.css' % }">
<style>
body{
font-family:Arial,sans-serif;
width:100%;
height:100%;
background-image:url("{ % static 'images/flight.jpg' % }");
background-repeat:no-repeat;
}
</style>
</head>
<body>
<nav>
<ul style="display:flex;justify-content:space-between">
<li><a href="{ % url 'aboutUs' % }">About Us</a></li>
{ % if user.is_authenticated % }
<li><a href="{ % url 'logout' % }" onclick="return confirm('Are you sure you want to log
out?');">Logout</a></li>
{ % endif % }
</ul>
</nav>
<div class="logo">
<center><imgsrc="{ % static 'images/logo.png' % }" alt="Airlines Logo"></center>
```

```
</div>
<main>
<section>
<center>
{ % if not user.is_authenticated % }
<a href="{ % url "login" % }">Log-in or Sign-up</a>
{ % else % }
<h1>Fly Airlines</h1>
<p>Your Journey, Our Priority</p>
<button onclick="redirectToBooking()">Book Now</button>
{ % endif % }
</center></section>
<section class="about-us">
<h2>About Airlines</h2>
<p>
Airlines is a premier airline committed to providing exceptional air travel experiences.
With a focus on passenger comfort, safety, and punctuality, we strive to exceed expectations.
Our dedicated team is passionate about delivering seamless journeys, whether you're a frequent
flyer or a first-time traveler. Explore our diverse network of destinations and experience
the difference of flying Airlines.
</p>
</section>
</main>
<footer>
<p>
<span>&copy; 2024 Airlines. All Rights Reserved.</span>
<span>
<a href="tel:+913705954448">Phone: +91 370 5954448</a> |
<a href="mailto:info@airlines.com">Email: info@airlines.com</a>
</span>
</p>
```

```
</footer>
<script>
functionredirectToBooking() {
  {% if user.is_authenticated %}
  window.location.href = "{% url 'user_home' %}";
  {% else %}
  alert("Please login to book a flight!")
  window.location.href = "{% url 'home' %}"
  {% endif %}
}
</script>
</body>
</html>
```

Booking.html

```
{% extends 'public/base.html' %}

{% load static %}

{% block title %}Booking Confirmation - Airlines{% endblock %}

{% block content %}

<style>

body {

font-family: 'Arial', sans-serif;

background: url("{% static 'images/flight.jpg' %}") no-repeat center center fixed;

background-size: cover;

color: white;

text-align: center;

margin: 0;

padding: 0;

}

.booking-container {

max-width: 500px;

margin: 50px auto;

padding: 25px;

background: rgba(255, 255, 255, 0.9);

border-radius: 12px;

box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);

text-align: center;

color: #333;

}

.booking-container h2 {

color: #1E3A8A;

margin-bottom: 15px;

}

.booking-container p {

font-size: 18px;
```

```
color: #444;
margin: 5px 0;
}
.ref-number {
font-size: 22px;
font-weight: bold;
color: #DC2626;
margin-top: 10px;
}
.btn {
display: inline-block;
padding: 12px 24px;
font-size: 18px;
color: white;
background: #1E40AF;
border: none;
border-radius: 6px;
cursor: pointer;
transition: 0.3s;
text-decoration: none;
margin-top: 15px;
}
.btn:hover {
background: #2563EB;
}
.error-container {
margin-top: 50px;
}
.error-container h3 {
font-size: 24px;
color: red;
```

```

}
</style>
<div class="booking-container">
{% if message %}
<h2>Booking Confirmed! □</h2>
<p>Your flight has been successfully booked.</p>
<p class="ref-number">Reference No: {{ booking_reference }}</p>
<a class="btn" href="{% url 'home' %}">Go to Home</a>
{% elif flight %}
<h2>Confirm Your Booking</h2>
<p><strong>Full Name:</strong> {{ flight.fullname }}</p>
<p><strong>Flight:</strong> {{ flight.airline }} - {{ flight.flight_number }}</p>
<p><strong>Departure:</strong> {{ flight.departure }} → {{ flight.destination }}</p>
<p><strong>Departure Date:</strong> {{ flight.departure_date }}</p>
<p><strong>Class:</strong> {{ flight.class_type }}</p>
{% if amount %}
<p><strong>Amount:</strong> {{ amount }}</p>
{% endif %}
<form method="post">
{% csrf_token %}
<button type="submit" class="btn">Confirm Booking</button>
</form>
{% else %}
<div class="error-container">
<h3>404 Not Found!</h3>
<p>Oops! The flight you are looking for does not exist.</p>
<a class="btn" href="{% url 'home' %}">Go to Home</a>
</div>
{% endif %}</div>{% endblock %}

```


Payment.html

```
{% extends "public/base.html" %}

{% load static %}

{% block title %}Payments - Airlines{% endblock %}

{% block content %}

<style>

.payment-container {
max-width: 500px;
margin: 50px auto;
padding: 20px;
background: rgba(255, 255, 255, 0.9);
border-radius: 12px;
box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
text-align: center;
color: #333;
}

.payment-container h2 {
color: #1E3A8A;
margin-bottom: 15px;
}

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

.form-group label {
font-weight: bold;
}

.form-group input {
width: 100%;
padding: 10px;
border: 1px solid #ddd;
```

```

border-radius: 6px;
}
.btn {
display: inline-block;
padding: 12px 24px;
font-size: 18px;
color: white;
background: #1E40AF;
border: none;
border-radius: 6px;
cursor: pointer;
transition: 0.3s;
text-decoration: none;
margin-top: 15px;
}
.btn:hover {
background: #2563EB;
}
</style>
<div class="payment-container">
<h2>Payment Information</h2>
<form id="paymentForm" method="post">
{% csrf_token %}
<div class="form-group">
<label for="cardholder-name">Cardholder Name</label>
<input type="text" id="cardholder-name" name="cardholder-name" placeholder="Enter
cardholder name" required>
</div>
<div class="form-group">
<label for="card-number">Card Number</label>

```

```

<input type="number" id="card-number" name="card-number" placeholder="XXXX
XXXXXXXXXXXX" maxlength="16" pattern="\d{16}" required>
</div>
<div class="form-group">
<label for="card-expiry">Expiry Date</label>
<input type="text" id="card-expiry" name="card-expiry" placeholder="MM/YY"
maxlength="5" pattern="(0[1-9]|1[0-2])\d{2}" required>
</div>
<button type="submit" class="btn">Pay Now</button>
</form>
</div>
<script>
document.getElementById("card-number").addEventListener("input", function (event) {
this.value = this.value.replace(/\D/g, "").slice(0, 16);
});
document.getElementById("card-expiry").addEventListener("input", function (event) {
let input = this.value.replace(/\D/g, "");
if (input.length >= 2) {
let month = input.slice(0, 2);
let year = input.slice(2, 4);
if (parseInt(month) > 12) {
month = "12";
} else if (parseInt(month) === 0) {
month = "01";
}
this.value = month + (year ? '/' + year : "");
} else {
this.value = input;
}
});</script>
{% endblock %}

```

Boarding.html

```
{% extends 'public/base.html' %}
{% load static %}
{% block title %}Booking Confirmation - Airlines{% endblock %}
{% block content %}
<style>
body {
font-family: 'Arial', sans-serif;
background: url("{% static 'images/flight.jpg' %}") no-repeat center center fixed;
background-size: cover;
color: white;
text-align: center;
margin: 0;
padding: 0;
}
.booking-container {
max-width: 500px;
margin: 50px auto;
padding: 25px;
background: rgba(255, 255, 255, 0.9);
border-radius: 12px;
box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
text-align: center;
color: #333;
}
.booking-container h2 {
color: #1E3A8A;
margin-bottom: 15px;
}
.booking-container p {
```

```
font-size: 18px;
color: #444;
margin: 5px 0;
}
.ref-number {
font-size: 22px;
font-weight: bold;
color: #DC2626;
margin-top: 10px;
}
.btn {
display: inline-block;
padding: 12px 24px;
font-size: 18px;
color: white;
background: #1E40AF;
border: none;
border-radius: 6px;
cursor: pointer;
transition: 0.3s;
text-decoration: none;
margin-top: 15px;
}
.btn:hover {
background: #2563EB;
}
.error-container {
margin-top: 50px;
}
.error-container h3 {
font-size: 24px;
```

```

color: red;
}
</style>
<div class="booking-container">
{% if message %}
<h2>Booking Confirmed! ☐</h2>
<p>Your flight has been successfully booked.</p>
<p class="ref-number">Reference No: {{ booking_reference }}</p>
<a class="btn" href="{% url 'home' %}">Go to Home</a>
{% elif flight %}
<h2>Confirm Your Booking</h2>
<p><strong>Full Name:</strong> {{ flight.fullname }}</p>
<p><strong>Flight:</strong> {{ flight.airline }} - {{ flight.flight_number }}</p>
<p><strong>Departure:</strong> {{ flight.departure }} → {{ flight.destination }}</p>
<p><strong>Departure Date:</strong> {{ flight.departure_date }}</p>
<p><strong>Class:</strong> {{ flight.class_type }}</p>
{% if amount %}
<p><strong>Amount:</strong> {{ amount }}</p>
{% endif %}
<form method="post">
{% csrf_token %}
<button type="submit" class="btn">Confirm Booking</button>
</form>
{% else %}
<div class="error-container">
<h3>404 Not Found!</h3>
<p>Oops! The flight you are looking for does not exist.</p>
<a class="btn" href="{% url 'home' %}">Go to Home</a>
</div>
{% endif %}</div>{% endblock %}

```

Rescheduling or cancel

```
{% extends 'public/base.html' %}

{% load static %}

{% block title %}Reschedule or Cancel Your Flight - Airlines{% endblock %}

{% block content %}

<style>

body {

font-family: 'Arial', sans-serif;

background: url("{% static 'images/flight.jpg' %}") no-repeat center center fixed;

background-size: cover;

color: white;

text-align: center;

margin: 0;

padding: 0;

}

.header {

font-size: 28px;

margin: 20px 0;

font-weight: bold;

color: black;

}

.form-container {

display: flex;

justify-content: center;

gap: 30px;

flex-wrap: wrap;

}

.form-box {

background: rgba(255, 255, 255, 0.9);

padding: 25px;

border-radius: 12px;
```

```
box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
width: 350px;
text-align: center;
}
.form-box h2 {
color: #1E3A8A;
margin-bottom: 15px;
}
.form-group {
margin-bottom: 15px;
text-align: left;
}
.form-group label {
font-weight: bold;
display: block;
margin-bottom: 5px;
}
.form-group input {
width: 100%;
padding: 10px;
border: 1px solid #ddd;
border-radius: 6px;
font-size: 16px;
}
.btn {
width: 100%;
padding: 12px;
font-size: 18px;
color: white;
border: none;
border-radius: 6px;
```



```
cursor: pointer;
transition: 0.3s;
text-decoration: none;
margin-top: 15px;
}
.btn-reschedule {
background: #1E40AF;
}
.btn-reschedule:hover {
background: #2563EB;
}
.btn-cancel {
background: #B91C1C;
}
.btn-cancel:hover {
background: #DC2626;
}
label{
color:black;
}
{% comment %} Error message {% endcomment %}
.messages {
text-align: center;
margin: 15px auto;
width: 80%;
}
.success {
color: green;
font-weight: bold;
}
.error {
```

```

color: red;
font-weight: bold;
}
</style>
<h1 class="header">Manage Your Booking</h1>
{% if messages %}
<div class="messages">
{% for message in messages %}
<p class="{{ message.tags }}">{{ message }}</p>
{% endfor %}
</div>
{% endif %}
<div class="form-container">
<div class="form-box">
<h2>Reschedule Your Flight</h2>
<form method="POST" id="rescheduleForm" action="{% url 'reschedule_flight' %}">
{% csrf_token %}
<div class="form-group">
<label for="booking-reference">Booking Reference:</label>
<input type="text" id="booking-reference" name="booking_reference" required>
</div>
<div class="form-group">
<label for="new-date">New Departure Date:</label>
<input type="date" id="new-date" name="new_date" min="{{ min_departure_date }}"
required>
</div>
<button type="submit" class="btn btn-reschedule">Reschedule Flight</button>
</form>
</div>
<div class="form-box">
<h2>Cancel Your Flight</h2>

```

```
<form method="POST" id="cancelForm" action="{% url 'cancel_flight' %}">
{% csrf_token %}
<div class="form-group">
<label for="booking-reference-cancel">Booking Reference:</label>
<input type="text" id="booking-reference-cancel" name="booking_reference" required>
</div>
<button type="submit" class="btn btn-cancel">Cancel Flight</button>
</form>
</div>
</div>
{% endblock %}
```

About.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>About us - Airlines</title>
<link rel="stylesheet" href="{% static 'css/style.css' %}">
<style>
body{
font-family:Arial,sans-serif;
width:100%;
height:100%;
background-image:url("{% static 'images/flight.jpg' %}");
background-repeat:no-repeat;
}
</style>
</head>
<body>
<header>
<nav>
<ul>
<li><a href="{% url 'home' %}">Home</a></li>
</ul>
</nav>
<div class="logo">
<center><imgsrc="{% static 'images/logo.png' %}" alt="Airlines Logo"></center>
</div>
</header>
<section class="about-us">
```

<h2>About Us</h2>

<p style="color:white;">Airlines is a premier airline committed to providing exceptional air travel experiences. With a focus on passenger comfort, safety, and punctuality, we strive to exceed expectations. Our dedicated team is passionate about delivering seamless journeys, whether you're a frequent flyer or a first-time traveler. Explore our diverse network of destinations and experience the difference of flying Airlines. We offer a range of services, including online booking, check-in, and baggage tracking. Our commitment to customer satisfaction is unwavering, and we strive to make every flight a memorable one.</p>

</section>

<footer>

<p style="display: flex; justify-content: space-between;">

© 2024 Airlines. All Rights Reserved.

Phone: +91 8072025504

Email: riddlebot@gmail.com

</p>

</footer>

</body>

</html>

404.html

```
{% extends "public/base.html" %}

{% load static %}

{% block title %}404 Not Found{% endblock %}

{% block content %}

<style>

    .error-container {
text-align: center;
margin-top: 100px;
    }

    .error-container h1 {
font-size: 100px;
color: #B91C1C;
    }

    error-container p {
font-size: 20px;
color: #444;
    }

    .btn-home {
display: inline-block;
padding: 12px 24px;
font-size: 18px;
color: white;
background: #1E40AF;
border: none;
border-radius: 6px;
cursor: pointer;
transition: 0.3s;
text-decoration: none;
margin-top: 15px;
    }
```

```
.btn-home:hover {  
background: #2563EB;  
}  
</style>  
<div class="error-container">  
<h1>404</h1>  
<p>Oops! The page you are looking for does not exist.</p>  
<a href="{ % url 'home' % }" class="btn-home">Go to Home</a></div>{ % endblock % }
```

.CSS

```
{% extends "public/base.html" %}
{% load static %}
{% block title %}404 Not Found{% endblock %}
{% block content %}
<style>
.error-container {
text-align: center;
margin-top: 100px;
}
.error-container h1 {
font-size: 100px;
color: #B91C1C;
}
.error-container p {
font-size: 20px;
color: #444;
}
.btn-home {
display: inline-block;
padding: 12px 24px;
font-size: 18px;
color: white;
background: #1E40AF;
border: none;
border-radius: 6px;
cursor: pointer;
transition: 0.3s;
text-decoration: none;
margin-top: 15px;
}
```



```
.btn-home:hover {  
background: #2563EB;  
}  
</style>  
<div class="error-container">  
<h1>404</h1>  
<p>Oops! The page you are looking for does not exist.</p>  
<a href="{ % url 'home' % }" class="btn-home">Go to Home</a>  
</div>  
{ % endblock % }
```

0001_initial.py :

```
import django.db.models.deletion
from django.conf import settings

from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [migrations.swappable_dependency(settings.AUTH_USER_MODEL), ]

    operations = [migrations.CreateModel(name='Flight', fields=[('id',
models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')), ('fullname', models.CharField(max_length=100, verbose_name='Full
Name')), ('age', models.PositiveIntegerField(verbose_name='Age')), ('gender',
models.CharField(choices=[('Male', 'Male'), ('Female', 'Female'), ('Others', 'Others')],
max_length=20, verbose_name='Gender')), ('airline', models.CharField(max_length=100,
verbose_name='Airline')), ('flight_number', models.CharField(max_length=20, unique=True,
verbose_name='Flight Number')), ('departure', models.CharField(max_length=100,
verbose_name='Departure City')), ('destination',
models.CharField(max_length=100)), ('boarding_time', models.TimeField(blank=True, null=True,
verbose_name='Boarding Time')), ('departure_date', models.DateField(verbose_name='Departure
Date')),

('return_date', models.DateField(blank=True, null=True, verbose_name='Return Date')),

('class_type', models.CharField(max_length=20, null=True, verbose_name='Class')),

('user', models.ForeignKey(blank=True, null=True,
on_delete=django.db.models.deletion.CASCADE, related_name='user_flights',
to=settings.AUTH_USER_MODEL)),

],

),
```

```
migrations.CreateModel(

    name='Booking',

    fields=[

        ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),

        ('booking_date', models.DateTimeField(auto_now_add=True)),

        ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='user_bookings', to=settings.AUTH_USER_MODEL)),

        ('flight', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='flight_bookings', to='flights.flight')),

    ],

),

]
```

Admin.py:

```
from django.contrib import admin
```

Apps.py:

```
from django.apps import AppConfig
```

```
class FlightsConfig(AppConfig):
```

```
    default_auto_field = 'django.db.models.BigAutoField'
```

```
    name = 'apps.flights'
```

```
    label = 'flights'
```

Models.py:

```
from django.db import models

from django.contrib.auth import get_user_model

User = get_user_model()

class Flight(models.Model):

    user = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="user_flights", null=True, blank=True)

    # PayanigalInfos

    fullname = models.CharField(max_length=100, null=False, verbose_name="Full Name")

    age = models.PositiveIntegerField(null=False, verbose_name="Age")

    GENDER_CHOICES = [

        ('Male','Male'),

        ('Female','Female'),

        ('Others','Others'),

    ]

    gender = models.CharField(max_length=20, null=False, choices=GENDER_CHOICES,
verbose_name="Gender")

    # VimanaInfos

    airline = models.CharField(max_length=100, verbose_name="Airline")

    flight_number = models.CharField(max_length=20, unique=True, verbose_name="Flight
Number")

    departure = models.CharField(max_length=100, verbose_name="Departure City")
```

```

destination = models.CharField(max_length=100)

# Date infos

boarding_time = models.TimeField(verbose_name="Boarding Time",null=True,blank=True)

departure_date = models.DateField(null=False, verbose_name="Departure Date")

return_date = models.DateField(null=True, blank=True, verbose_name="Return Date")

class_type = models.CharField(max_length=20,null=True,verbose_name="Class")

def __str__(self):

    return f"{self.airline} - {self.flight_number} ({self.departure} -> {self.destination})"

class Booking(models.Model):

    user = models.ForeignKey(User,
on_delete=models.CASCADE,related_name="user_bookings")

    flight = models.ForeignKey(Flight,
on_delete=models.CASCADE,related_name="flight_bookings")

    booking_date = models.DateTimeField(auto_now_add=True)

def __str__(self):

    returnf"Booking: {self.user.full_name} -> {self.flight.flight_number}"

```

Tests.py:

```
from django.test import TestCase
```

Urls.py:

```
from django.urls import path
```

```
from .views import *
```

```
urlpatterns = [
```

```
    path('confirm_booking/', confirm_booking, name='confirm_booking'),
```

```
    path('boardings/', boardingPass_view, name='boardings'),
```

```
    path('payments/', payment_view, name='payments'),
```

```
    path('rechedules/', resched_cancel_view, name='rechedules'),
```

```
    path('aboutUs/', aboutUs_view, name='aboutUs'),
```

```
    path('book_flight/', book_flight, name='book_flight'),
```

```
    path('cancel_flight/', cancel_flight, name='cancel_flight'),
```

```
    path('reschedule_flight/', reschedule_flight, name='reschedule_flight')
```

Views.py:

```
from django.shortcuts import render, redirect, get_object_or_404

from django.contrib.auth.decorators import login_required

from .models import Flight, Booking

from django.contrib import messages

import random

def resched_cancel_view(request):

    return render(request, 'public/resched_cancel.html')

@login_required(login_url='login')

def boardingPass_view(request):

    # bookings = Booking.objects.filter(user=request.user).order_by("-id")

    bookings = Booking.objects.filter(user=request.user).order_by("-id")

    if not bookings.exists():

        return render(request, 'public/boarding.html', {"message": "No Bookings Found!"})

    return render(request, 'public/boarding.html', {

        "bookings": bookings,

        "seat_number": random.randint(1, 50),

    })

@login_required(login_url='login')

def payment_view(request):

    flight_data = request.session.get("flight_data")
```



```

if not flight_data:

    return redirect("book_flight")

if request.method == "POST":

    request.session["payment_success"] = True

    return redirect("confirm_booking")

return render(request, "public/payment.html", {"flight": flight_data})

@login_required(login_url='login')

def confirm_booking(request):

    if request.method == "POST":

        if not request.user.is_authenticated:

            return redirect('login')

        flight_data = request.session.get('flight_data')

        if not flight_data:

            return redirect('user_home')

from datetime import time

boarding_time = time(random.randint(0,23),random.randint(0,59))

flight = Flight.objects.create(

    fullname = flight_data['fullname'],

    age = flight_data['age'],

    gender = flight_data['gender'],

    airline = flight_data['airline'],

```

```

        flight_number = flight_data['flight_number'],

        departure = flight_data['departure'],

        destination=flight_data["destination"],

        departure_date=flight_data["departure_date"],

        return_date=flight_data["return_date"],

        class_type=flight_data["class_type"],

        boarding_time=boarding_time

    )

booking = Booking.objects.create(user=request.user, flight=flight)

delrequest.session["flight_data"]

amount = random.randint(3000, 5000)

return render(request, "public/booking.html", {

    "booking":booking,

    "booking_reference":booking.id,

    "message": "Booking Confirmed!",

    "amount": amount })

flight_data = request.session.get("flight_data", { })

return render(request,'public/booking.html',{ "flight":flight_data})

@login_required(login_url='login')

defbook_flight(request):

    ifrequest.method == "POST":

```

```

if not request.user.is_authenticated:

    return redirect('login')

flight_data = {

    "fullname": request.POST.get("fullname"),

    "age": request.POST.get("age"),

    "gender": request.POST.get("gender"),

    "airline": request.POST.get("airline"),

    "departure": request.POST.get("departure"),

    "destination": request.POST.get("destination"),

    "departure_date": request.POST.get("departure-date"),

    "return_date": request.POST.get("return-date"),

    "class_type": request.POST.get("class"),

}

airline_code = flight_data["airline"][:2].upper()

flight_data["flight_number"] = f"{airline_code}{random.randint(100, 999)}"

request.session["flight_data"] = flight_data

return redirect("payments")

return render(request, 'public/booking.html')

@login_required(login_url='login')

defcancel_flight(request):

    ifrequest.method == "POST":

```

```

reference_no = request.POST.get("booking_reference")

booking = Booking.objects.filter(id=reference_no,user=request.user).first()

if booking:

    booking.delete()

    messages.success(request,"Your flight booking has been successfully canceled.")

else:

    messages.error(request,"Booking not found! Check Your Reference Number!")

    return redirect("reschedules")

@login_required(login_url='login')

def reschedule_flight(request):

    if request.method == "POST":

        reference_number = request.POST.get('booking_reference')

        new_departure_date = request.POST.get('new_date')

        booking = Booking.objects.filter(id=reference_number,user=request.user).first()

        if booking:

            booking.flight.departure_date = new_departure_date

            booking.flight.save()

            messages.success(request,"Flight has been rescheduled successfully!")

        else:

            messages.error(request,"Booking not found! Check Your Reference Number!")

        return redirect('reschedules')

```

Non functional page

defaboutUs_view(request):

 return render(request,'public/about.html')

login_signup.html:

```
{% extends "public/base.html" %}

{% load static %}

{% block title %}Login / Signup - Airlines{% endblock %}

{% block content %}

<div class="logo">

<center><imgsrc="{% static 'images/logo.png' %}" alt="Airlines Logo"></center>

</div>

</header>

<div class="container">

<center>

<div class="login-signup">

    {% comment %} Login form {% endcomment %}

<div class="form-container">

<form method="post" action="{% url 'login' %}" id="login-form">

    {% csrf_token %}

<h2>Login</h2>

<h3>{{ message }}</h3>

<label for="email">Email:</label>

<input type="email" id="email" name="email" placeholder="Enter your email" required>

<label for="password">Password:</label>
```

```
<input type="password" id="password" name="password" placeholder="Enter your password"
required>
```

```
<br><br>
```

```
<center><button type="submit">Login</button></center>
```

```
<p>Don't have an account? <a href="#" onclick="toggleForm('signup')">Sign Up</a></p>
```

```
</form>
```

```
{ % comment % } Signup Form { % endcomment % }
```

```
<form method="POST" action="{ % url 'signup' % }" id="signup-form" style="display: none;">
```

```
{ % csrf_token % }
```

```
<h2>Sign Up</h2>
```

```
<label for="fullname">Full Name:</label>
```

```
<input type="text" id="fullname" name="fullname" placeholder="Enter your full name"
required>
```

```
<label for="email">Email:</label>
```

```
<input type="email" id="email" name="email" placeholder="Enter your email" required>
```

```
<label for="password">Password:</label>
```

```
<input type="password" id="password" name="password1" placeholder="Enter your password"
required>
```

```
<label for="password">Re-Enter Password:</label>
```

```
<input type="password" id="password2" name="password2" placeholder="Retype your
password" required>
```

```
<br><br>
```

```

<center><button type="submit">Sign Up</button></center>

<p>Already have an account? <a href="#" onclick="toggleForm('login')">Log In</a></p>

</form>

</div>

</div>

</center>

</div>

<script>

functiontoggleForm(formId) {

constloginForm = document.getElementById('login-form');

constsignupForm = document.getElementById('signup-form');

if (formId === 'login') {

loginForm.style.display = 'block';

signupForm.style.display = 'none';

    } else {

loginForm.style.display = 'none';

signupForm.style.display = 'block';

    }

};

document.addEventListener("DOMContentLoaded", function () {

letloginEmail = document.querySelector("#login-form input[name='email']");

```



```
letsignupEmail = document.querySelector("#signup-form input[name='email']");

letloginForm = document.getElementById("login-form");

letsignupForm = document.getElementById("signup-form");

letemailPattern = /^[a-zA-Z0-9._%+-]+@gmail\.com$/;

functionvalidateEmail(input) {

  if (!emailPattern.test(input.value)) {

    input.style.border = "2px solid red";

    return false;

      } else {

        input.style.border = "2px solid green";

        return true;

      }

  }

  loginEmail.addEventListener("input", function () {

    validateEmail(loginEmail);

  });

  signupEmail.addEventListener("input", function () {

    validateEmail(signupEmail);

  });

  loginForm.addEventListener("submit", function (event) {

    if (!validateEmail(loginEmail)) {
```

```

event.preventDefault();

alert("Please enter a valid Gmail address (e.g., example@gmail.com).");

    }

});

signupForm.addEventListener("submit", function (event) {

if (!validateEmail(signupEmail)) {

event.preventDefault();

alert("Please enter a valid Gmail address (e.g., example@gmail.com).");

    }

});

});

</script>

{% endblock %}

```

User_home.html:

```
{% extends "public/base.html" %}

{% load static %}

{% block title %}Welcome {{ user.fullname }} - Airlines{% endblock %}

{% block content %}

<style>

body {

font-family: 'Arial', sans-serif;

background: url("{% static 'images/flight.jpg' %}") no-repeat centercenter fixed;

background-size: cover;

color: white;

text-align: center;

margin: 0;

padding: 0;

}

.booking-container {

max-width: 500px;

margin: 50px auto;

padding: 25px;

background: rgba(255, 255, 255, 0.9);

border-radius: 12px;
```

```
box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
```

```
text-align: center;
```

```
color: #333;
```

```
}
```

```
.booking-container h2 {
```

```
color: #1E3A8A;
```

```
margin-bottom: 20px;
```

```
}
```

```
.form-group {
```

```
margin-bottom: 15px;
```

```
text-align: left;
```

```
}
```

```
.form-group label {
```

```
font-weight: bold;
```

```
display: block;
```

```
margin-bottom: 5px;
```

```
}
```

```
.form-group input,
```

```
  .form-group select {
```

```
width: 100%;
```

```
padding: 10px;
```

```
border: 1px solid #ddd;

border-radius: 6px;

font-size: 16px;

}

submit-button {

display: inline-block;

padding: 12px 24px;

font-size: 18px;

color: white;

background: #1E40AF;

border: none;

border-radius: 6px;

cursor: pointer;

transition: 0.3s;

text-decoration: none;

margin-top: 15px;

width: 100%;

}

.submit-button:hover {

background: #2563EB;

}
```

```
</style>

<div class="booking-container">

<h2>Book Your Flight</h2>

<form method="post" action="{ % url 'book_flight' % }" id="booking-form">

    { % csrf_token % }

<div class="form-group">

<label for="name">Full Name:</label>

<input type="text" id="name" name="fullname" placeholder="Enter your full name" required>

</div>

<div class="form-group">

<label for="age">Age:</label>

<input type="number" id="age" name="age" placeholder="Enter your age" min="1" max="120"
required>

</div><div class="form-group">

<label for="gender">Gender:</label>

<select id="gender" name="gender" required>

<option value="" disabled selected>Select your gender</option>

<option value="Male">Male</option>

<option value="Female">Female</option>

<option value="Other">Other</option>

</select>
```

</div>

<div class="form-group">

<label for="airlines">Airlines:</label>

<select id="airlines" name="airline" required>

<option value="" disabled selected>Select Airline</option>

{% for airline in airlines %}

<option value="{{ airline }}">{{ airline }}</option>

{% endfor %}

</select>

</div>

<div class="form-group">

<label for="departure">Departure City:</label>

<select id="departure" name="departure" required>

<option value="" disabled selected>Select departure city</option>

{% for city in cities %}

<option value="{{ city }}">{{ city }}</option>

{% endfor %}

</select>

</div>

<div class="form-group">

<label for="destination">Destination City:</label>

```

<select id="destination" name="destination" required>

<option value="" disabled selected>Select destination city</option>

    { % for city in cities % }

<option value="{{ city }}">{{ city }}</option>

    { % endfor % }

</select>

</div>

<div class="form-group">

<label for="departure-date">Departure Date:</label>

<input type="date" id="departure-date" name="departure-date" min="{{ min_departure_date }}"
required>

</div>

<div class="form-group">

<label for="return-date">Return Date:</label>

<input type="date" id="return-date" name="return-date" min="{{ min_departure_date }}"
required>

</div>

<div class="form-group">

<label for="class">Class:</label>

<select id="class" name="class" required>

<option value="" disabled selected>Select class</option>

<option value="economy">Economy</option>

```



```

<option value="business">Business</option>

<option value="first">First Class</option>

</select>

</div>

<button type="submit" class="submit-button">Book Now</button>

</form>

</div>

<script>

document.addEventListener("DOMContentLoaded", function () {

constdepartureDateInput = document.getElementById("departure-date");

constreturnDateInput = document.getElementById("return-date");

departureDateInput.addEventListener("change", function () {

returnDateInput.min = departureDateInput.value;

    });

returnDateInput.addEventListener("change", function () {

if (returnDateInput.value<departureDateInput.value) {

alert("Return date cannot be earlier than the departure date!");

returnDateInput.value = "";

    }

    });

});

```

```
</script>
```

```
{% endblock %}
```

Wsgi.py:

```
import os
```

```
from django.core.asgi import get_asgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'flightApp.settings')
```

```
application = get_asgi_application()
```

.gitignore:

```
__pycache__/
```

```
.venv
```

```
db.sqlite3
```

```
__pycache__/
```

```
*.pyc
```

```
*.pyo
```

```
*.pyd
```

Manage.py:

```
#!/usr/bin/env python
```

```
"""Django's command-line utility for administrative tasks."""
```

```
import os
```

```
import sys
```

```
def main():
```

```

"""Run administrative tasks."""

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'flightApp.settings')

try:

    from django.core.management import execute_from_command_line

except ImportError as exc:

    raise ImportError(

        "Couldn't import Django. Are you sure it's installed and "

        "available on your PYTHONPATH environment variable? Did you "

        "forget to activate a virtual environment?"

    ) from exc

execute_from_command_line(sys.argv)

if __name__ == '__main__':

    main()

```

Requirement.txt:

```

asgiref==3.8.1

Django==5.1.5

sqlparse==0.5.3

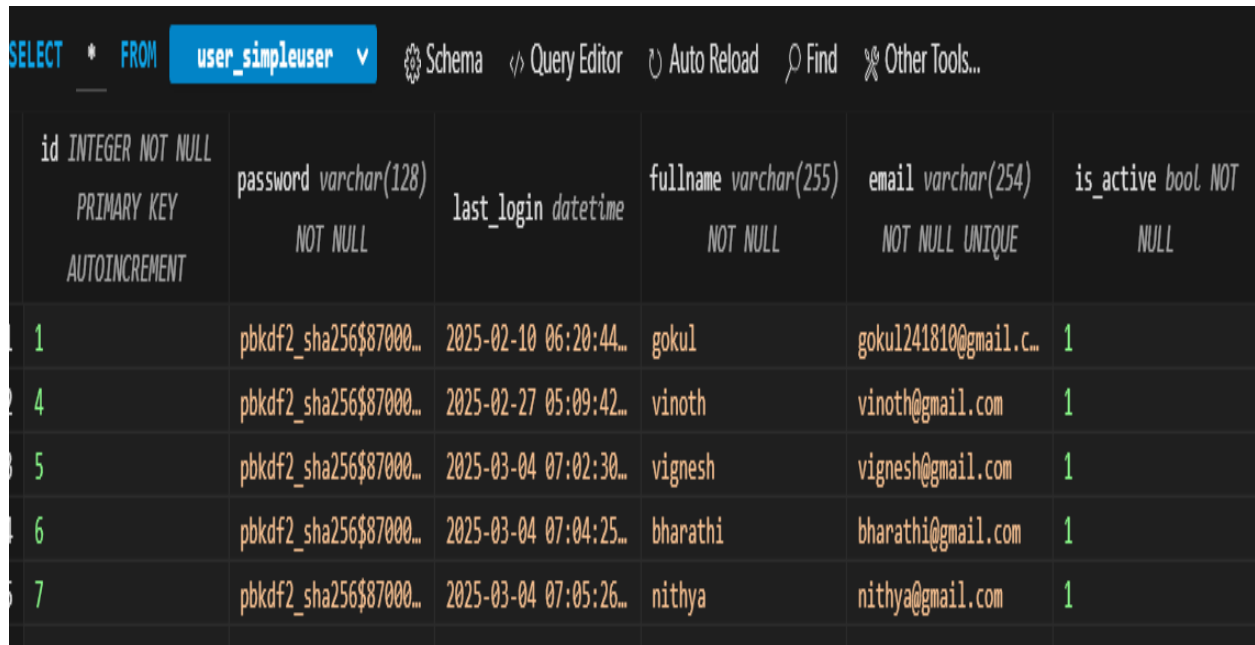
```

11. SCREEN LAYOUTS AND DATABASE

MODULES

- Admin
- User
- Cites
- Flight
- Passenger Profile
- Payment
- Ticket
- Users

1.ADMIN:



The screenshot displays a database management interface. At the top, there is a toolbar with options: 'SELECT * FROM user_simpleuser', 'Schema', 'Query Editor', 'Auto Reload', 'Find', and 'Other Tools...'. Below the toolbar, a table structure is shown with columns: 'id' (INTEGER NOT NULL, PRIMARY KEY, AUTOINCREMENT), 'password' (varchar(128) NOT NULL), 'last_login' (datetime), 'fullname' (varchar(255) NOT NULL), 'email' (varchar(254) NOT NULL UNIQUE), and 'is_active' (bool NOT NULL). The table contains five rows of data, each with a green ID number in the first column.

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	password varchar(128) NOT NULL	last_login datetime	fullname varchar(255) NOT NULL	email varchar(254) NOT NULL UNIQUE	is_active bool NOT NULL
1	1	pbkdf2_sha256\$87000...	2025-02-10 06:20:44...	gokul	gokul241810@gmail.c...	1
2	4	pbkdf2_sha256\$87000...	2025-02-27 05:09:42...	vinoth	vinoth@gmail.com	1
3	5	pbkdf2_sha256\$87000...	2025-03-04 07:02:30...	vignesh	vignesh@gmail.com	1
4	6	pbkdf2_sha256\$87000...	2025-03-04 07:04:25...	bharathi	bharathi@gmail.com	1
5	7	pbkdf2_sha256\$87000...	2025-03-04 07:05:26...	nithya	nithya@gmail.com	1

- This webpage contains the information about the flight and can modify the flights details.
- The admin file is a critical component of the Flight Reservation System project, responsible for managing administrative tasks and ensuring the system's smooth operation.

Admin File Responsibilities:

1. User Management: Creating, updating, and deleting user accounts.
2. Flight Management: Adding, updating, and deleting flight schedules.
3. Booking Management: Managing flight bookings and cancellations.
4. Payment Processing: Handling payment transactions.
5. System Configuration: Configuring system settings and parameters.

Admin File Features:

1. Login Authentication: Secure login for admin access.
2. Dashboard: Overview of system activity and statistics.
3. User Roles: Assigning roles and permissions.
4. Flight Scheduling: Creating and managing flight schedules.
5. Booking Reports: Generating booking reports.

Benefits:

1. Centralized Management: Easy administration.
2. Improved Security: Controlled access.
3. Efficient Operations: Streamlined processes.

DATABASE:

- Database are used to store the data
- The overview of the database design for online flight ticket booking website:
 1. Flights Details
 2. Passengers Details
 3. Booking & Payment Details

Database Requirements:

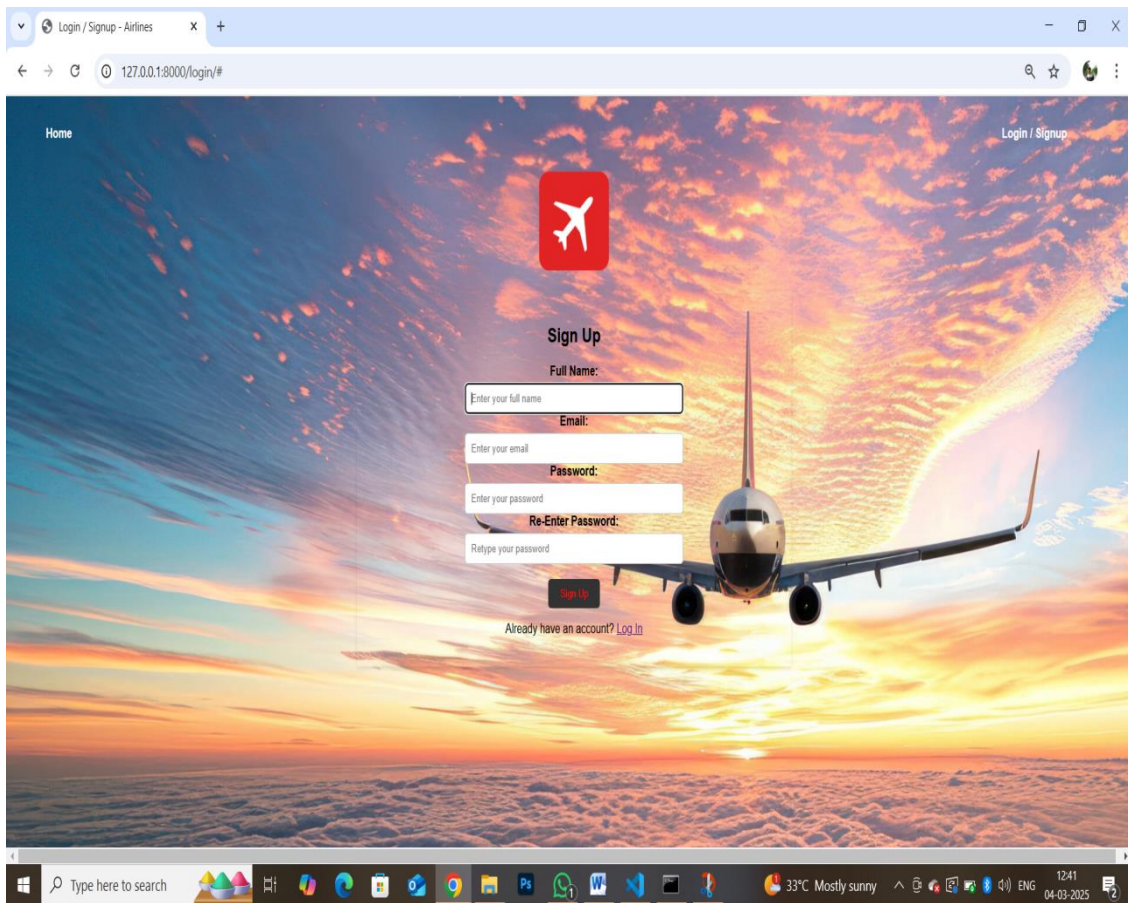
1. Flight Information: Store flight schedules, departure and arrival times, prices, and availability.
2. Passenger Data: Store passenger information, such as names, contact details, and travel documents.
3. Booking Records: Store booking information, including flight numbers, passenger names, and booking status.
4. Payment Details: Store payment information, including credit card numbers, expiration dates, and transaction amounts.
5. Security: Ensure data encryption, access controls, and auditing to protect sensitive information.

Database Design Considerations:

1. Data Normalization: Normalize data to eliminate redundancy and improve data integrity.
2. Data Relationships: Establish relationships between tables to enable efficient data retrieval.
3. Indexing: Use indexing to improve query performance.
4. Scalability: Design the database to scale with increasing traffic and user growth.

Database Management System Options:

1. Relational Databases: MySQL, PostgreSQL, Microsoft SQL Server.
2. NoSQL Databases: MongoDB, Cassandra, Redis.
3. Cloud Databases: Amazon Aurora, Google Cloud SQL, Microsoft Azure SQL Database.



db.sqlite3 X

db.sqlite3

SELECT * FROM flights_booking Schema Query Editor Auto Reload Find Other Tools...

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	booking_date datetime NOT NULL	user_id bigint NOT NULL REFERENCES user_simpleuser(id) idx ²	flight_id bigint NOT NULL REFERENCES flights_flight(id) idx ¹	+
1	1	2025-02-10 06:23:40...	1 password:'pbkdf2...	1 fullname:'gokul'...	
2	4	2025-03-04 07:19:08...	4 password:'pbkdf2...	4 fullname:'vinoth...	
3	5	2025-03-04 07:21:10...	5 password:'pbkdf2...	5 fullname:'vignes...	
4	6	2025-03-04 07:22:41...	6 password:'pbkdf2...	6 fullname:'bharat...	
5	7	2025-03-04 07:24:15...	7 password:'pbkdf2...	7	

The admin has a separate login to log in and admin has a built in password which is stored in the database.

- This is the dashboard for the admin page.
- Here we can see the total number of passengers and total amount available airlines.

Booking Ticket:

- To Book a ticket , The user need to login the account and fill up their details
- After fillup their details , the User can book the ticket by online

The booking ticket process in a flight ticket reservation system involves a series of steps that enable passengers to reserve and pay for their flights, including selecting flights, providing passenger information, making payment, and receiving confirmation.

Booking Ticket Process:

1. Flight Search: Passengers search for available flights based on departure and arrival cities, dates, and times.
2. Flight Selection: Passengers select their preferred flight and class of travel.
3. Passenger Information: Passengers provide required information, such as name, contact details, and travel documents.
4. Payment: Passengers make payment for their booking using a credit card or other accepted payment methods.
5. Booking Confirmation: The system generates a booking confirmation, including a booking reference number and flight details.

6. Ticket Issuance: The system issues a ticket, either electronically or physically, depending on the airline's policies.

Booking Ticket Features:

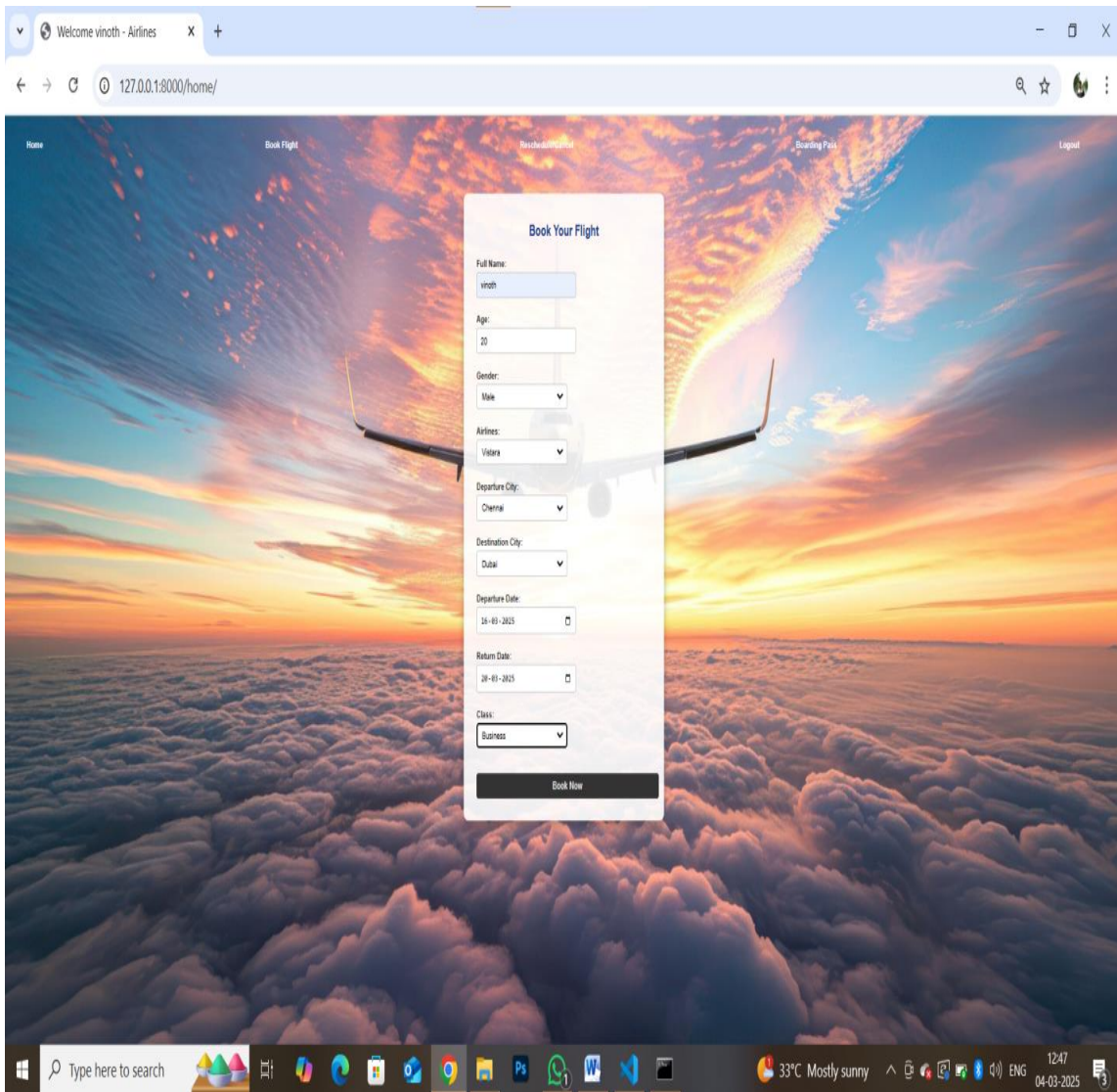
1. Availability Check: The system checks for flight availability and updates in real-time.

2. Fare Calculation: The system calculates fares based on the selected flight and class of travel.

3. Payment Processing: The system processes payments securely and efficiently.

4. Booking Status Update: The system updates the booking status in real-time, reflecting changes such as cancellations or modifications.

5. Ticket Printing: The system prints tickets, either electronically or physically, depending on the airline's policies.



SELECT * FROM flights_flight Schema Query Editor Auto Reload Find Other Tools... SQLite 3.49.1

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	fullname varchar(100) NOT NULL	age integer unsigned NOT NULL	gender varchar(20) NOT NULL	airline varchar(100) NOT NULL	flight_number varchar(20) NOT NULL UNIQUE	departure varchar(100) NOT NULL	destination varchar(100) NOT NULL	boarding_t
1	1	gokul	20	Male	Air India	AI517	Bangalore	Mumbai	03:12:00
2	3	vinoth	20	Male	Air India	AI964	Chennai	Delhi	13:41:00
3	4	vinoth	20	Male	Vistara	VI561	Chennai	Dubai	08:31:00
4	5	vignesh	20	Male	GoAir	G0704	Bangalore	London	07:07:00
5	6	bharathi	20	Male	Indigo	IN975	Hyderabad	Delhi	10:46:00
6	7	nithya	28	Female	Emirates	EM359	Delhi	Ahmedabad	12:28:00

SELECT * FROM flights_flight Schema Query Editor Auto Reload Find Other Tools... SQLite 3.49.1

	flight_number varchar(20) NOT NULL UNIQUE	departure varchar(100) NOT NULL	destination varchar(100) NOT NULL	boarding_time time	departure_date date NOT NULL	return_date date	class_type varchar(20)	user_id bigint REFERENCES user_simpleuser(id) idx	
1	AI517	Bangalore	Mumbai	03:12:00	2025-02-12	2025-02-22	first	NULL	
2	AI964	Chennai	Delhi	13:41:00	2025-02-27	2025-03-06	business	NULL	
3	VI561	Chennai	Dubai	08:31:00	2025-03-16	2025-03-20	business	NULL	
4	G0704	Bangalore	London	07:07:00	2025-03-06	2025-03-22	economy	NULL	
5	IN975	Hyderabad	Delhi	10:46:00	2025-03-16	2025-03-29	first	NULL	
6	EM359	Delhi	Ahmedabad	12:28:00	2025-03-15	2025-03-29	business	NULL	

- In this admin place, admin can able to see the customer's details like booking details and date, time as a database.

Login page:

- The user already created the account can able to login directly using login
- After the user login ,they can see the details about their flight details

The login page in a flight ticket booking system is a secure entry point that authenticates users, allowing them to access their accounts, manage bookings, and perform other authorized actions.

Login Page Features:

1. Username and Password Fields: Users enter their credentials to access their accounts.
2. Forgot Password Link: Users can reset their passwords if forgotten.
3. Registration Link: New users can create accounts.
4. Captcha or Security Measures: Prevents automated login attempts.
5. Error Handling: Displays error messages for invalid credentials.

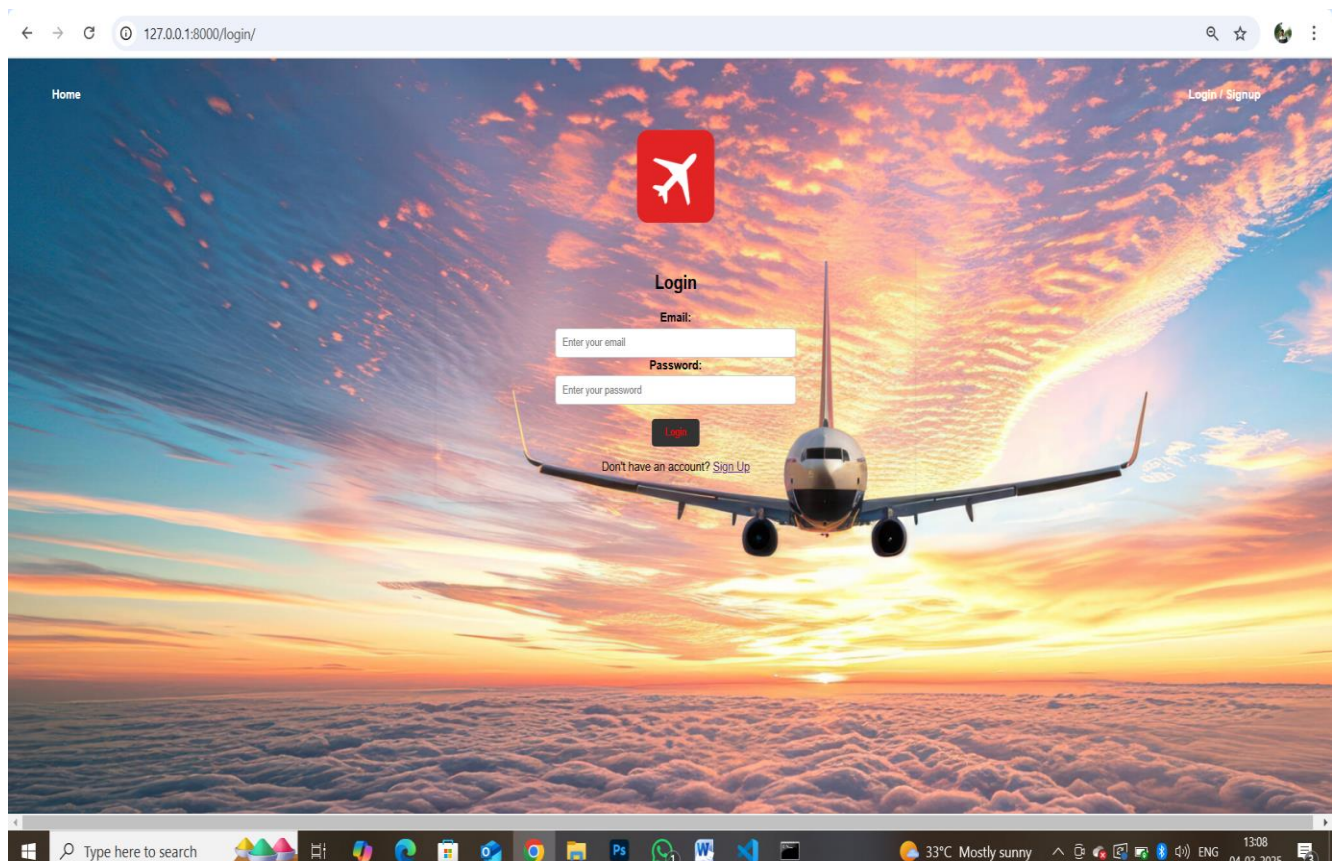
Login Page Requirements:

1. Authentication: Verifies user credentials against stored data.
2. Authorization: Grants access to authorized features and data.
3. Security: Protects user data and prevents unauthorized access.

4. Usability: Provides an intuitive and user-friendly interface.

Login Page Technologies:

1. Frontend: HTML, CSS, JavaScript .
2. Backend: Server-side programming languages (Python).
3. Database: Relational databases (MySQL,ITE).



Log out:

- After user book the ticket , they can log out their account and they can sign in again when their need to book a ticket

The logout page in a flight ticket booking system is a feature that allows users to securely exit their accounts, ensuring that their personal data and booking information remain protected.

Logout Page Features:

1. Logout Button: Users can click to initiate logout.
2. Session Termination: Ends user session, invalidating access tokens.
3. Cookie Deletion: Removes session cookies.
4. Redirect: Redirects users to login page or homepage.

Logout Page Requirements:

1. Security: Ensures user data remains protected.
2. Session Management: Properly terminates user sessions.
3. User Experience: Provides clear logout confirmation.

Logout Page Technologies:

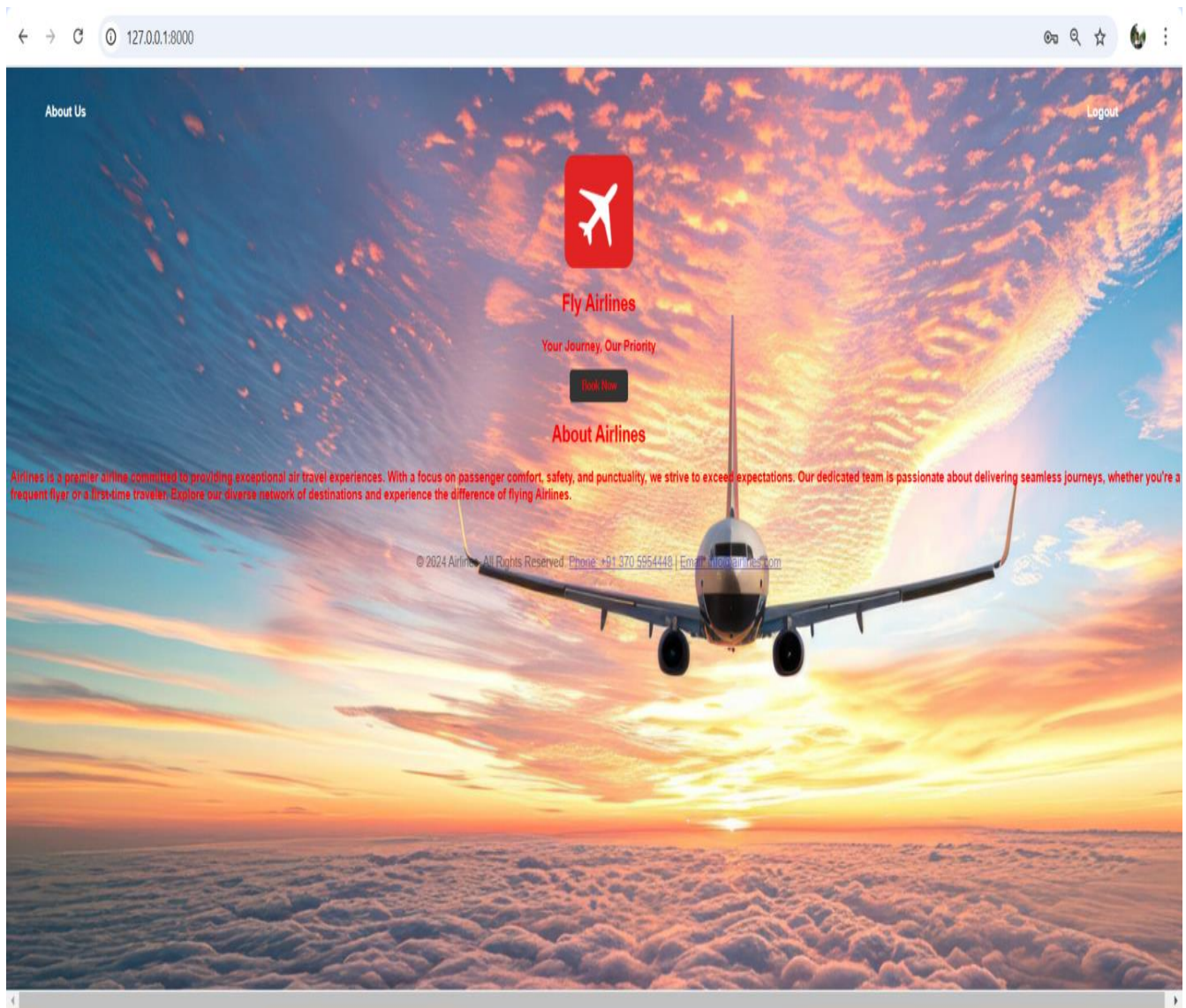
1. Frontend: HTML, CSS, JavaScript .
2. Backend: Server-side programming languages (Python).

Best Practices:

1. Use HTTPS: Ensure secure logout process.
2. Validate User Session: Verify user session before logout.
3. Clear Session Data: Remove sensitive data.
4. Test Logout: Verify logout functionality.

Common Logout Issues:

1. Session Expiration: Logout fails due to expired sessions.
2. Cookie Persistence: Cookies persist after logout.
3. Security Vulnerabilities: Logout process vulnerable to attacks.



Boarding Pass:

- The user can see their boarding pass as whenever they need to see they flight details
- In this website , the user can see their boarding Pass and the flight ticket details
- In this website, we developed the user can see the ticket details and their boarding pass.

A boarding pass is a document or electronic ticket issued by an airline to a passenger, indicating that they are cleared to board a flight.

Types of Boarding Passes:

1. Physical Boarding Pass: Printed document.
2. Electronic Boarding Pass (e-Boarding Pass): Digital ticket.
3. Mobile Boarding Pass: Mobile app-based ticket.

Information on Boarding Pass:

1. Flight Details: Flight number, departure and arrival airports, date, and time.
2. Passenger Information: Name, date of birth, and travel document details.
3. Seat Information: Seat number and class.
4. Barcode or QR Code: Unique identifier.

Boarding Pass Generation:

1. Flight Booking: Passenger books a flight.
2. Check-in: Passenger checks-in online or at airport.
3. Boarding Pass Issuance: System generates boarding pass.

Boarding Pass Technologies:

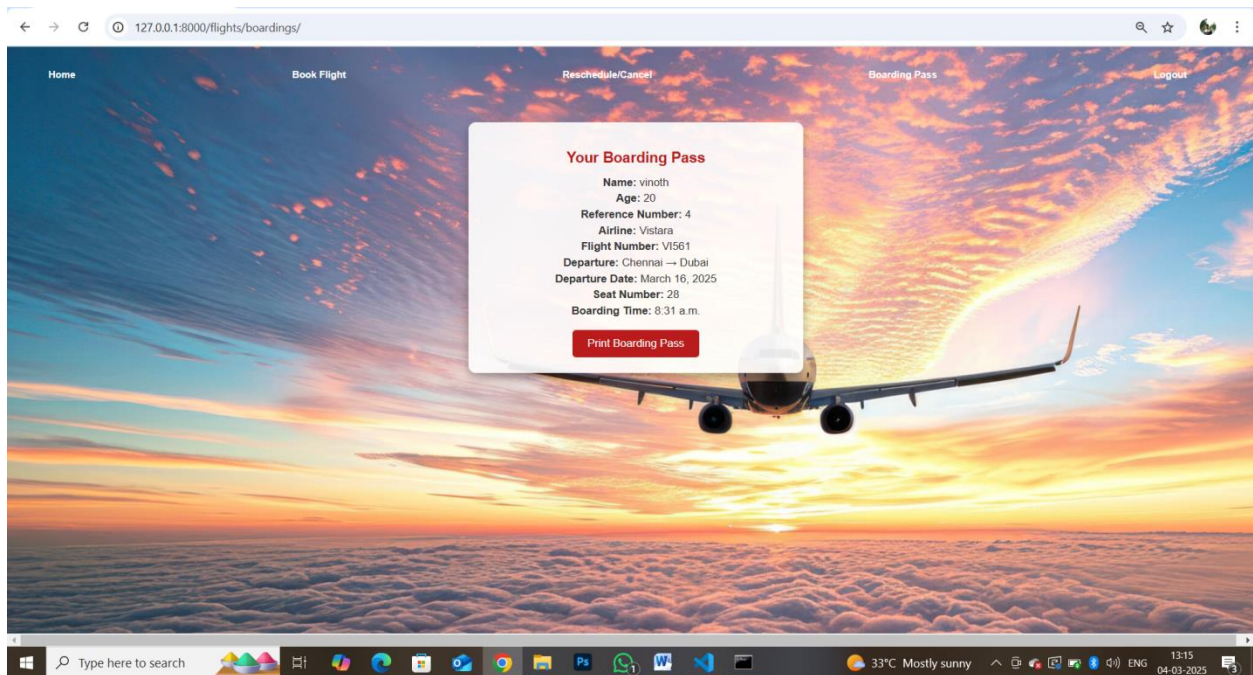
1. Frontend: HTML, CSS, JavaScript.
2. Backend: Server-side programming languages (Python).
3. Mobile Apps: Native or cross-platform apps.

Benefits of Electronic Boarding Passes:

1. Convenience: Easy to access and store.
2. Reduced Paper Waste: Environmentally friendly.
3. Increased Security: Less prone to loss or theft.

Common Boarding Pass Issues:

1. Lost or Misplaced: Passengers lose their boarding passes.
2. Invalid or Expired: Passes become invalid due to expired flights.



Manage the Ticket:

Ticket Management Features:

1. Ticket Issuance: Generate and issue tickets to passengers.
2. Ticket Cancellation: Cancel tickets and refund passengers.
3. Ticket Modification: Update ticket information, such as seat selection.
4. Ticket Expiration: Manage ticket expiration dates.
5. Ticket Refund: Process refunds for cancelled or modified tickets.

Ticket Management Requirements:

1. Data Accuracy: Ensure accurate ticket information.
2. Security: Protect ticket data from unauthorized access.
3. Compliance: Adhere to airline and regulatory requirements.
4. Scalability: Handle large volumes of ticket transactions.

Ticket Management Technologies:

1. Frontend: HTML, CSS, JavaScript.
2. Backend: Server-side programming languages (Python).
3. Database: Relational databases (MySQLITE).

Benefits of Effective Ticket Management:

1. Improved Passenger Experience: Accurate and timely ticket information.
2. Increased Efficiency: Automated ticket processing.
3. Reduced Errors: Minimized risk of ticket-related errors.

Common Ticket Management Issues:

1. Ticket Loss: Passengers lose their tickets.
2. Ticket Expiration: Tickets expire due to missed flights.
3. Technical Issues: System errors prevent ticket processing.

Rescheduling/ Cancel Ticket:

- In this website, we created the rescheduling/ cancel option for the user convenor
- In this option, the user are friendly interact with this website for their convenor easily.

Reschedule/cancel refers to the process of modifying or cancelling a flight booking, including changing flight dates, times, or routes.

Reschedule/Cancel Features:

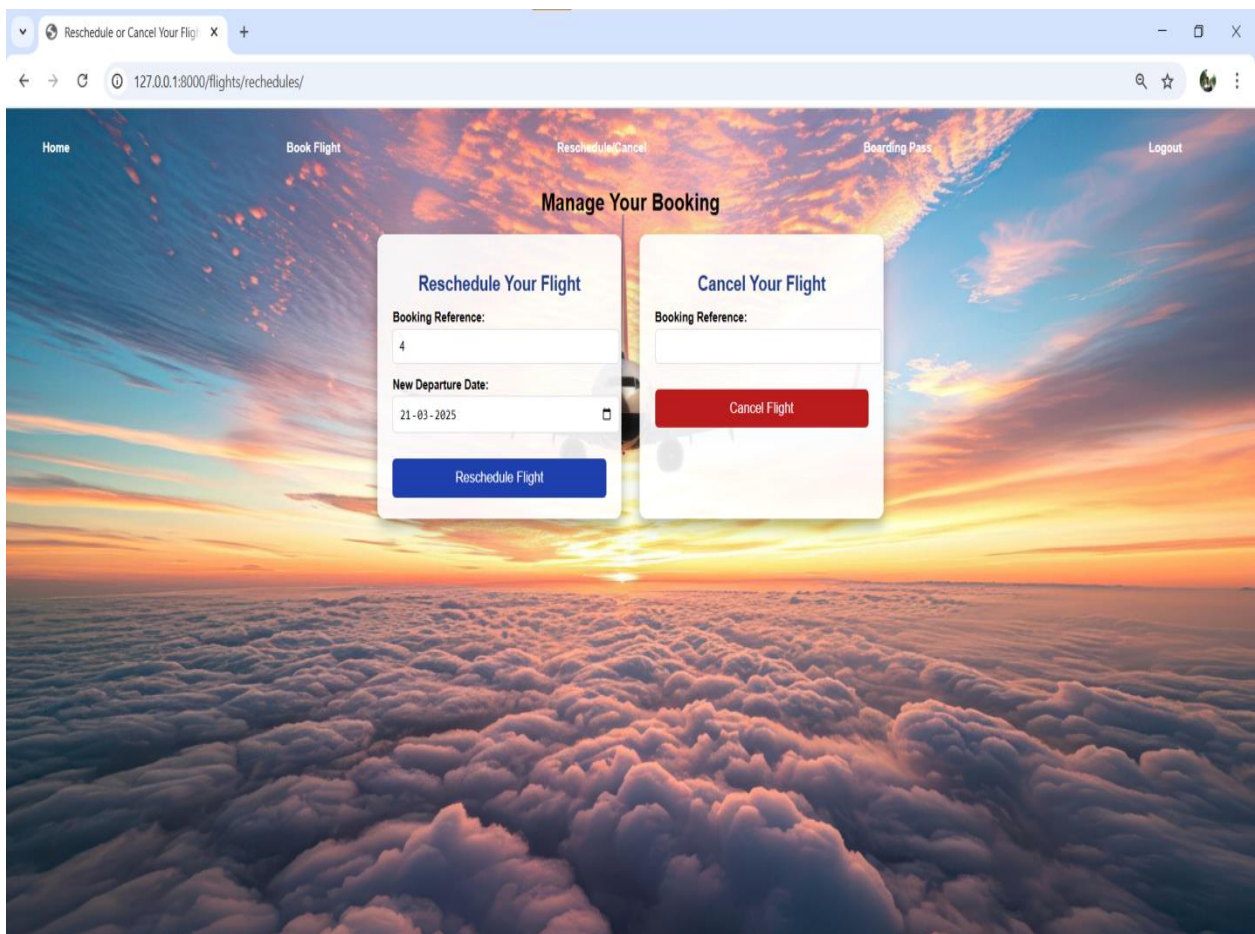
1. Flight Rescheduling: Change flight dates, times, or routes.
2. Flight Cancellation: Cancel flight bookings.
3. Refund Processing: Process refunds for cancelled flights.
4. Penalty Application: Apply penalties for rescheduling or cancelling flights.

Reschedule/Cancel Requirements:

1. Data Accuracy: Ensure accurate flight information.
2. Security: Protect flight data from unauthorized access.
3. Compliance: Adhere to airline and regulatory requirements.
4. Scalability: Handle large volumes of reschedule/cancel requests.

Reschedule/Cancel Technologies:

1. Frontend: HTML, CSS, JavaScript.
2. Backend: Server-side programming languages (Python).
3. Database: Relational databases (MySQLITE).



Benefits of Effective Reschedule/Cancel:

1. Improved Passenger Experience: Flexible rescheduling options.
2. Increased Efficiency: Automated reschedule/cancel processing.
3. Reduced Errors: Minimized risk of reschedule/cancel-related errors.

Common Reschedule/Cancel Issues:

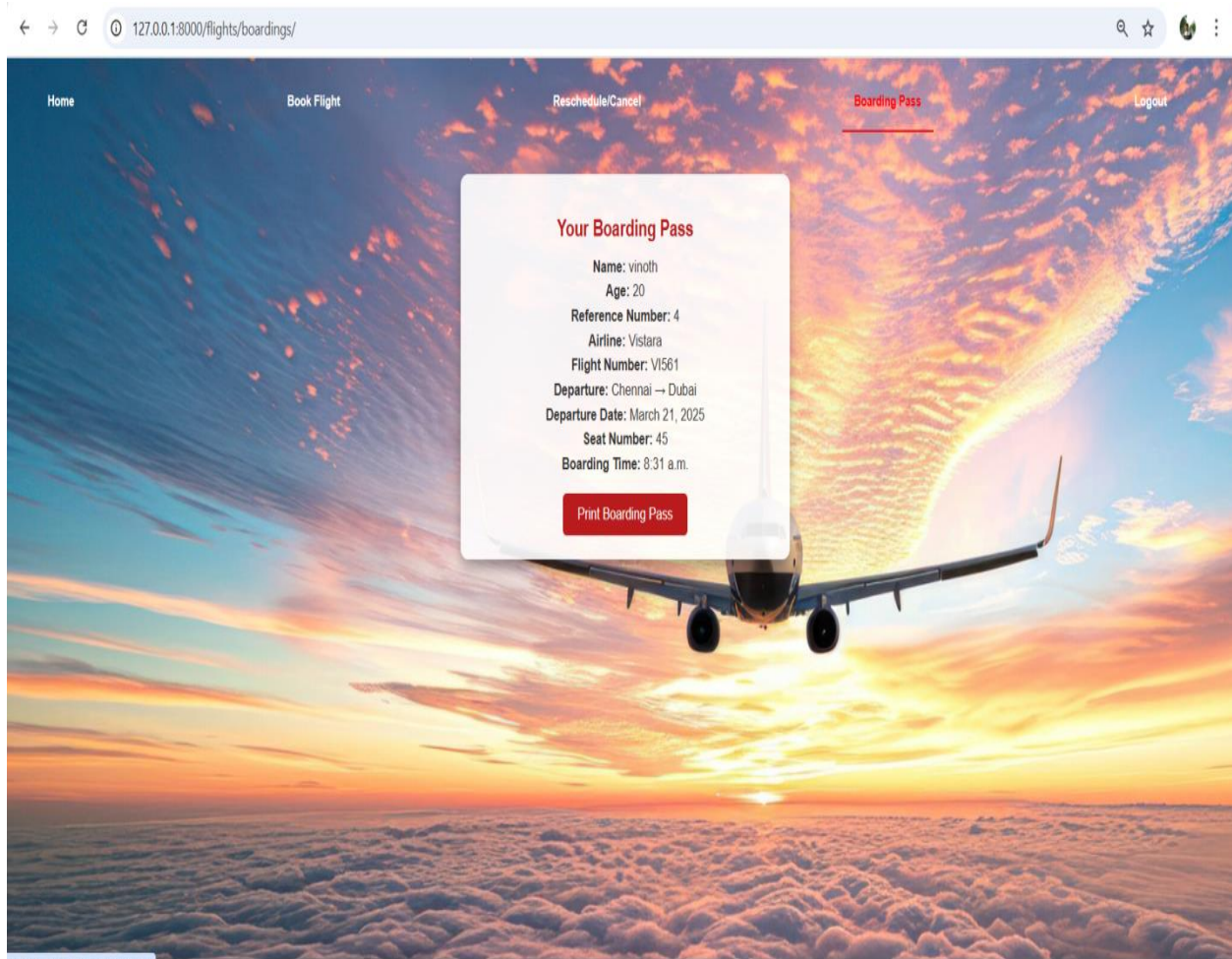
1. Flight Availability: Limited flight options.
2. Penalty Application: Incorrect penalty application.
3. Technical Issues: System errors prevent reschedule/cancel.

Reschedule :

- In this option, the user can reschedule their ticket as the convenor by using the reference number
- After their reschedule the flight ticket the user can see the current boarding pass.

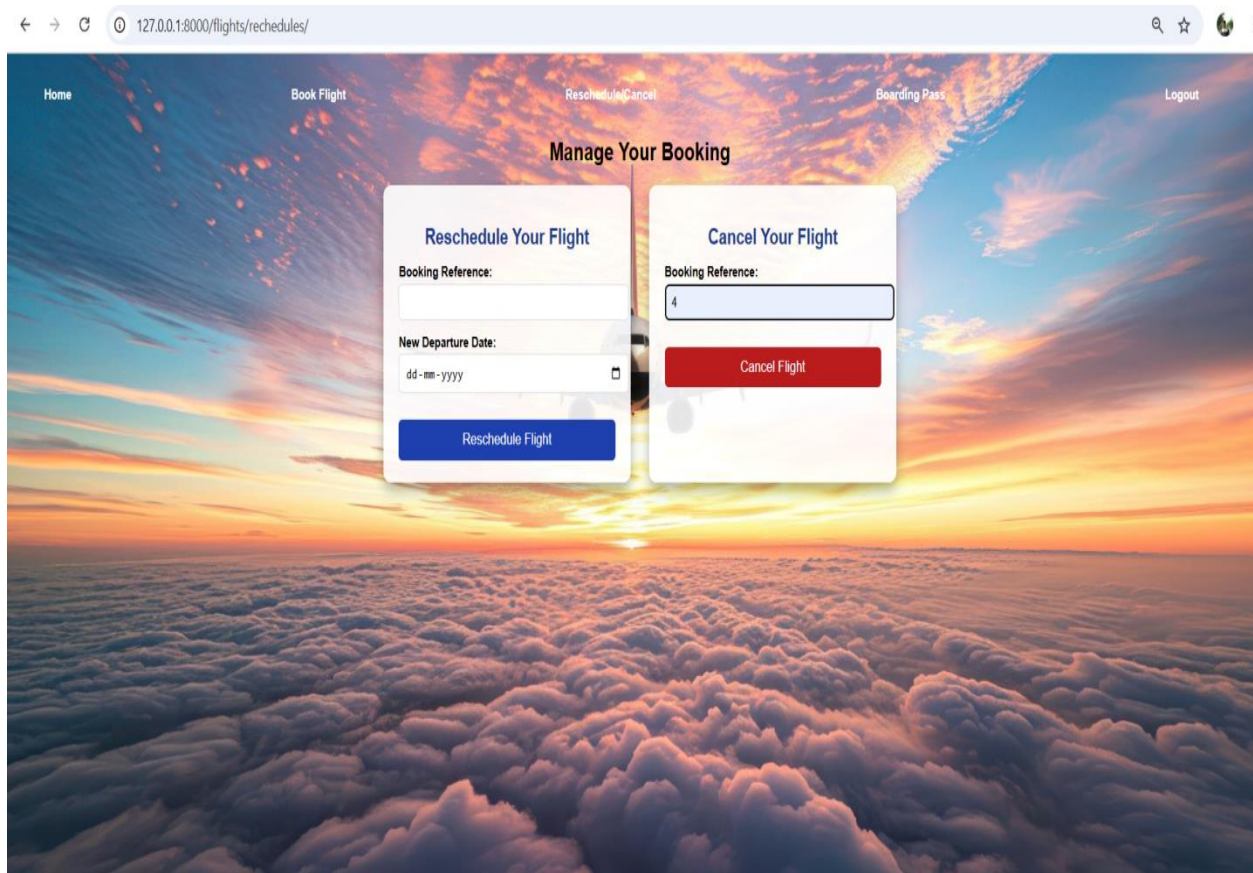
Reschedule boarding pass:

- After the user Rescheduling their boarding pass ,the user can check the new boarding pass flight ticket in the boarding pass option
- The online flight ticket reschedule can be done by reference number.
- When the user need to change the flight ticket time or date can be change in the rescheduling option



Cancel your Ticket:

- In this website, the user can cancel the flight ticket by using the reference number.
- The reference number can help us to change the reschedule the ticket and cancel the ticket.
- The online flight ticket can be cancelled easily.



The screenshot displays a web browser window with the address bar showing "127.0.0.1:8000/flights/rechedules/". The website has a navigation bar with links: Home, Book Flight, Reschedule/Cancel, Boarding Pass, and Logout. The main heading is "Manage Your Booking". Below this, there are two side-by-side forms. The left form is titled "Reschedule Your Flight" and contains a "Booking Reference:" field, a "New Departure Date:" field with a date picker (format dd-mm-yyyy), and a blue "Reschedule Flight" button. The right form is titled "Cancel Your Flight" and contains a "Booking Reference:" field with the number "4" entered, and a red "Cancel Flight" button. The background of the website is a scenic image of a sunset over a sea of clouds.

After Cancellation Ticket:

- After the user cancel the flight ticket , the user can check the boarding pass if the ticket are cancelled or not.
- When the user cancel the online flight ticket, they can log out the account or they can rebook the flight ticket.

After cancellation ticket refers to the process of handling and updating ticket information after a cancellation request has been made.

After Cancellation Ticket Features:

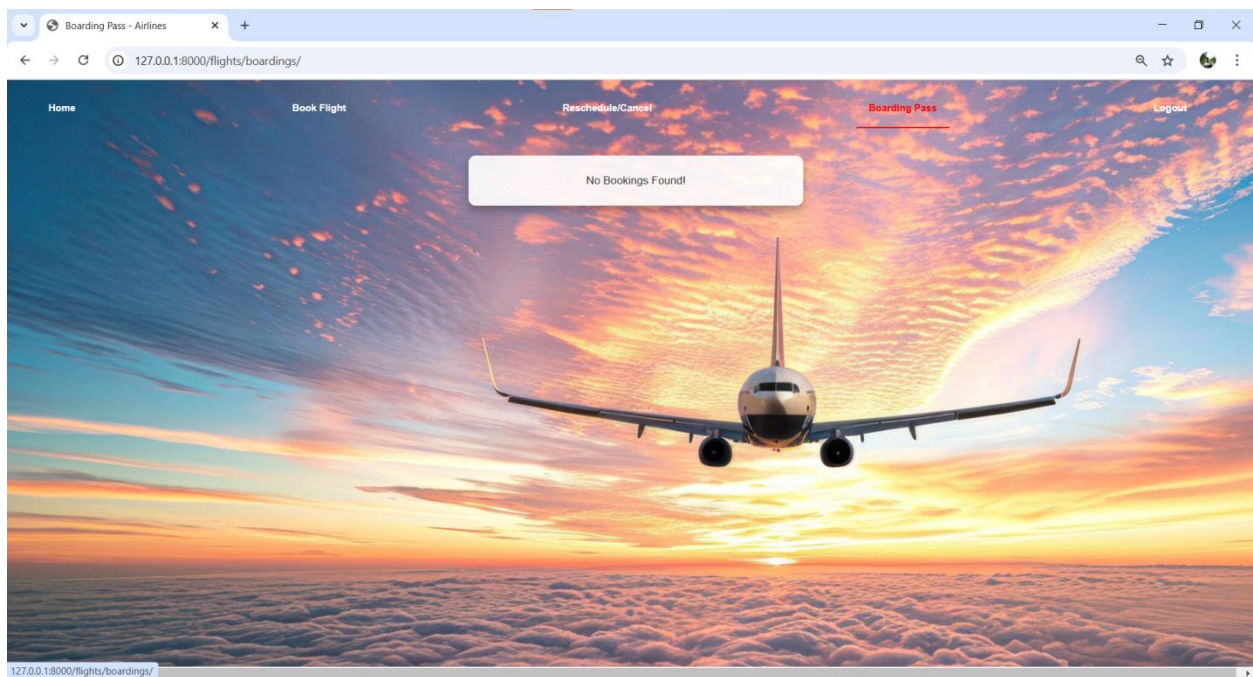
1. Ticket Cancellation Confirmation: Confirm cancellation request.
2. Refund Processing: Process refunds for cancelled tickets.
3. Ticket Expiration: Update ticket expiration date.
4. Penalty Application: Apply penalties for cancellations.
5. Notification: Send notifications to passengers.

After Cancellation Ticket Requirements:

1. Data Accuracy: Ensure accurate ticket information.
2. Security: Protect ticket data from unauthorized access.
3. Compliance: Adhere to airline and regulatory requirements.
4. Scalability: Handle large volumes of cancellation requests.

After Cancellation Ticket Technologies:

1. Frontend: HTML, CSS, JavaScript.
2. Backend: Server-side programming languages (Python).
3. Database: Relational databases (MySQLITE).



Benefits of Effective After Cancellation Ticket:

1. Improved Passenger Experience: Smooth cancellation process.
2. Increased Efficiency: Automated refund processing.
3. Reduced Errors: Minimized risk of cancellation-related errors.

Common After Cancellation Ticket Issues:

1. Refund Processing: Delayed or incorrect refunds.
2. Penalty Application: Incorrect penalty application.
3. Technical Issues: System errors prevent cancellation.

Database:

- The database are used to store the different data type in the file.
- The database are help us to store multiple file in the single file
- Database are used sqlite 3 version for store the database.

A database is a collection of organized data that stores information about flights, passengers, bookings, and payments.

Database Requirements:

1. Data Storage: Store flight schedules, passenger information, bookings, and payments.
2. Data Retrieval: Retrieve data efficiently to support flight search, booking, and cancellation operations.
3. Data Security: Protect sensitive data from unauthorized access.
4. Scalability: Handle large volumes of data and user traffic.

Database Design:

1. Tables: Flights, Passengers, Bookings, Payments.
2. Columns: Flight ID, Departure Airport, Arrival Airport, Departure Date, Passenger Name, Email, Phone, Booking ID, Payment Method.
3. Relationships: Flights to Bookings, Passengers to Bookings.

Database Technologies:

1. Relational Databases: MYSQLITE.
2. NoSQL Databases: MongoDB.
3. Cloud Databases: Amazon Aurora, Google Cloud SQL.

SELECT * FROM flights_booking				
Schema Query Editor Auto Reload Find				
	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	booking_date datetime NOT NULL	user_id bigint NOT NULL REFERENCES user_simpleuser(id) idx ²	flight_id bigint NOT NULL REFERENCES flights_flight(id) idx ¹
1	1	2025-02-10 06:23:40...	1 password:'pbkdf2...	1 fullname:'gokul'...
2	5	2025-03-04 07:21:10...	5 password:'pbkdf2...	5 fullname:'vignes...
3	6	2025-03-04 07:22:41...	6 password:'pbkdf2...	6 fullname:'bharat...
4	7	2025-03-04 07:24:15...	7 password:'pbkdf2...	7

SELECT * FROM flights_flight									
	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	fullname varchar(100) NOT NULL	age integer unsigned NOT NULL	gender varchar(20) NOT NULL	airline varchar(100) NOT NULL	flight_number varchar(20) NOT NULL UNIQUE	departure varchar(100) NOT NULL	destination varchar(100) NOT NULL	boarding
1	1	gokul	20	Male	Air India	AI517	Bangalore	Mumbai	03:12:00
2	3	vinoth	20	Male	Air India	AI964	Chennai	Delhi	13:41:00
3	5	vignesh	20	Male	GoAir	G0704	Bangalore	London	07:07:00
4	6	bharathi	20	Male	Indigo	IN975	Hyderabad	Delhi	10:46:00
5	7	nithya	28	Female	Emirates	EM359	Delhi	Ahmedabad	12:28:00
+									

SELECT * FROM flights_flight								
	flight_number varchar(20) NOT NULL UNIQUE	departure varchar(100) NOT NULL	destination varchar(100) NOT NULL	boarding_time time	departure_date date NOT NULL	return_date date	class_type varchar(20)	user_id bigint REFERENCES user_simpleuser(id) idx
1	AI517	Bangalore	Mumbai	03:12:00	2025-02-12	2025-02-22	first	NULL
2	AI964	Chennai	Delhi	13:41:00	2025-02-27	2025-03-06	business	NULL
3	G0704	Bangalore	London	07:07:00	2025-03-06	2025-03-22	economy	NULL
4	IN975	Hyderabad	Delhi	10:46:00	2025-03-16	2025-03-29	first	NULL
5	EM359	Delhi	Ahmedabad	12:28:00	2025-03-15	2025-03-29	business	NULL

- After the user checkup their details, the can logout the account in the website.

12.OVERVIEW

Here's a detailed outline for an online flight ticket booking website:

- **Website Features**
- **Technical Requirements**
- **Developments Steps**
- **Future Enhancements**
- **Benefits**
- **Revenue Streams**

✓ **Website Features:**

1. Flight Search:

Users can search for flights by origin, destination, departure date, and return date.

2. Flight Results:

Display a list of available flights with details such as flight number, departure and arrival times, and prices.

3. Flight Booking:

Users can book flights by selecting the desired flight and entering passenger details.

4. Payment Gateway:

Integrate a payment gateway to facilitate secure online payments.

5. Booking Confirmation:

Send a confirmation email to users with booking details and flight itinerary.

6. User Account:

Allow users to create an account to view booking history, cancel or modify bookings, and save passenger details.

7. Admin Panel:

Provide an admin panel for managing flight schedules, prices, and bookings.

✓ Technical Requirements

1. Frontend:

HTML, CSS, JavaScript (with a framework like React or Angular).

2. Backend:

Node.js (with Express.js) or Python (with Flask or Django).

3. Database:

Relational database (e.g., MySQL) or NoSQL database (e.g., MongoDB).

4. API Integration:

Integrate with flight APIs (e.g., Amadeus, Sabre, or Expedia) to retrieve flight data.

5. Payment Gateway Integration:

Integrate with payment gateways (e.g., PayPal, Stripe, or (link unavailable)) to facilitate secure payments.

✓ **Development Steps**

1. Plan the database schema:

Design the database structure to store flight data, booking information, and user details.

2. Set up the backend:

Create the backend API using Node.js or Python.

3. Develop the frontend:

Build the frontend using HTML, CSS, and JavaScript.

4. Integrate with flight APIs:

Integrate with flight APIs to retrieve flight data.

5. Integrate with payment gateways:

Integrate with payment gateways to facilitate secure payments.

6. Test and deploy:

Test the website thoroughly and deploy it to a production environment.

✓ **Future Enhancements**

1. Mobile App:

Develop a mobile app for the website to provide a seamless booking experience on-the-go.

2. Personalized Recommendations:

Implement a recommendation engine to suggest flights based on user preferences and booking history.

3. Real-time Updates:

Provide real-time updates on flight schedules, prices, and availability.

4. Social Sharing:

Allow users to share their booking details on social media platforms.

5. Loyalty Program:

Implement a loyalty program to reward repeat customers with discounts, free flights, or other perks.

✓ Benefits

1. Convenience: Book flights online from anywhere, at any time.

2. Time-saving: Quickly compare prices and flight schedules.

3. Cost-effective: Find the best deals and discounts.

4. Easy booking: Book flights with just a few clicks.

5. Secure payments: Make secure payments online.

✓ Revenue Streams

- 1. Commission-based bookings:** Earn a commission for each booking made through the website.
- 2. Advertising:** Display ads from airlines, travel agencies, or other travel-related businesses.
- 3. Affiliate marketing:** Partner with airlines or travel agencies to earn a commission for each booking made through the website.
- 4. Premium services:** Offer premium services, such as flight insurance or travel assistance, for an additional fee.

13.CONCLUSION:

- This website will help us to understand about the flight booking process from the admin side and as well as from the user side. This website will reduce time while when want to book any flight urgently . It has a friendly interface and easy to understand .
- The system is having some benefits for customers too. The webpage will work on any platforms through networks. By this Customers can book and cancel the flight ticket from this website. So, overall this System will give the customers a better user friendly environment to book the online flight ticket.

Reference Links:

For frontend: https://www.w3schools.com/howto/howto_blog_become_frontenddev.asp

Backend development: <https://learnpython.org> , <https://t.me/pythondevelopersindia/314>

For database: Database: <https://mode.com/sql-tutorial/introduction-to-sql>

<https://www.sqltutorial.org/wp-content/uploads/2016/04/SQL-cheat-sheet.pdf>

<https://books.goalkicker.com/MySQLBook/MySQLNotesForProfessionals.pdf>

Reference Books:

HTML and CSS: Design and Build Websites by Jon Duckett.

Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics by Jennifer Niederst Robbins

JavaScript: The Good Parts" by Douglas Crockford

Django using python: by Eric Matthes.

MySQLITE: Web Development with Live Projects" by Kevin Yank.