

Chapter - 1 Project Overview

The Flight Management System is a Java-based booking solution for flight tickets. It consolidates data provided by different airline carriers and hence provides the user details and rates in real-time. Travellers may want to make changes in their bookings. The application allows them to book, cancel and view their bookings with ease. Other than this, it eases the management of bookings too. All the bookings, flights, schedules and routes can be viewed, added and modified on a single application by the administrator.

Chapter - 2 Features

This website provides Features like

- Passenger Management
- Flight scheduling and management
- Date wise Booking
- Flexible ticket booking
- User Feedback

Chapter -3 Modules And Functionalities

“Flight management system” provides two modules:

- Admin
- Customer

Admin

Admin can manage and view Airports Routes, Flights details. Admin can also view user feedbacks, ticket details and passenger details.

The functionalities provided for admin module are:

- Login
- Add/Update/View Airports
- Add/Update/View Routes
- Add/Update/View Flights
- View User Feedbacks
- View User Details
- View Ticket Details
- Logout

Customer

Customer can book flights, cancel bookings and give feedback.

The functionalities provided in this module consist of:

- Create user account
- Login
- Check for available flights
- Book/View flight details
- Give Feedback
- Logout

Chapter - 4. Technologies

❖ Frontend:

- JSP
- CSS
- Java script
- Bootstrap

❖ Backend:

- Java
- Spring Boot
- Spring Security
- Spring Data JPA

❖ Database:

- MySQL

Chapter-5. Design

5.1 Data Dictionary

➤ airport

Field Name	Data Type	Relational Key
airport_code	Varchar	Primary Key
airport_location	Varchar	-

Table 5.1.1 airport

➤ feedback

Field Name	Data Type	Relational Key
username	Varchar	Primary Key
content	Longtext	-

Table 5.1.2 feedback

➤ flight

Field Name	Data Type	Relational Key
flight_number	Bigint	Primary Key
arrival_time	Varchar	-
carrier_name	Varchar	-
departure_time	Varchar	-
route_id	Bigint	-
seat_booked	Int	-
seat_capacity	Int	-

Table 5.1.3 flight

➤ flight_datewise

Field Name	Data Type	Relational Key
date	Varchar	Primary Key
flight_number	Bigint	Primary Key
seat_booked	Int	-

Table 5.1.4 flight_datewise

➤ flight_user

Field Name	Data Type	Relational Key
username	Varchar	Primary Key
password	Varchar	-
type	Varchar	-

Table 5.1.5 flight_user

➤ passenger

Field Name	Data Type	Relational Key
serial_number	Int	Primary Key
ticket_number	Bigint	Primary Key
fare	Double	-
passenger_dob	Varchar	-
passenger_name	Varchar	-

Table 5.1.6 passenger

➤ route

Field Name	Data Type	Relational Key
route_id	Bigint	Primary Key
destination_airport_code	Varchar	-
source_airport_code	Varchar	-
ticket_cost	Double	-

Table 5.1.7 route

➤ ticket

Field Name	Data Type	Relational Key
ticket_number	Bigint	Primary Key
carrier_name	Varchar	-
flight_number	Bigint	-
route_id	Bigint	-
total_amount	Double	-
date	Varchar	-

Table 5.1.8 ticket

5.2 Class and Method Description

5.2.1 DTO Layer

1) Airport: This class stores the details of an airport.

Attributes:

airportCode: String
airportLocation: String

Methods:

Getter and setter methods for all attributes

2) Feedback: This class stores the details of feedback.

Attributes:

username: String
content: String

Methods:

Getter and setter methods for all attributes

3) Flight: This class stores all the details of a flight.

Attributes:

flightNumber: Long
carrierName: String
routeId: Long
seatCapacity: Integer
departureTime: String
arrivalTime: String
seatBooked: Integer

Methods:

Getter and setter methods for all attributes

4) FlightDateEmbed: This class stores the flight number and date.

Attributes:

flightNumber: Long
date: String

Methods:

Getter and setter with hashCode() and equals() method

5) FlightDatewise: This class stores the seat booked on a particular flight on a particular date.

Attributes:

embeddedId: FlightDateEmbed
seatBooked: Integer

Methods:

Getter and setter for all attributes

6) FlightUser: This class stores the user type(admin or customer) and all user information

Attributes:

username: String
password: String
type: String

Methods:

Getter and setter for all attributes

7) Route: This class stores route details.

Attributes:

routeId: Long
sourceAirportCode: String
destinationAirportCode: String
ticketCost: Double

Methods:

Getter and setter for all attributes

8) Ticket: This class stores ticket details.

Attributes:

ticketNumber: Long
routeId: Long
flightNumber: Long
carrierName: String
totalAmount: Double
date: String

Methods:

Getter and setter for all attributes

9) TicketPassengerEmbed: This class stores the ticket number and serial number.

Attributes:

ticketNumber: Long
serialNumbe: Integer

Methods:

Getter and setter with hashCode() and equals() method

10) Passenger: This class stores the seat booked on a particular flight on a particular date.

Attributes:

embeddedId: TicketPassengerEmbed
passengerName: String
passengerDOB: String
fare: Double

Methods:

Getter and setter for all attributes

5.2.2 Service Layer

1) FlightService:

Attributes:

flightDao: FlightDao

routeDao: RouteDao

Methods:

- i) createReturnFlight(Flight flight, String dtime, String atime) : Flight
 - Creates a return flight based on the given flight details.
- ii) getFlightsByDate(String date): List<Flight>
 - Retrieves flights by date and sets the number of seats booked for each flight.
- iii) setSeatBookedToZero(): void
 - Sets the number of seats booked to zero for all flights.

2) FlightUserService:

Attributes:

repository : FlightUserRepository

type: String

Methods:

- i) save(FlightUser user): void
 - Saves the user details in the database.
- ii) getType(): String
 - Retrieves the type of the current user.
- iii) loadUserByUsername(String username): UserDetails
 - Loads the user by their username.

3) RouteService:

Attributes: -

Methods:

- i) createReturnRoute(Route route): Route
 - Creates a return route based on the given route. The return route will have the source and destination airport codes swapped and a new route ID.

4) TicketService:

Attributes:

flightDao: FlightDao

flightDatewiseDao: FlightDatewiseDao

Methods:

- i) discountCalculation(Passenger passenger): Double
 - Calculates the discounted fare based on passenger age.
 - Passengers 14 years old or younger get a 50% discount.
 - Passengers 60 years old or older get a 30% discount.
- ii) ageCalculation(String passengerDOB): Integer
 - Calculates the age of the passenger based on their date of birth.
- iii) capacityCalculation(int numberOfSeatBooking, long flightNumber, String date) : **Boolean**
 - Checks if there is enough capacity on the flight for the given number of seats. If capacity allows, updates the booked seat count.
- iv) ticketCancellation(String date, Long flightNumber, int noOfPassengers): void
 - Cancels the specified number of tickets and updates the booked seat count.

5.2.3 DAO Layer

1) AirportDaoImpl:

Attributes:

repository: AirportRepository

Methods:

- i) addAirport(Airport **airport**): void
 - saves airport object in database
- ii) showAllAirports(): List<Airport>
 - returns all details of all airports stored in database
- iii) showAirport(String **id**): Airport
 - returns airport details of airport where airport code is id
- iv) findAllAirportCodes(): List<String>
 - returns all airport codes stored in database
- v) findAllAirportLocation(): List<String>
 - returns all airport locations stored in database
- vi) findAirportLocationByCode(String **airportCode**): String
 - returns airport location whose airport code is airportCode
- vii) findAirportCodeByLocation(String **airportLocation**): String
 - returns airport code whose airport location is airportLocation

2) FeedbackDaoImpl:

Attributes:

repository: FeedbackRepository

Methods:

- i) save(Feedback **feedback**): void
 - saves feedback object in database
- ii) showAllFeedbacks(): List<Feedback>
 - returns all feedback stored in database

3) FlightDaoImpl:

Attributes:

repository: FlightRepository

Methods:

- i) addFlight(Flight **flight**): void
 - saves flight object in database

- ii) showAllFlights(): List<Flight>
 - returns all details of all flights stored in database
- iii) showFlight(Long id): Flight
 - returns flight details of flight where flight number is id
- iv) findAllFlightNumbers(): List<Long>
 - returns all flight numbers stored in database
- v) findByRouteId(Long routeId): List<Flight>
 - returns all flights from in database whose route id is routeId
- vi) findFlightsByDate(String date): List<Object[]>
 - returns flights and seat booked from flight datewise table where date is given date.

4) FlightDatewiseDaoImpl:

Attributes:

repository: FlightDatewiseRepository

Methods:

- i) save(FlightDatewise flightDatewise): void
 - saves flightDatewise object in database
- ii) findByDateAndFlightNumber(String date, Long flightNumber): FlightDatewise
 - returns flight datewise object where flight number is flightNumber and date is given date.

5) PassengerDaoImpl:

Attributes:

repository: PassengerRepository

Methods:

- i) save(Passenger passenger): void
 - saves passenger object in database
- ii) findByTicketNumber(Long ticketNumber): List<Passenger>
 - returns all passenger's details whose ticket number is ticketNumber
- iii) deletePassenger(Passenger p) : void
 - Deletes the given passenger record from the database.
- iv) showAllPassengerDetails(): List<Passenger>
 - Returns all passenger details stored in the database

6) RouteDaoImpl:

Attributes:

repository: RouteRepository

Methods:

- i) addRoute(Route route): void
 - saves route object in database
- ii) showRoute(Long id): Route
 - returns a route object whose route id is id
- iii) showAllRoutes(): List<Route>
 - Returns all route details of all routes stored in the database
- iv) findAllRouteId(): List<Long>
 - Returns all route id stored in the database
- v) findRouteBySourceAndDestination(String source, String destination): Route
 - returns route whose source and destination code matches the given source and destination code
- vi) generateRouteId(): Long
 - Find the last route id and return its value by adding one to it.

7) TicketDaoImpl:

Attributes:

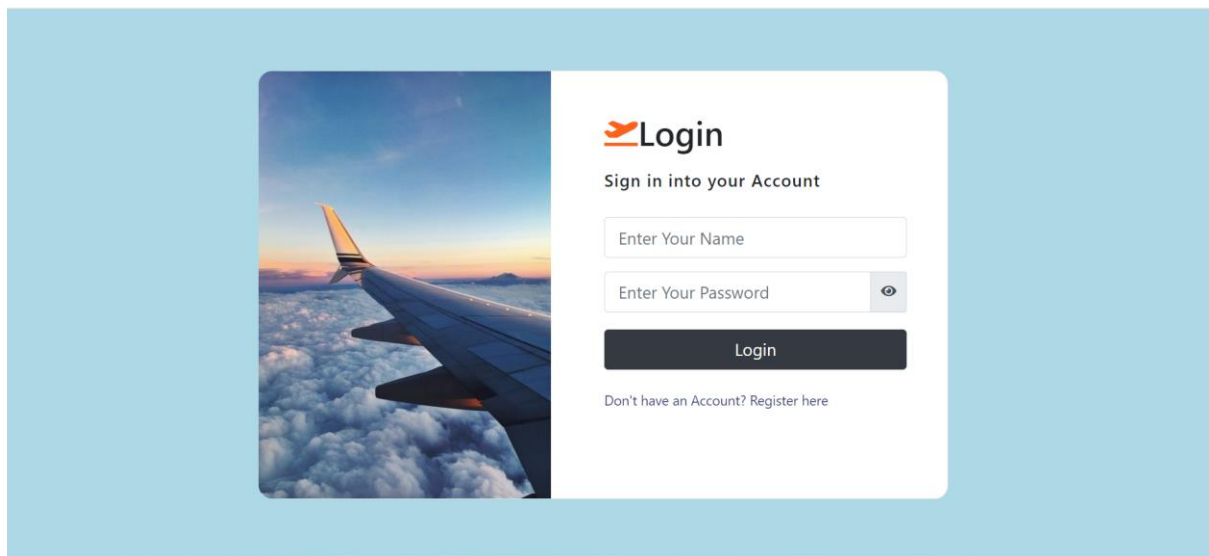
repository: TicketRepository

Methods:

- ii) save(Ticket ticket): void
 - saves ticket object in database
- ii) findLastTicketNumber(): Long
 - Find the last ticket number and return its value by adding one to it.
- iii) findByTicketNumber(Long ticketNumber): Ticket
 - Returns a ticket object whose ticket number is ticketNumber
- iv) cancelTicket(Long ticketNumber): void
 - Deletes record of the ticket whose ticket number is ticketNumber
- v) showAllTickets(): List<Ticket>
 - returns all ticket records stored in the database

Chapter-6. Implementation Screenshots

6.1 Login Page:



The screenshot shows a login page with a light blue background. On the left is a vertical image of an airplane wing flying over clouds at sunset. On the right is a white login form. The form has a logo with an orange bird icon and the word 'Login'. Below the logo is the text 'Sign in into your Account'. There are two input fields: 'Enter Your Name' and 'Enter Your Password' (with a toggle icon). Below these is a dark 'Login' button. At the bottom of the form is a link: 'Don't have an Account? Register here'.

Login

Sign in into your Account

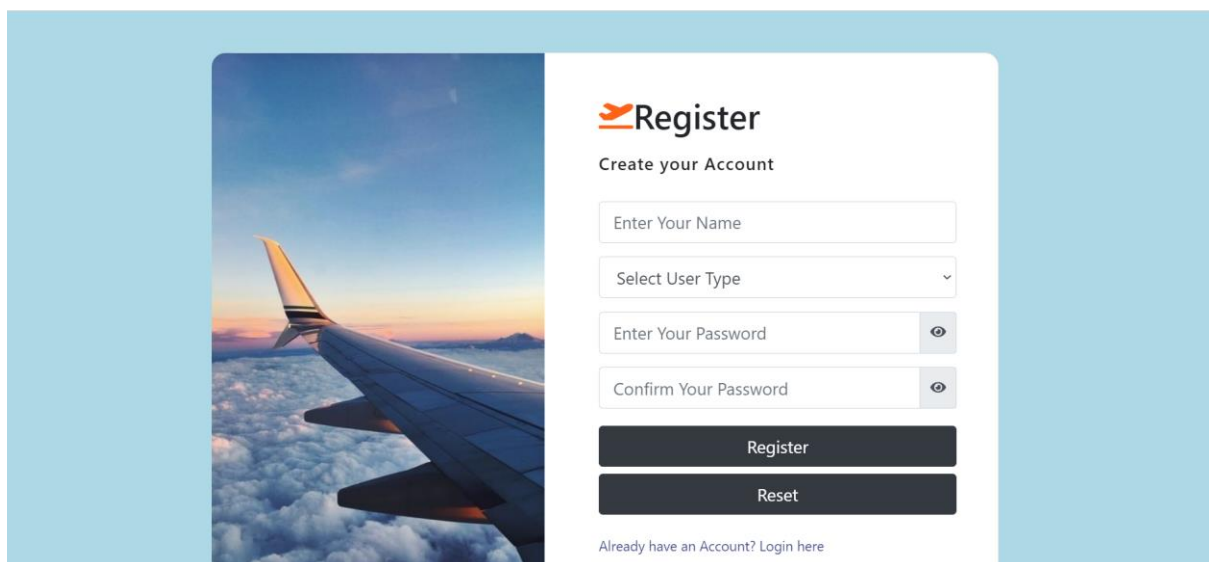
Enter Your Name

Enter Your Password

Login

Don't have an Account? Register here

6.2 Sign Up Page:



The screenshot shows a register page with a light blue background. On the left is a vertical image of an airplane wing flying over clouds at sunset. On the right is a white register form. The form has a logo with an orange bird icon and the word 'Register'. Below the logo is the text 'Create your Account'. There are four input fields: 'Enter Your Name', 'Select User Type' (with a dropdown arrow), 'Enter Your Password' (with a toggle icon), and 'Confirm Your Password' (with a toggle icon). Below these are two dark buttons: 'Register' and 'Reset'. At the bottom of the form is a link: 'Already have an Account? Login here'.

Register

Create your Account

Enter Your Name

Select User Type

Enter Your Password

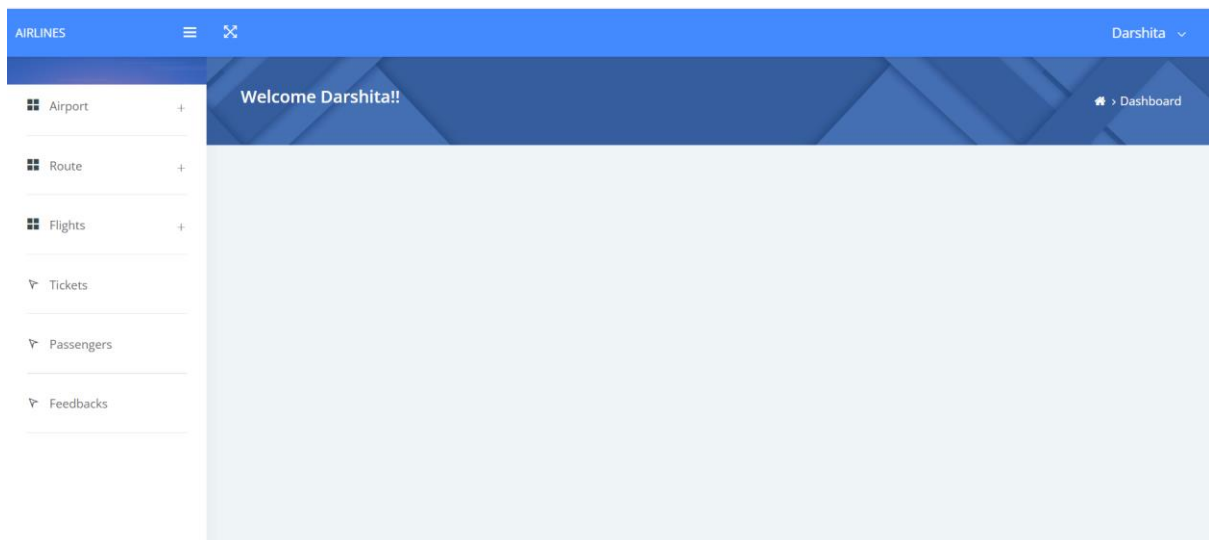
Confirm Your Password

Register

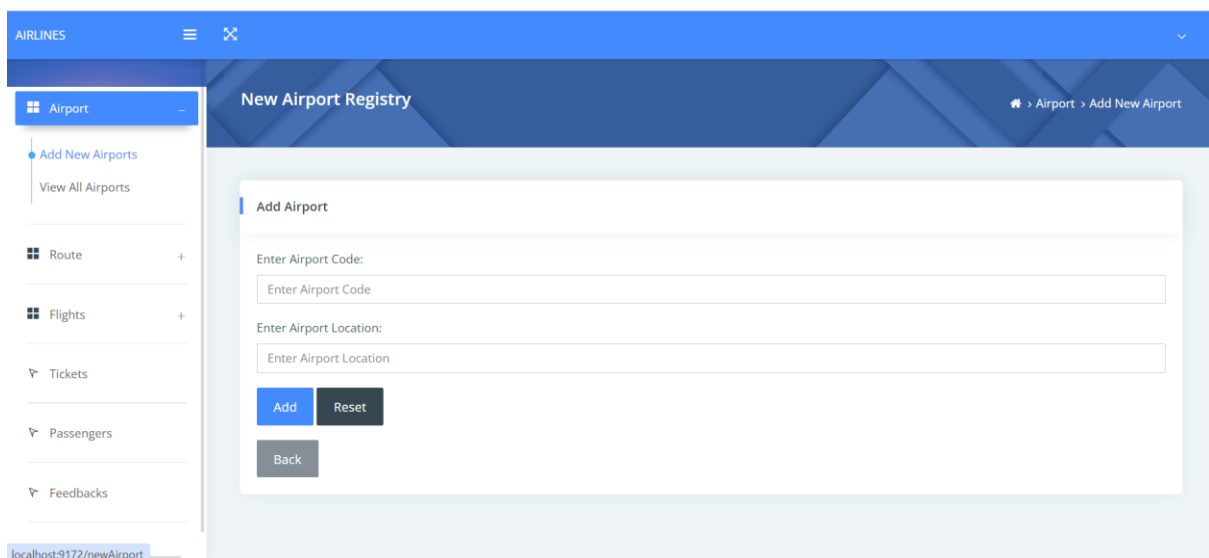
Reset

Already have an Account? Login here

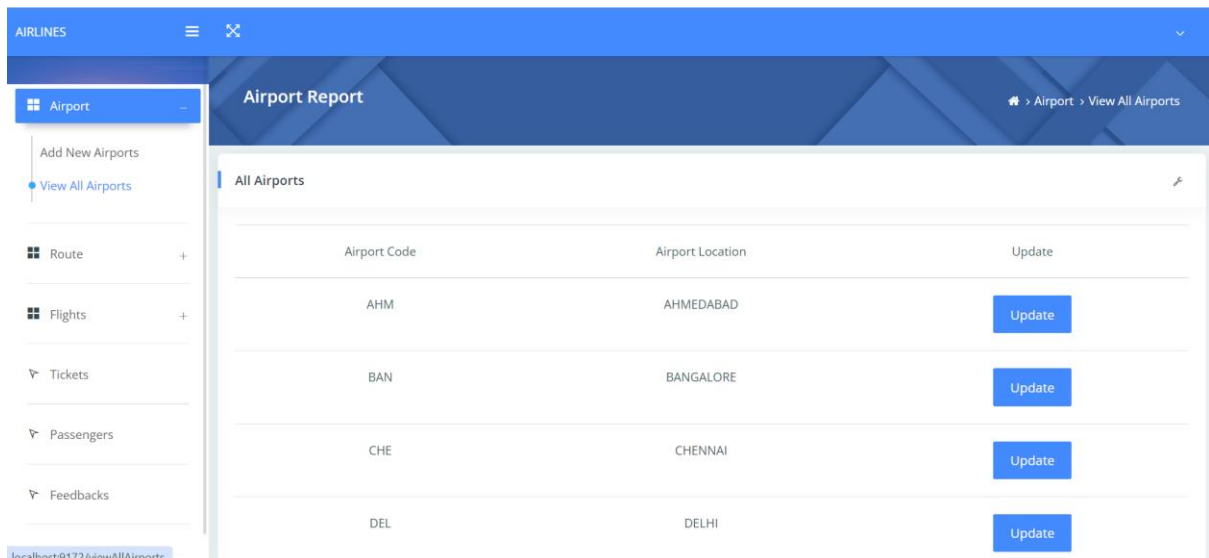
6.3 Admin Features:



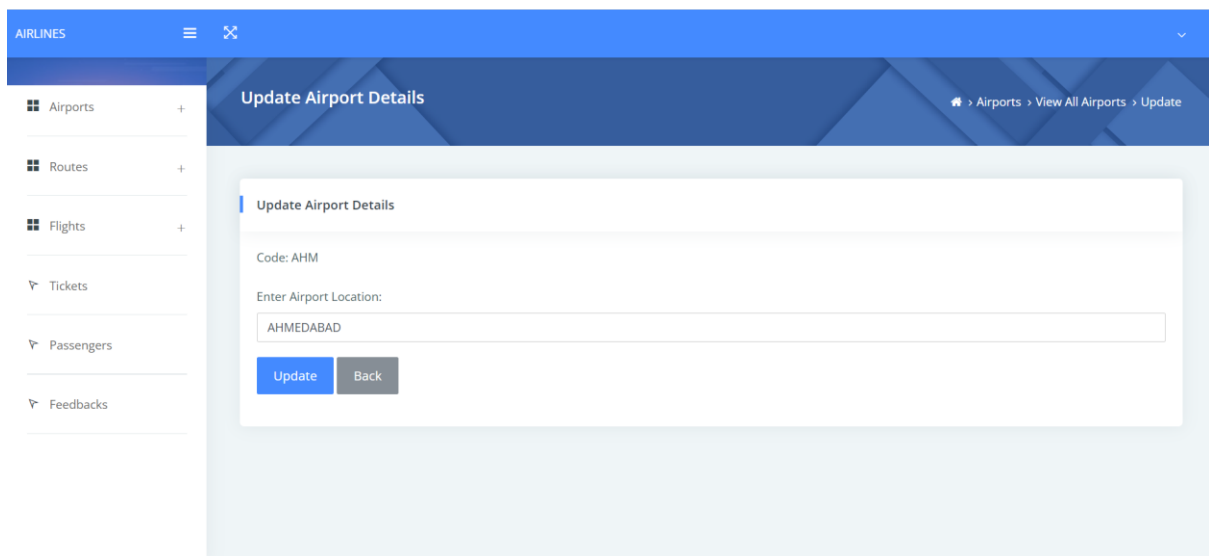
6.3.1 Admin Dashboard



6.3.2 Add New Airport Page



6.3.3 View All Airports Page



6.3.4 Update Airport Details Page

AIRLINES

Airport

+

Route

+

Add New Route

View All Routes

Flights

+

Tickets

Passengers

Feedbacks

localhost:9172/addRoute

New Route Registry

Route

Add New Route

Add Route

Select Source Airport City:

Select Source Airport City

Select Destination Airport City:

Select Destination Airport City

Enter Ticket Cost:

Enter Ticket Cost

Add

Reset

Back

6.3.5 Add New Route Page

AIRLINES

Airport

+

Route

+

Add New Route

View All Routes

Flights

+

Tickets

Passengers

Feedbacks

localhost:9172/viewAllRoutes

Route Report

Route

View All Routes

All Routes

Route Id	Source Airport Code	Destination Airport Code	Ticket Cost	Update
101	AHM	BAN	5000.0	Update
102	BAN	AHM	5000.0	Update
103	AHM	DEL	4500.0	Update
104	DEL	AHM	4500.0	Update

6.3.6 View All Routes Page

AIRLINES

Airport

+

Route

+

Flights

+

Tickets

Passengers

Feedbacks

Update Route Details

Route

>

View All Routes

>

Update Route Details

Update Route Details

Route Id: 102

Source Airport Code: BAN

Destination Airport Code: AHM

Enter Ticket Cost:

5000.0

Update

Back

6.3.7 Update Route Details Page

AIRLINES

Airport

+

Route

+

Flights

-

Add New Flight

View Flights Datewise

View/Update Flights Record

Tickets

Passengers

New Flight Entry

Flights

>

Add New Flight

Add Flight

Enter Flight Id:

Enter Flight Id

Enter Carrier Name:

Enter Airlines Name

Select Route ID:

Select Route ID:

Enter Seat Capacity:

Enter Seat Capacity

Enter Departure Time:

--:--

Enter Arrival Time:

--:--

Enter Return Flight's Departure Time:

--:--

Enter Return Flight's Arrival Time:

--:--

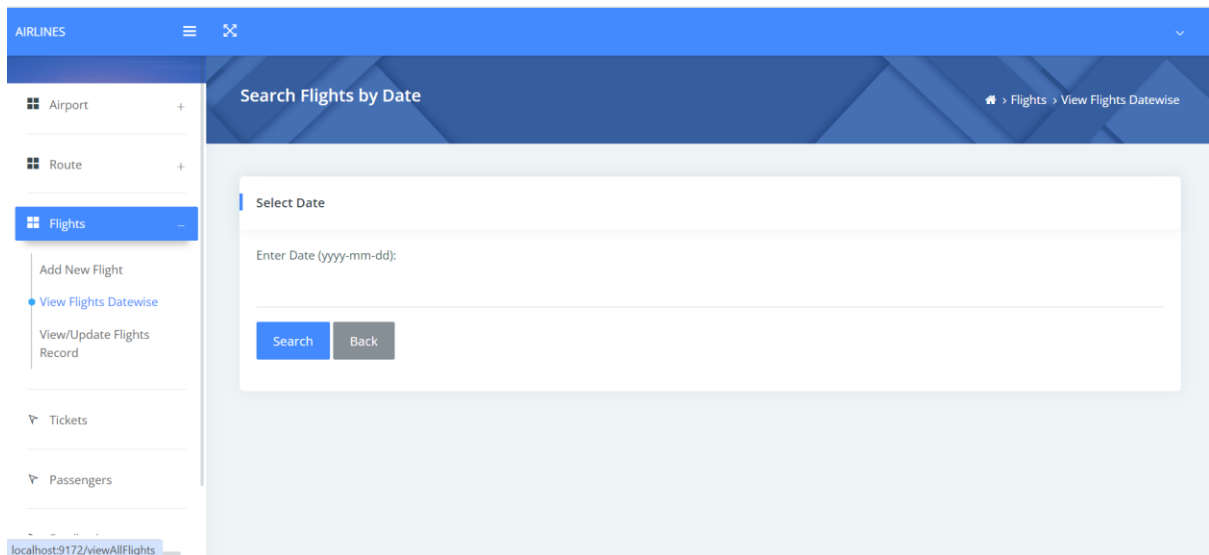
Add

Reset

Back

6.3.8 Add New Flight Page

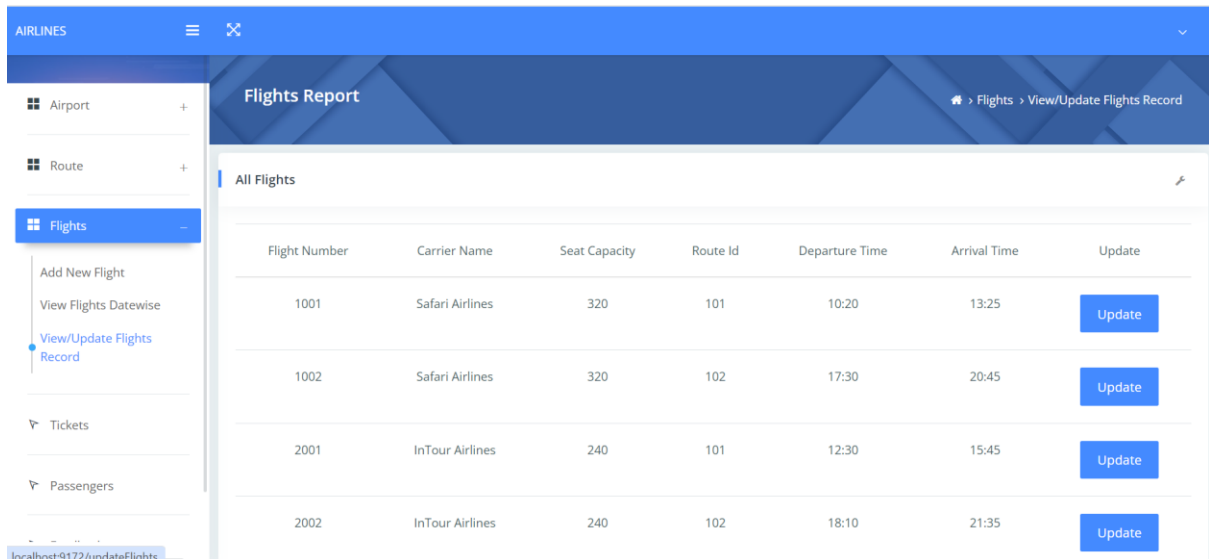
Page | 17



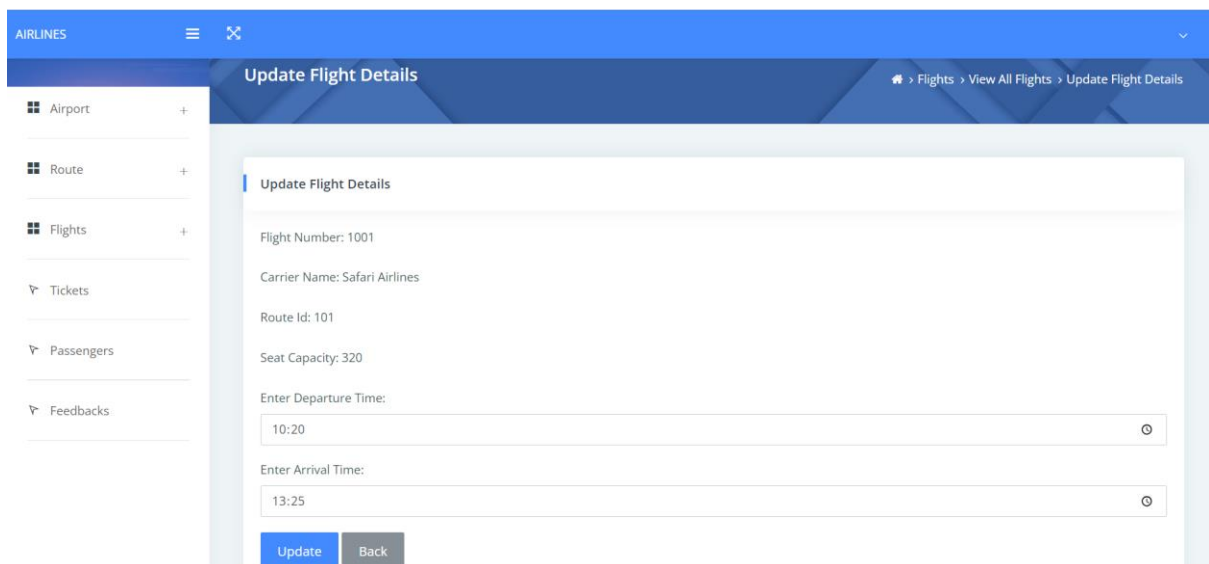
6.3.9 Search Flights by Date Page

Flight Number	Carrier Name	Seat Capacity	Route Id	Arrival Time	Departure Time	Seat Available
1001	Safari Airlines	320	101	13:25	10:20	318
1002	Safari Airlines	320	102	20:45	17:30	320
2001	InTour Airlines	240	101	15:45	12:30	240
2002	InTour Airlines	240	102	21:35	18:10	240
3001	Star Airlines	320	104	16:40	12:50	320

6.3.10 View Flights Datewise Page



6.3.11 View All Flights Page



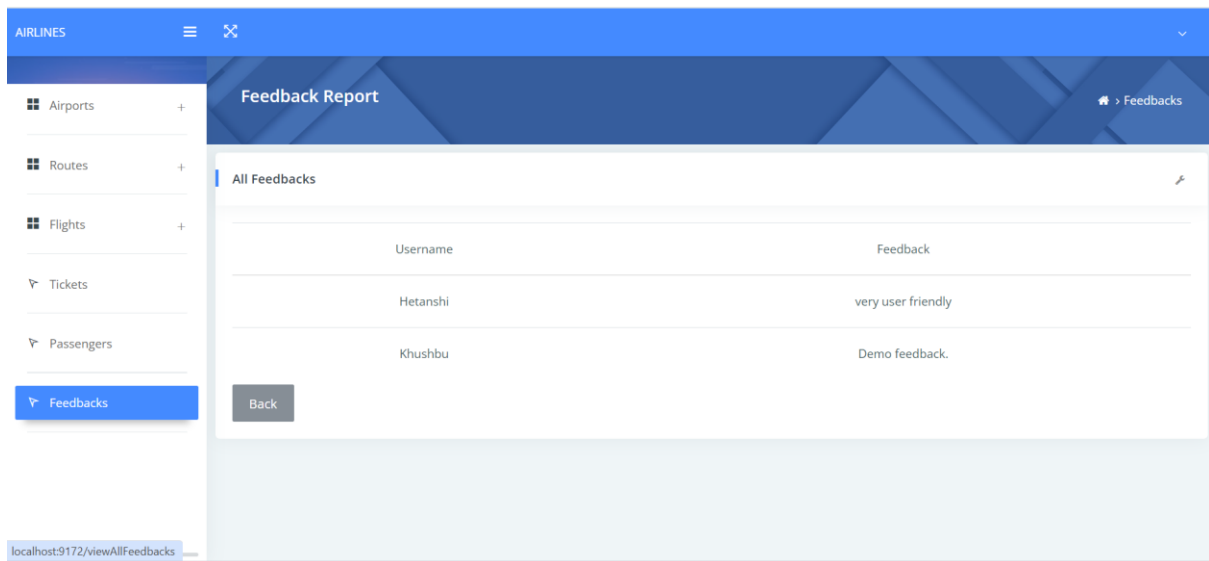
6.3.12 Update Flight Details Page

AIRLINES						
Ticket Report						
All Tickets						
Ticket Number	Flight Number	Carrier Name	Route Id	Date	Total Amount	
1000001	1001	Safari Airlines	101	2024-07-28	10000.0	
Back						

6.3.13 Ticket Details Page

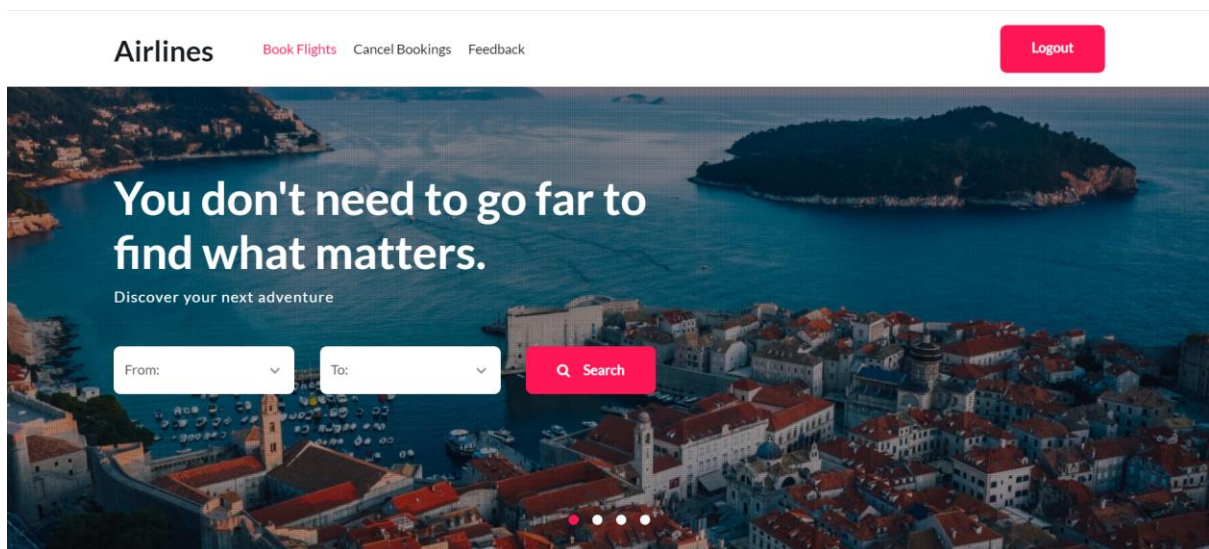
AIRLINES					
Passenger Report					
All Passengers					
Passenger Name	Passenger DOB	Ticket Number	Serial Number	Fare	
Keval Shah	2000-06-01	1000001	1	5000.0	
Dhruvil Joshi	2001-08-12	1000001	2	5000.0	
Back					

6.3.14 Passenger Details Page



6.3.14 Feedback Report Page

6.4 Customer Features:



6.4.1 Home Page

Available Flights

[Home](#) → [Available Flights](#)

Available Flights from AHMEDABAD to BANGALORE

Safari Airlines

10:20

Duration: 3h 05m

13:25

Flight No: 1001

Departure: AHMEDABAD

Arrival: BANGALORE

Rs. 5000.0

Book

InTour Airlines

12:30

Duration: 3h 15m

15:45

Flight No: 1002

Rs. 5000.0

Book

6.4.2 Available Flights Page

Book Flight

[Home](#) → [Available Flights](#) → [Book Flight](#)

Book Your Flight

Ticket Number: 1000002

Fare: Rs 5,000.00

Airlines Name: Safari Airlines

From: AHMEDABAD

Flight Number: 1001

To: BANGALORE

Journey Date:

Enter Passenger Details

Name: <input type="text" value="--"/>	Date of Birth: <input type="text" value="dd-mm-yyyy"/>
Name: <input type="text" value="--"/>	Date of Birth: <input type="text" value="dd-mm-yyyy"/>
Name: <input type="text" value="--"/>	Date of Birth: <input type="text" value="dd-mm-yyyy"/>
Name: <input type="text" value="--"/>	Date of Birth: <input type="text" value="dd-mm-yyyy"/>
Name: <input type="text" value="--"/>	Date of Birth: <input type="text" value="dd-mm-yyyy"/>
Name: <input type="text" value="--"/>	Date of Birth: <input type="text" value="dd-mm-yyyy"/>

[Back to Flights](#)[Submit](#)

6.4.3 Book Flight Page

Airlines

[Book Flights](#) [Cancel Bookings](#) [Feedback](#)

Logout

Confirmation

[Home](#) → [Available Flights](#) → [Book Flight](#) → Ticket Confirmation

Ticket Confirmation

Ticket No:	1000002	Airlines Name:	Safari Airlines	Flight Number:	1001
From:	AHMEDABAD	To:	BANGALORE	Date:	2024-08-04

Passenger Information

Passenger Name	Date Of Birth	Fare
Darshita Patel	2014-12-19	2500.0
Khushbu Patel	1950-09-06	3500.0
Total		6000.0

Back to Flights

Proceed to Pay

6.4.4 Booking Confirmation Page

Airlines

[Book Flights](#) [Cancel Bookings](#) [Feedback](#)

Logout

Cancel Bookings

[Home](#) → Ticket Cancellation

Enter Ticket Number:

Back

Search

6.4.5 Cancel Booking Page

Airlines

[Book Flights](#)[Cancel Bookings](#)[Feedback](#)

Logout

Confirmation

[Home](#) → [Ticket Cancellation](#) → [Confirm Cancellation](#)

Ticket

Ticket No:	1000002	Airlines Name:	Safari Airlines	Flight Number:	1001
From:	AHMEDABAD	To:	BANGALORE	Date:	2024-08-04

Passenger Information

Passenger Name	Date Of Birth	Fare
Darshita Patel	2014-12-19	2500.0
Khushbu Patel	1950-09-06	3500.0
Total Amount		6000.0

Back

Cancel Booking

6.4.6 Cancellation Confirmation Page

Airlines

[Book Flights](#)[Cancel Bookings](#)[Feedback](#)

Logout

Feedback

[Home](#) → [Feedback](#)

Enter Your Username:

Type Your Feedback/Remarks:

Type Your Message

Back

Submit

6.4.7 Feedback Page