



Konatham Naga Mukesh – 221CS132
Sreyas Lakkimsetti – 221CS134
Prayag Ganesh Prabhu – 221CS140
Tanay Praveen Shekokar – 221CS159

**The Best
Airline Ticket
Booking
System**

INDEX

S.No.	Topic	Page No.
1	Introduction	1
2	Literature Survey	1-3
3	Objectives	3-4
4	Features	4-5
5	Requirement Specification : Frontend & Backend	5
6	Database Schema Design	6
7	Entity Relationship Diagram	7
8	Normalization Process	8-9
9	Tables	9-13
10	Implementation	13-16
11	Conclusion	17
12	References	17-18
13	Appendix	18-21

Airwise

Konatham Naga Mukesh, Lakkimsetti Sreyas,
Prayag Ganesh Prabhu, Tanay Praveen Shekokar
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore, India
8639550996, 9611275188, 9353997270, 7022420056
konathamnagamukesh.221cs132@nitk.edu.in,
mrlakkimsettisreyas.221cs134@nitk.edu.in,
prayagganeshprabhu.221cs140@nitk.edu.in
tanayshekokar.221cs159@nitk.edu.in

March 30, 2024

A. Introduction

An airline ticket management system effectively handles reservations, cancellations, and modifications by streamlining the ticketing process. It optimises seat allocation, lowers manual errors, improves customer service, and makes revenue management easier. It also improves airline operational efficiency and allows real-time tracking of flight availability.

Furthermore, an airline ticket management system offers important insights via analytics, allowing airlines to make data-driven decisions. [1] It improves overall operational synergy by easily integrating with other airline systems, including booking platforms, payment gateways, and loyalty programs. In the end, it enhances the general traveller experience and increases airline revenue.

With the help of an efficient management system, any user can book their flight in a seamless manner. Additional features aiding the customer to help resolve their issues is an important factor to include in the reservation system.

The whole process should be very convenient for a user who is not that familiar with online interfaces. This made us build our airline ticket booking system on the above factors, by which we can make the user experience very comfortable.

An important feature of our system is the inclusion of a dual access/login option, where one is meant for the general users, while the other serves the admin. The admin includes all the professionals involved in updating the live data and monitoring the activity on the website. It includes periodic addition of details to the existing database, modifications if required, and new features which are updated onto the front end of the site. [2]

Thus our aim is to make the system as user friendly as possible, while incorporating all the necessary features at the same time.

B. Literature Survey

1) Existing Database Systems in use:

- Many airlines use the Oracle Database for their ticket reservation and management as it is easy to scale, is very reliable and has an extensive feature set.
- Both MySQL and PostgreSQL are open-source database systems in use due to their cost-effectiveness and reliability. It is popular among the smaller carriers.
- IBM's DB2 is an enterprise-level DBMS that is commonly used in the industry.
- It provides various features such as advanced analytic capabilities, disaster recovery and high availability.

2) Industry Practices:

- To estimate demand, maximise income, and optimise pricing tactics, airlines employ advanced mathematical models.
- Various flight scheduling algorithms are employed that maximize aircraft utilization while minimizing cost and meeting the passenger demand.
- Network optimization models have been incorporated to design efficient route networks that connect the destinations with minimum cost.
- Customer Relationship Management (CRM) systems are used to manage customer interactions, track preferences and patterns, and identify opportunities for personalized services.
- Airlines make use of inventory management systems to optimize the allocation of seats, cargo space, and other flight resources.

3) User Authentication and Profile Management:

- In addition to the username and password, many airlines have implemented two-factor authentication as an extra security layer.
- Some companies have incorporated biometric authentication techniques, such as fingerprint and facial recognition, to verify the passenger identities.
- A Secure Socket Layer (SSL) encryption is used to secure sensitive information, such as login credentials and personal details.
- Open Authorization (OAuth) is protocol in use that allows users to grant third-party applications limited access to their account without sharing their login credentials.

4) Flight and Booking Management:

- Airlines analyse various factors such as aircraft availability, flight routes, demand, and regulatory requirements to choose their desired software system which optimizes the scheduling.
- Airlines offer multiple channels for customers to book flights, including their websites, mobile apps, travel agencies, and third-party booking platforms.
- Once a customer makes a booking, the airline allocates the seat based on factors such as passenger preference, seat availability, and seating configurations.
- Airlines manage the crew scheduling, aircraft maintenance, fuelling, catering and ground handling services.
- Customer support is present throughout the booking process to assist the passengers with inquiries, reservation changes, and special requests.

5) PNR Enquires:

- PNR is a unique identifier associated with a passenger's booking.
- Passengers can check their PNR status, which includes details such as flight itinerary, seat assignments, ticketing information, and any special requests or services. [\[2\]](#)
- Passengers can retrieve their PNR information by entering their booking reference number and last name through the airline channels.

6) Payment Integration:

- Payment gateways such as Stripe, UPI, PayPal and Paytm are used to securely process transactions.
- Airline companies make use of tokenization to enhance security and minimize data breach impacts.
- Payment Card Industry Data Security Standard (PCI DSS) compliance is crucial for handling payment card information securely. Airlines must adhere to PCI standards to protect cardholder data during storage, processing, and transmission.
- Airlines employ fraud detection systems that analyse transaction patterns and detect potentially fraudulent activities in real-time. These systems use machine learning algorithms to identify suspicious behaviours and flag transactions for further review.

7) Security and Compliance:

- Access control mechanisms are employed to ensure that only authorized personnel can access sensitive information within the system.
- Firewalls are deployed to monitor and control incoming and outgoing network traffic, while IDS/IPS systems detect and prevent malicious activities such as unauthorized access or denial-of-service attacks.
- Data Loss Prevention (DLP) solutions are implemented to prevent unauthorized transmission or leakage of sensitive data. This includes monitoring and blocking the transfer of sensitive information outside of authorized channels.
- Flight ticket management systems adhere to industry standards and regulations such as the Payment Card Industry Data Security Standard (PCI DSS) and the General Data Protection Regulation (GDPR).

8) Performance and Scalability:

- In order to prevent any one server from becoming overloaded, load balancing divides incoming traffic across several servers. This facilitates better reaction times and resource optimization.
- To address growing demand, airlines frequently use horizontal scaling, which involves expanding their infrastructure with additional servers. With this method, capacity may be added smoothly as traffic increases.
- By momentarily stopping requests to a malfunctioning service, circuit breakers can be used to assist prevent cascading failures. This prevents the system from entirely failing in the event of a service interruption or heavy demand, allowing it to gracefully decline functionality.
- Content Delivery Networks (CDNs) cache static assets such as images, stylesheets, and scripts at edge locations closer to the user, reducing latency and offloading bandwidth from origin servers.
- Implementing rate limiting and throttling mechanisms helps in controlling the rate of incoming requests to prevent overload situations and ensure fair resource allocation among users.

C. Objectives

1. Database Selection and Implementation:

- Evaluate the requirements of the airline industry, considering factors such as scalability, reliability, and feature set, to select an appropriate database system.
- Implement MySQL as the preferred database system based on its cost-effectiveness, reliability, and suitability for airline operations.

2. Schema Design and Optimization:

- Design and implement a MySQL database schema tailored to the needs of airline operations, including tables for ticket reservation, flight scheduling, passenger profiles, payment records, and security information. [3]
- Optimize the database schema for efficient data retrieval and manipulation, ensuring proper indexing, normalization, and data integrity constraints.

3. Data Integration and Migration:

- Integrate existing data from disparate sources, such as legacy systems or third-party applications, into the MySQL database system to ensure a centralized and consistent data repository for airline operations. [3]
- Develop data migration strategies and scripts to transfer data from legacy systems to MySQL while maintaining data integrity and minimizing downtime.

4. Query Development and Optimization:

- Develop SQL queries and stored procedures to support various functionalities, including flight booking, passenger management, payment processing, and reporting. [2]
- Optimize query performance by analyzing execution plans, indexing appropriate columns, and tuning SQL statements to improve response time and resource utilization.

5. Security and Compliance:

- Implement robust security measures within the MySQL database system to safeguard sensitive information, such as passenger profiles, payment details, and operational data, from unauthorized access or data breaches.
- Ensure compliance with industry standards and regulations, such as PCI DSS and GDPR, by implementing access control mechanisms, encryption protocols, and audit trails.

6. Backup and Disaster Recovery:

- Establish backup and disaster recovery procedures for the MySQL database system to mitigate the risk of data loss or corruption in the event of hardware failures, natural disasters, or cyberattacks.
- Regularly backup database files and transaction logs, and implement strategies for data replication and failover to secondary servers or cloud storage for enhanced resilience.

7. Performance Monitoring and Scaling:

- Monitor database performance metrics, such as CPU usage, memory utilization, and disk I/O, to identify bottlenecks and optimize resource allocation for optimal performance.
- Implement scalability measures, such as horizontal scaling with additional servers or vertical scaling with upgraded hardware, to accommodate growing data volumes and user traffic in airline operations.

D. Features 4

1. Dual Interface - Administrative and User Access (Web Portal):

This feature provides distinct interfaces tailored for administrators and regular users within the website's portal. The administrative interface enables administrators to manage flights, bookings, and user accounts, while the user interface facilitates flight search, booking, and profile management. 5 It ensures a segregated yet cohesive experience for users with different roles and permissions.

2. Flight Query System:

This system allows users to search for available flights based on various criteria such as date, destination, and departure time. It interfaces with the flight database to retrieve relevant flight information and present it to the user.

3. Reservation Management System:

This system handles the entire life cycle of flight reservations, including creation, modification, and cancellation. 4 It maintains a database of booked flights and manages seat allocations.

4. Seat Inventory Management:

This component keeps track of available seats on each flight and manages seat assignments based on user bookings. It ensures that seats are allocated efficiently and accurately.

5. Authentication and User Profiles Database:

This system manages user authentication and stores user profile information securely. It allows users to log in to the system securely and maintains their personal details for future bookings.

6. Flight Operations Control:

This system oversees various aspects of flight operations, including scheduling, crew assignments, and aircraft maintenance. It ensures the smooth and efficient operation of flights within regulatory and safety guidelines.

7. Booking Lifecycle Management:

This component governs the entire lifecycle of flight bookings, from initial reservation to ticket issuance and beyond. It tracks each booking through its various stages and updates relevant stakeholders accordingly.

8. Payment Gateway Integration:

This feature integrates with third-party payment gateways to facilitate secure online payments for flight bookings. 2 It encrypts sensitive payment information and ensures seamless transaction processing.

9. Notification Service:

This service sends out notifications to users at various stages of the booking process, including booking confirmation, itinerary updates, and payment reminders. 5 It helps keep users informed and engaged throughout

their journey.

10. Search Parameter Handling:

This component processes user input and translates it into search queries to retrieve relevant flight information. It validates user input, applies search filters, and retrieves matching results from the flight database.

11. Cancellation and Refund Processing:

This feature handles user requests for booking cancellations and initiates refund processing as per the airline's policies. It updates the booking status, releases reserved seats, and processes refund transactions.

12. Seat Class Upgradation Module:

This module allows users to upgrade their seat class after booking, subject to availability and fare differences. It manages seat inventory across different classes and facilitates seat upgrades seamlessly.

13. Passenger Name Record (PNR) Inquiry System:

This system enables users to inquire about their booking status and itinerary using their unique PNR code. 2 It retrieves booking details from the database and provides real-time updates to users.

E. Requirement Specification

1. Frontend:

- Hyper Text Markup Language (HTML)
- Cascading Style Sheets (CSS)
- JavaScript (JS)
- Bootstrap

2. Backend

- Node JS
- Express JS
- EJS
- MySQL
- JSON Web Tokens (JWT)

F. System Design

1. Database Schema Design

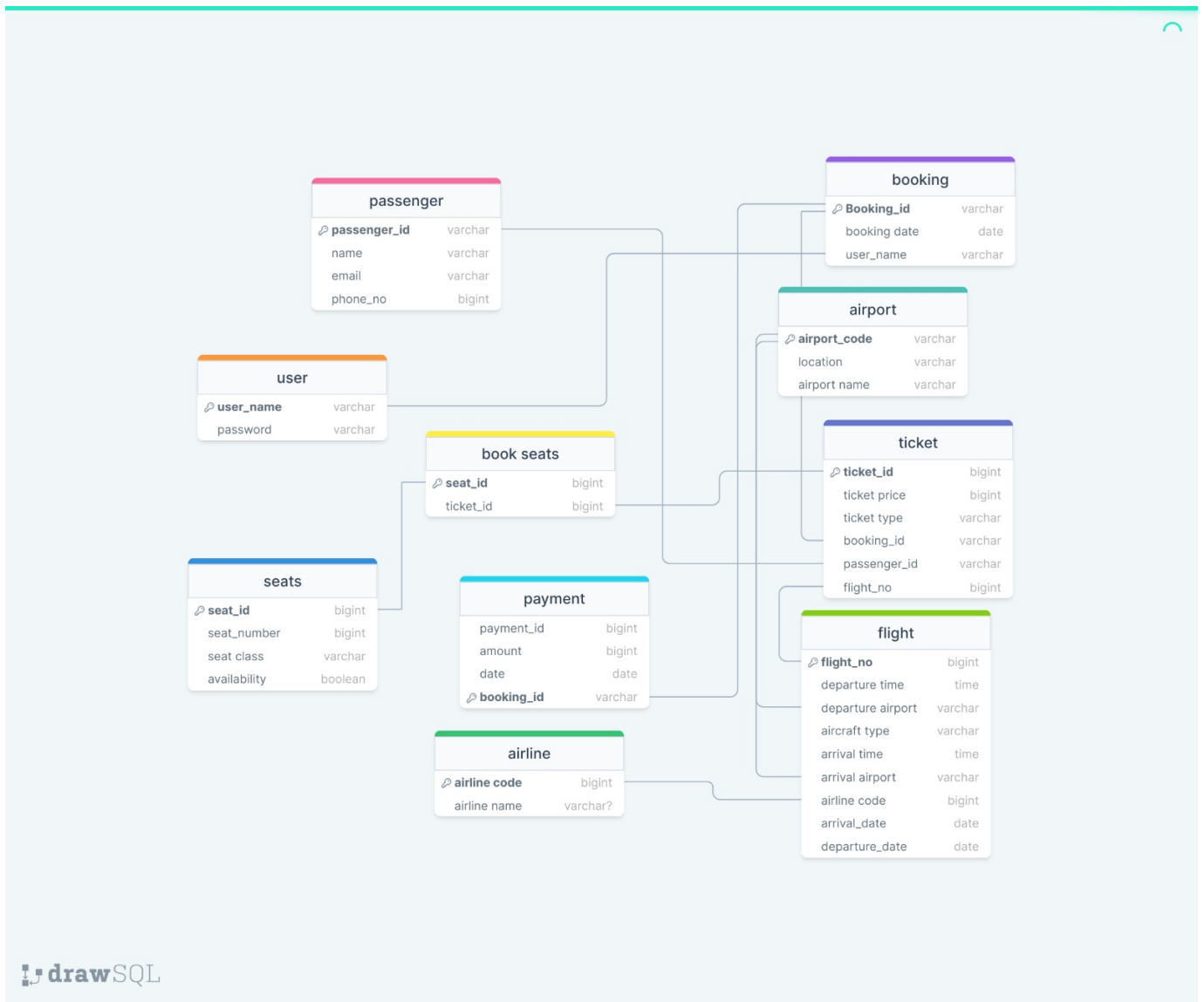


Figure 1: Flight Database Schema

2. Entity Relationship Diagram

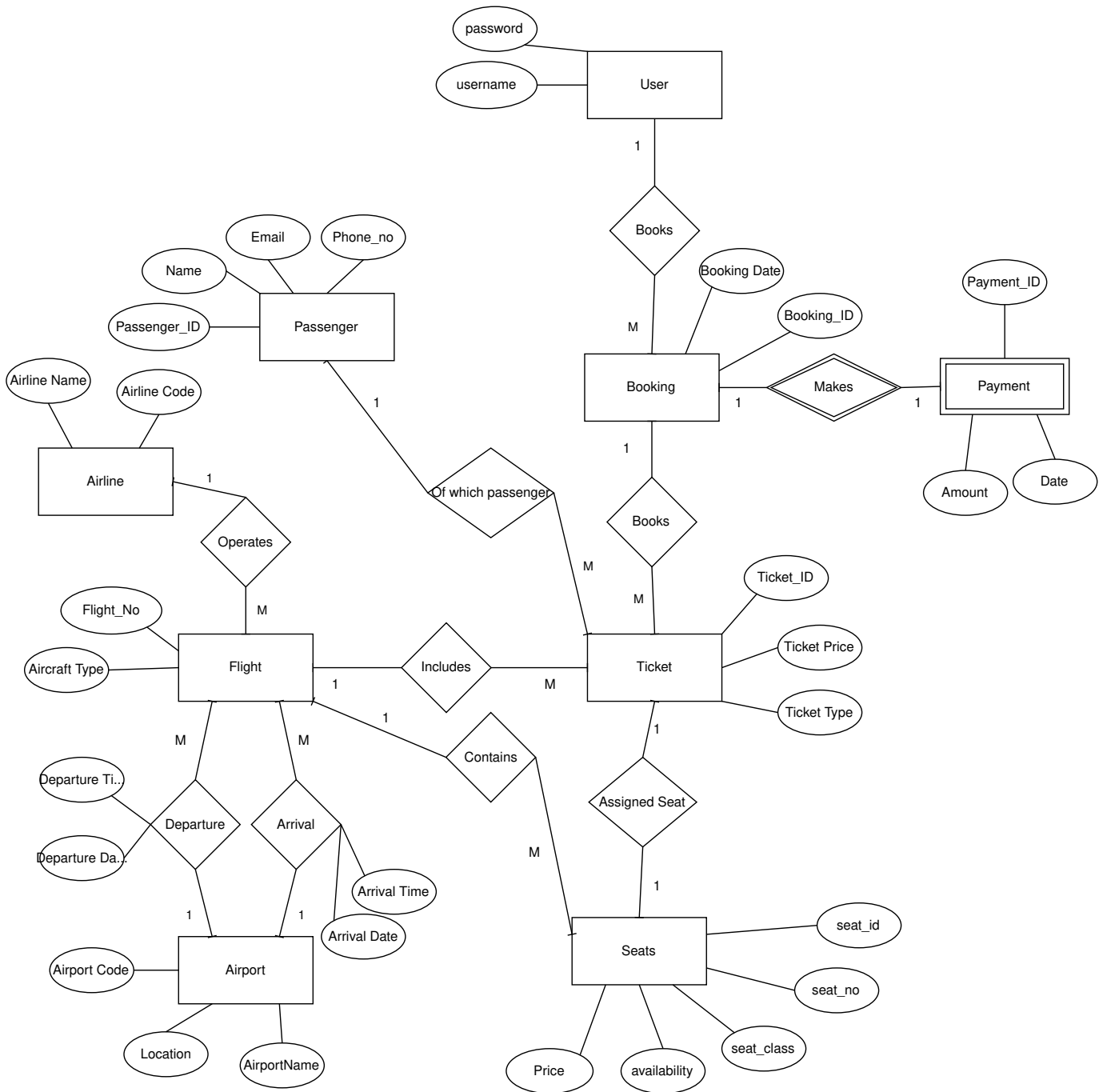


Figure 2: Flight ER Diagram

3. Normalization Process

- Normalization is the process of organizing data in a database. It includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency. [6]
- First Normal Form (1NF): This is the most basic level of normalization. In 1NF, each table cell should contain only a single value, and each column should have a unique name. The first normal form helps to eliminate duplicate data and simplify queries. [6] Let us consider the following example to explain 1NF:

Column	Sample Record
product name	Nike Air Max
usage	Casual
category	Men, Woman
manufacturer	Nike
color	Black, White, Green
price	125
size	US Mens 9
description	A casual shoe for everyday wear
number in stock	15

Figure 3: A Product table with various attributes and sample record

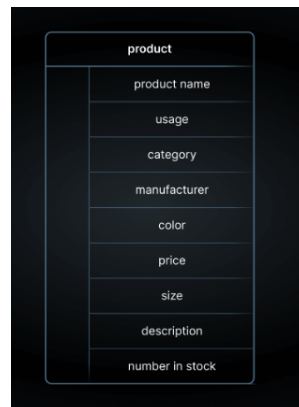


Figure 4: Visual Representation in ER Diagram

For a database design to be in first normal form, or 1NF, each row needs to have a primary key, and each field needs to contain a single value.

In our example, the Category column can contain several values (Men or Women or Kids), and the Color column contains several values (Black, White, Green, and many others).

We also do not have a primary key in the table. So, this design does not meet first normal form.

To adhere to the first normal form, we can add a new product ID column to the table, and move the Category and Color values to separate tables. This implies that our resulting database will look like:

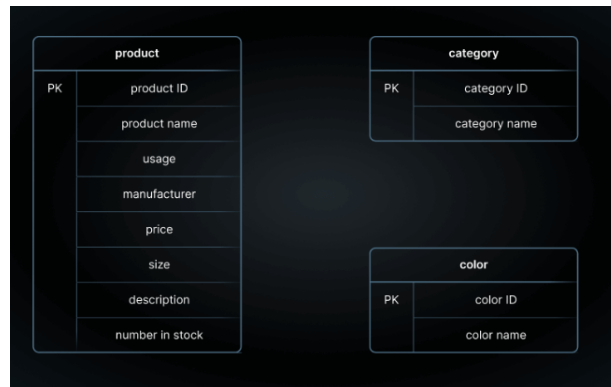


Figure 5: 1NF compliant design

- Second Normal Form (2NF) : The idea of fully functional dependency serves as the foundation for the second Normal Form (2NF). Relations with composite keys, or relations with a main key made up of two or more attributes, are subject to the second Normal Form. [6] A relation that has a primary key with only one attribute is inherently in at least two networks. Update anomalies may affect a relation that is not in 2NF. A relation must be in the first normal form and not have any partial dependencies in order to be in the second normal form. If a relation has No Partial Dependency, meaning that no non-prime attribute (i.e., attributes not included in any candidate key) depends on any appropriate subset of any candidate key in the table, then the relation is in 2NF. [6] For example if you see our flight and airline tables,

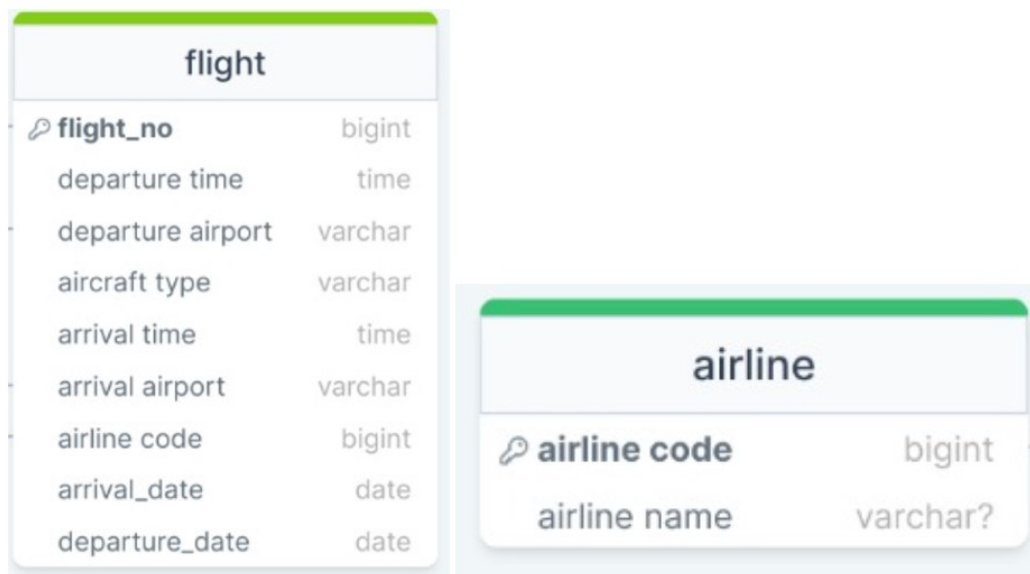


Figure 6: 2NF depiction in our database schema

If the airline name was included in the flight table itself then partial dependency would have occurred as airline name is dependent on airline code leading to multiple duplicate values. Hence we decided to create a new table called airline which will have the airline code and airline name as its attributes and not include airline name in the flight table. We have followed the same steps while creating all the other entities in our database. In this way we have ensured that our schema design satisfies 2NF.

4. Tables

The following are the tables used in our Airline Ticket Reservation Database:

- Passenger :

```
create table Passenger(
passenger_id varchar(50) primary key,
name varchar(100),
email varchar(100),
phone_no bigint);
```

Field	Type	Null	Key	Default	Extra
passenger_id	varchar(50)	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
email	varchar(100)	YES		NULL	
phone_no	bigint	YES		NULL	

- User :

```
create table User(
user_name varchar(100) primary key,
password varchar(100));
```

Field	Type	Null	Key	Default	Extra
user_name	varchar(100)	NO	PRI	NULL	
password	varchar(100)	YES		NULL	

- Payment :

```
create table Payment(
payment_id bigint,
amount bigint,
date date,
booking_id varchar(100) primary key);
```

Field	Type	Null	Key	Default	Extra
payment_id	bigint	YES		NULL	
amount	bigint	YES		NULL	
date	date	YES		NULL	
booking_id	varchar(100)	NO	PRI	NULL	

- Seats :

```
create table seats(
seat_id bigint primary key,
seat_number bigint,
seat_class varchar(100),
availability boolean);
```

Field	Type	Null	Key	Default	Extra
seat_id	bigint	NO	PRI	NULL	
seat_number	bigint	YES		NULL	
seat_class	varchar(100)	YES		NULL	
availability	tinyint(1)	YES		NULL	

- Booking :

```
create table booking(
  booking_id varchar(100) primary key,
  booking_date date,
  user_name varchar(100),
  foreign key(user_name) references User(user_name));
```

Field	Type	Null	Key	Default	Extra
booking_id	varchar(100)	NO	PRI	NULL	
booking_date	date	YES		NULL	
user_name	varchar(100)	YES	MUL	NULL	

- Airport :

```
create table airport(
  airport_code varchar(100) primary key,
  location varchar(100),
  airport_name varchar(50));
```

Field	Type	Null	Key	Default	Extra
airport_code	varchar(100)	NO	PRI	NULL	
location	varchar(100)	YES		NULL	
airport_name	varchar(50)	YES		NULL	

- Airline :

```
create table airline(
  airline_code bigint primary key,
  airline_name varchar(100));
```

Field	Type	Null	Key	Default	Extra
airline_code	bigint	NO	PRI	NULL	
airline_name	varchar(100)	YES		NULL	

- Flight :

```
create table flight(
flight_no bigint primary key,
departure_time time,
departure_airport varchar(50),
aircraft_type varchar(50),
arrival_time time,
arrival_airport varchar(50),
airline_code bigint,
arrival_date date,
departure_date date,
foreign key(arrival_airport) references airport(airport_code),
foreign key(departure_airport) references airport(airport_code),
foreign key(airline_code) references airline(airline_code));
```

Field	Type	Null	Key	Default	Extra
flight_no	bigint	NO	PRI	NULL	
departure_time	time	YES		NULL	
departure_airport	varchar(50)	YES	MUL	NULL	
aircraft_type	varchar(50)	YES		NULL	
arrival_time	time	YES		NULL	
arrival_airport	varchar(50)	YES	MUL	NULL	
airline_code	bigint	YES	MUL	NULL	
arrival_date	date	YES		NULL	
departure_date	date	YES		NULL	

- Ticket :

```
create table ticket(
ticket_id bigint primary key,
ticket_price bigint,
ticket_type varchar(50),
booking_id varchar(100),
passenger_id varchar(50),
flight_no bigint,
foreign key(booking_id) references booking(booking_id),
foreign key(flight_no) references flight(flight_no),
foreign key(passenger_id) references Passenger(passenger_id));
desc ticket;
```

Field	Type	Null	Key	Default	Extra
ticket_id	bigint	NO	PRI	NULL	
ticket_price	bigint	YES		NULL	
ticket_type	varchar(50)	YES		NULL	
booking_id	varchar(100)	YES	MUL	NULL	
passenger_id	varchar(50)	YES	MUL	NULL	
flight_no	bigint	YES	MUL	NULL	

- Booked Seats :


```
create table book_seats(
seat_id bigint primary key,
ticket_id bigint);
```

Field	Type	Null	Key	Default	Extra
seat_id	bigint	NO	PRI	NULL	
ticket_id	bigint	YES		NULL	

G. Implementation

1. Frontend

For the frontend of our airline ticket reservation website, we incorporated various tools such as HTML,CSS,JavaScript and Bootstrap. A brief description with respect to all of them is as follows:

- Hyper Text Markup Language (HTML):
HTML is the standard markup language for creating web pages and web applications. It provides the structure of the webpage by using a system of elements and tags to define the different parts of a document, such as headings, paragraphs, images, links, forms, etc. [7] With HTML, you can define the content and layout of the webpage.
- Cascading Style Sheets (CSS) :
CSS is used to style the HTML elements, controlling their appearance on the webpage. [7] It allows developers to define styles like colors, fonts, spacing, layout, and more. CSS helps in creating visually appealing and consistent designs across the website.
- JavaScript (JS) :
JavaScript is a high-level programming language that adds interactivity and dynamic behavior to web pages. [8] It enables developers to manipulate the HTML and CSS elements in real-time, respond to user actions, and dynamically update the content without reloading the entire page. JavaScript can be used for tasks such as form validation, animation, handling events, making AJAX requests to fetch data from servers, etc.
- Bootstrap:
Bootstrap is a popular front-end framework for developing responsive and mobile-first websites and web applications. It provides a collection of pre-designed HTML, CSS, and JavaScript components, as well as a grid system and utilities, to streamline the process of building modern and visually appealing user interfaces. [9]

- Now let us analyze why the above tools are suitable for Airwise :
Structure and Content (HTML):

HTML is perfect for defining the structure and content of the webpage. It allows you to organize elements such as flight listings, booking forms, navigation menus, and more. You can use HTML to create semantic markup, making the webpage accessible and easily understandable by search engines and screen readers. [7]

Visual Styling (CSS):

CSS enables you to style the webpage according to the branding and design requirements of Airwise. You can use CSS to create visually appealing layouts, define colors, typography, margins, borders, and other visual elements to enhance the user experience.

Interactivity and Functionality (JavaScript):

JavaScript is essential for adding interactivity and dynamic functionality to the Airwise website. With JavaScript, you can implement features such as interactive forms for flight searches and bookings, real-time updates on available flights and prices, validation of user input, and handling of user interactions like clicks and swipes. [8] JavaScript also enables integration with APIs provided by airlines or third-party services for fetching real-time flight information, prices, and availability.

Responsive Design and Customization (Bootstrap) :

Bootstrap is built with responsive design principles in mind, meaning that websites and applications developed using Bootstrap automatically adjust their layout and appearance to different screen sizes

and devices, such as desktops, tablets, and smartphones. [9] It provides extensive customization options through Sass variables and mixins, allowing developers to tailor the framework to meet the specific design and branding requirements of their projects.

2. Backend

For the backend side of the airline ticket reservation website, we have implemented various frameworks such as NodeJS, ExpressJS, EJS, JSON Web Tokens (JWT) and for the database system we have used MySQL. A brief description with respect to all of them is given as follows:

- **NodeJS**
Node.js is a runtime environment that allows you to run JavaScript code outside of the browser, typically on the server-side. [10]: It uses an event-driven, non-blocking I/O model, making it lightweight and efficient for building real-time, data-intensive applications. It enables developers to use JavaScript for both client-side and server-side development, resulting in a more cohesive development experience. [10]
- **ExpressJS:**
Express.js is a minimalist web application framework for Node.js, designed for building web applications and APIs. It provides a robust set of features for defining routes, handling HTTP requests and responses, managing middleware, and organizing application logic. It simplifies the process of building server-side logic and handling complex routing scenarios, making it a popular choice for building web servers and RESTful APIs in Node.js applications [10].
- **Extended JavaScript (EJS) :**
EJS, or Embedded JavaScript, is a simple templating language that lets you generate HTML markup with plain JavaScript. It allows you to embed JavaScript code directly into your HTML templates, making it easy to dynamically generate content based on data from your server or application.
- **JSON Web Tokens (JWT) :**
JWT, or JSON Web Tokens, is a compact, URL-safe means of representing claims to be transferred between two parties securely. [11] It is commonly used for authentication and authorization purposes in web applications.
JSON-Based: JWT is based on JSON (JavaScript Object Notation), making it easy to parse and generate.
Compact: JWTs are compact, which means they can be sent as URL parameters, placed in HTTP headers, or even within the body of an HTTP request.
Self-Contained: JWTs contain all the necessary information about the user or session within the token itself, reducing the need to query a database for user information.
Cryptographically Signed: JWTs can be digitally signed using a secret key or a public/private key pair, ensuring that the token has not been tampered with during transmission. [11]
- **MySQL :**
MySQL is an open-source relational database management system (RDBMS) that is widely used for storing and managing structured data. [12] It provides a powerful set of features for creating, querying, and manipulating databases, including support for transactions, indexes, views, and stored procedures. MySQL is known for its reliability, scalability, and performance, making it suitable for web applications that require efficient data storage and retrieval.
- Now let us see why the above frameworks are suitable for Airwise :
Real-time Updates and Interactivity:

Node.js enables real-time communication between the client and server through techniques like WebSockets and server-sent events (SSE) [10]. This allows Airwise to provide real-time updates on flight availability, prices, and bookings without the need for page refreshes. Express.js simplifies the implementation of real-time features by providing a flexible and extensible middleware architecture.

Data Management and Reliability:

MySQL offers a reliable and robust database solution for storing critical information such as user profiles, flight details, bookings, and transactions. Its support for ACID (Atomicity, Consistency, Isolation, Durability) properties ensures data integrity and consistency, which are essential for a flight ticket booking application like Airwise.

Community Support and Ecosystem:

Node.js, Express.js, and MySQL have large and active developer communities with extensive documentation, tutorials, and libraries available. This vast ecosystem provides Airwise developers with access to a wealth of resources and tools for building, deploying, and maintaining the application effectively.

3. Code Snippets

The following are some of the code snippets involved in our code.

```
JS search.js

const pool = mysql.createPool({
  connectionLimit : 10,
  host             : process.env.MY_SQL_HOST,
  user             : process.env.MY_SQL_USER,
  password         : process.env.MY_SQL_PASSWORD,
  database         : process.env.MY_SQL_DATABASE
});

router.post('/', encoder, async (req, res) => {
  const departure_date = req.body.departure_date;
  const Departure_City = req.body.Departure_City;
  const Destination_City = req.body.Destination_City;

  pool.query('SELECT * FROM flight F JOIN airport Dep ON F.departure = Dep.airport_code
  JOIN airport Ar ON F.destination = Ar.airport_code WHERE F.departure_date = ? AND Dep.location = ? AND Ar.location = ?',
  [departure_date, Departure_City, Destination_City],
  (error, result, fields) => {
    if (error) {
      console.error('Error fetching data:', error);
      res.status(500).json({ error: 'Error fetching data' });
    } else if (result.length === 0) {
      res.status(404).send("Flight not found");
    } else {
      res.render('searchResult', { flights: result });
    }
  })
})

router.post('/selectedFlight', (req, res) => {
  const selectedFlightID = req.body.selectedFlight;

  pool.query('SELECT * FROM flight WHERE flightID = ?', [selectedFlightID], (error, result, fields) => {
    if (error) {
      console.error('Error fetching data:', error);
      res.status(500).json({ error: 'Error fetching data' });
    } else if (result.length === 0) {
      res.status(404).send("Selected flight not found");
    } else {
      const flight = result[0];
      res.render('selectedFlight', { flight: flight });
    }
  })
})
})
```

Figure 7: This code enables a user to search for an available flight

```
JS seat_selection.js

const pool = mysql.createPool({
  connectionLimit : 10,
  host             : process.env.MY_SQL_HOST,
  user             : process.env.MY_SQL_USER,
  password         : process.env.MY_SQL_PASSWORD,
  database         : process.env.MY_SQL_DATABASE
});

router.post('/', (req, res) => {
  const selectedFlightID = req.body.flightID;

  const seatInfo = {
    flightID: selectedFlightID,
    // Add other seat information here
  };

  console.log(req.body);

  pool.query('SELECT * FROM flight WHERE flightID = ?', [selectedFlightID], (error, result, fields) => {
    if (error) {
      console.error('Error fetching data:', error);
      res.status(500).json({ error: 'Error fetching data' });
    } else if (result.length === 0) {
      console.log(selectedFlightID);
      res.status(404).send("Selected flight not found");
    } else {
      const flight = result[0];
      console.log(flight);
      res.render('seatSelection', { seatInfo: flight });
    }
  })
})
```

Figure 8: The code decides the seat for the customer

```

JS admin.js

const pool = mysql.createPool({
  connectionLimit : 10,
  host            : process.env.MY_SQL_HOST,
  user            : process.env.MY_SQL_USER,
  password        : process.env.MY_SQL_PASSWORD,
  database        : process.env.MY_SQL_DATABASE
});

router.post('/', encoder, async (req, res) => {
  const username = req.body.username;
  const password = req.body.password;
  const role = 1;
  console.log(req.body);
  console.log("username:", username);
  console.log(password);
  pool.query('SELECT * from user where user_name=? and admin=?', [username, role], (error, result, fields) => {
    if (error) console.log(error);
    if (result[0]) {
      if (result[0].password === password) {
        const token = jwt.sign({ username: username }, process.env.JWT_SECRET);
        pool.query('SELECT * FROM flight', (error, results) => {
          if (error) {
            console.error('Error fetching flights:', error);
            res.status(500).send('Error fetching flights');
          } else {
            res.render('admin/home', { flights: results });
          }
        });
      } else {
        res.send('Login failed! incorrect password')
      }
    } else {
      res.send("admin not found!")
    }
  });
});

```

Figure 9: The admin can access the database with the help of this code

```

JS flight_and_user.js

router.post('/add-flight', (req, res) => {
  const { flightID, departure, destination, departureTime, arrivalTime } = req.body;
  pool.query('INSERT INTO flight (flightID, departure, destination, departureTime, arrivalTime) VALUES (?, ?, ?, ?, ?)',
    [flightID, departure, destination, departureTime, arrivalTime],
    (error, results) => {
      if (error) {
        console.error('Error adding flight:', error);
        res.status(500).send('Error adding flight');
      } else {
        res.redirect('/admin');
      }
    });
});

router.post('/update-flight', (req, res) => {
  const { flightID, departure, destination, departureTime, arrivalTime } = req.body;
  pool.query('UPDATE flight SET departure = ?, destination = ?, departureTime = ?, arrivalTime = ? WHERE flightID = ?',
    [departure, destination, departureTime, arrivalTime, flightID],
    (error, results) => {
      if (error) {
        console.error('Error updating flight:', error);
        res.status(500).send('Error updating flight');
      } else {
        res.redirect('/admin');
      }
    });
});

router.post('/remove-flight', (req, res) => {
  const { flightID } = req.body;
  pool.query('DELETE FROM flight WHERE flightID = ?', [flightID], (error, results) => {
    if (error) {
      console.error('Error removing flight:', error);
      res.status(500).send('Error removing flight');
    } else {
      res.redirect('/admin');
    }
  });
});

router.post('/users', (req, res) => {
  pool.query('SELECT * FROM user', (error, results) => {
    if (error) {
      console.error('Error fetching flights:', error);
      res.status(500).send('Error fetching flights');
    } else {
      res.render('admin/users', { users: results });
    }
  });
});

```

Figure 10: This code enables the admin to make changes in the flight data and view the user access

H. Conclusion

1. Summary

In this project our aim was to design an airline ticket management system, in order to create a hassle free user experience for anyone who wants to book a flight. Our website Airwise is designed to enhance the efficiency of reservations, modifications and cancellations while minimizing errors at the same time. Our main feature, which is our dual login mode enables both the user as well as the admin to access the website. The user can book a given flight of their choice, while the admin can make the necessary changes in the database to address issues or logistical changes that might occur in the flight schedules. We prioritize our customer's convenience through our seamless booking process and our user friendly interface. Our dual access feature facilitates live data updates and website monitoring capability. In conclusion, our project aims for user friendly functionality while incorporating the essential features of an efficient airline ticket management system.

2. Achievement of Project Objectives

With the main focus on providing a seamless user experience, our primary features include dual login functionality for administrators and users, distinct pages for login, home, seat booking, and payment, all powered by MySQL, Node.js, HTML, JSON, JavaScript, CSS, and Bootstrap. From the outset, our goal was clear: to develop a platform that caters to the diverse needs of both administrators and users while ensuring security, reliability, and efficiency. Implementing dual login functionality was a cornerstone of our project. Leveraging Node.js, we engineered secure authentication mechanisms that grant access to distinct portals for administrators and users. This ensures that each user experiences tailored functionalities and privileges according to their role. The user interface, crafted with HTML, CSS, and Bootstrap, embodies our commitment to intuitive design and seamless navigation. Each page, from the login interface to the home, seat booking, and payment pages, has been meticulously designed to enhance user engagement and streamline the booking process. The seat booking feature, powered by MySQL, serves as the backbone of our platform. We developed a robust database schema to manage flight details, seat availability, and user bookings, ensuring accuracy and reliability at every step. In addition to user-facing functionalities, we prioritized the development of an intuitive administrative interface. Admins have access to specialized tools and dashboards, empowering them to manage flights, view booking data, and oversee system operations efficiently. This is how all our project objectives were achieved.

3. Lessons Learned

This project has taught us all some important lessons regarding the design of a good reservation system. A developer should always keep the user's interests in mind while designing a website and ensure that it is user friendly and has smooth access overall. There are many aspects of our website which need to be kept in mind regularly. The database management system requires regular updates, modifications and additions. The admin access calls for periodic maintenance of the overall database. Prioritizing user experience is extremely crucial in maintaining the reliability of the site. We can incorporate more features in the future that enhance the user comfort and ensure a smooth booking process. We can implement robust security measures in order to safeguard user information and prevent unauthorized access or data breaches. We also need modify our system to accommodate future growth and handle higher volumes of transactions. The design process of Airwise has given each one of us a better understanding of the challenges faced by airline reservation sites and the features and solutions that they have incorporated. Nonetheless, we have successfully managed to design our website, ensuring efficiency, user friendly interface, good transactional ability and a smooth user experience.

References

- [1] X. L. F. L. Zhao Wei, Rongsheng Dong, Model Checking Airline Tickets Reservation System Based on BPEL, <https://ieeexplore.ieee.org/document/5402900/> (2009).
- [2] Flight Booking Process: Structure, Steps, and Key Systems, <https://www.altexsoft.com/blog/flight-booking-process-structure-steps-and-key-systems/>, [Accessed: 01-03-2024] (2021).
- [3] How to Design Database for Flight Reservation System, <https://www.geeksforgeeks.org/how-to-design-database-for-flight-reservation-system/>, [Accessed: 25-02-2024] (2019).
- [4] Features of flight reservation system, <https://www.otrams.com/features-flight-reservation-system/>, [Accessed: 21-2-2024] (2020).
- [5] K. J.Marrenbach, M.Pauly, Design of a User Interface for a Future Flight Management System, <https://www.sciencedirect.com/science/article/pii/S1474667017401108>, [Accessed: 04-03-2024] (1998).
- [6] Normalization of databases, <https://www.geeksforgeeks.org/normal-forms-in-dbms/>.

- [7] HTML MDN Docs, <https://developer.mozilla.org/en-US/docs/Web/HTML>.
- [8] JavaScript MDN Docs, <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- [9] Bootstrap Documentation, <https://getbootstrap.com/>.
- [10] Node Js Documentation, <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>.
- [11] JWT Documentation, <https://jwt.io/>.
- [12] MYSQL, <https://en.wikipedia.org/wiki/MySQL>.

Appendix



Figure 11: Home Page of our Airwise Website

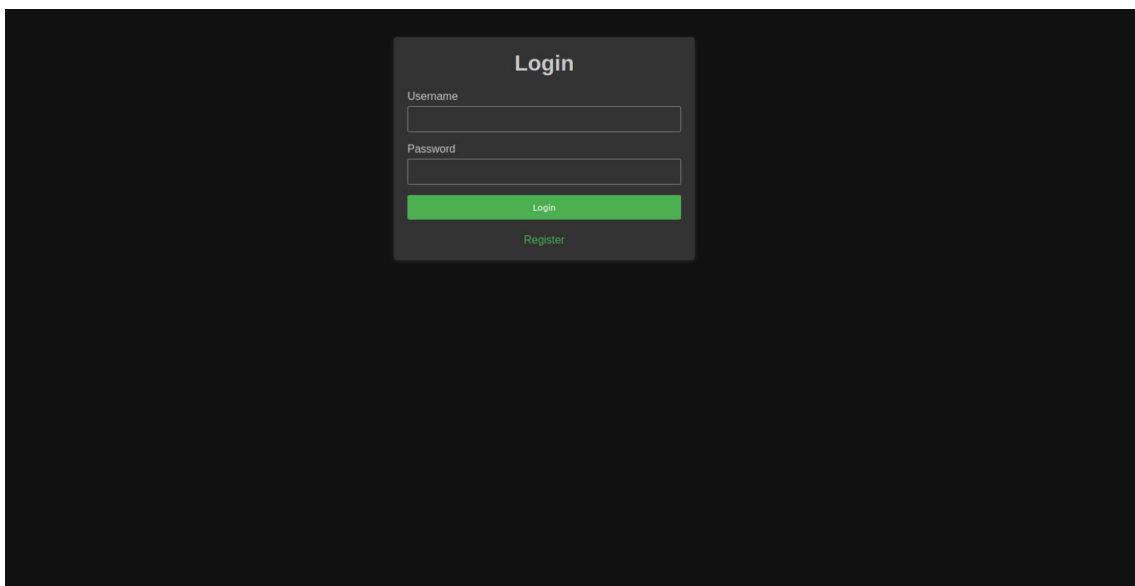


Figure 12: User and Admin Login page

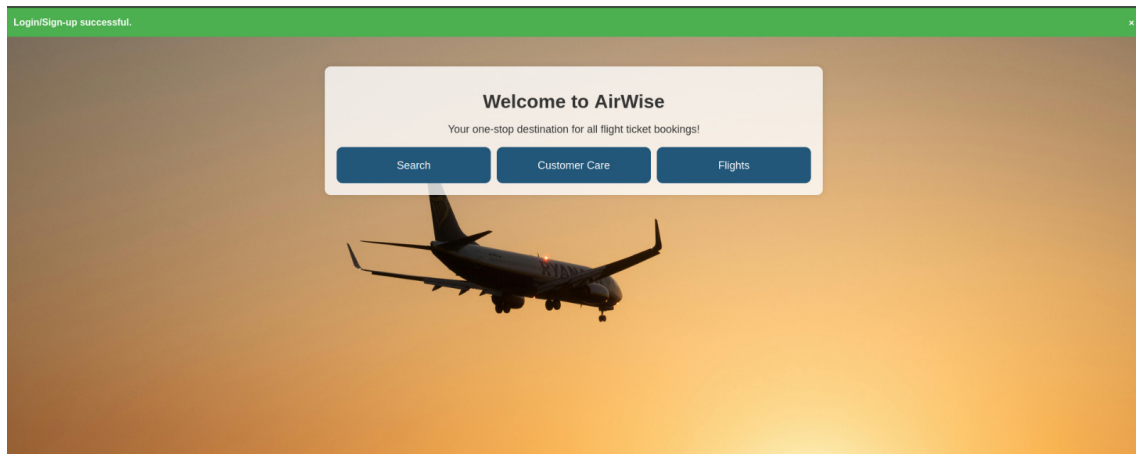


Figure 13: User page post Login

Admin Page

Add New Flight

Flight ID	Departure	Destination	Departure Time
Arrival Time	<button>Add Flight</button>		

Update Flight

Flight ID	Departure	Destination	Departure Time
Arrival Time	<button>Update Flight</button>		

Remove Flight

Flight ID to remove	<button>Remove Flight</button>
---------------------	--------------------------------

Available Flights

Flight ID	Departure	Destination	Departure Time	Arrival Time
101	BOM	DEL	08:00:00	10:30:00
111	BOM	DEL	09:00:00	11:30:00
112	BOM	DEL	11:30:00	14:00:00
113	BOM	DEL	12:00:00	14:30:00
114	BOM	DEL	14:00:00	16:30:00
115	BOM	DEL	15:30:00	18:00:00
116	DEL	BOM	10:00:00	12:30:00
117	DEL	BOM	13:00:00	15:30:00

Figure 14: Admin page post Login

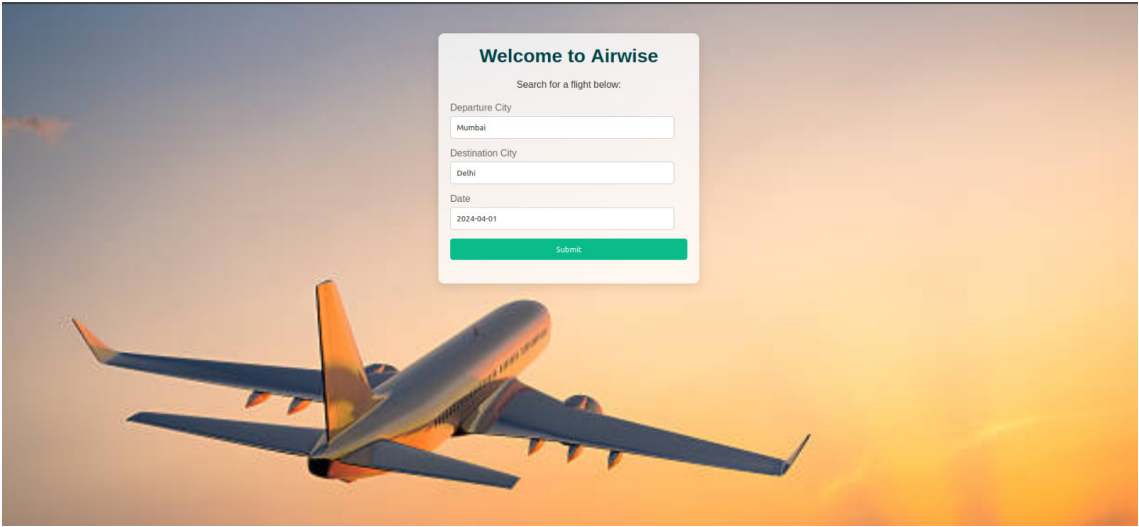
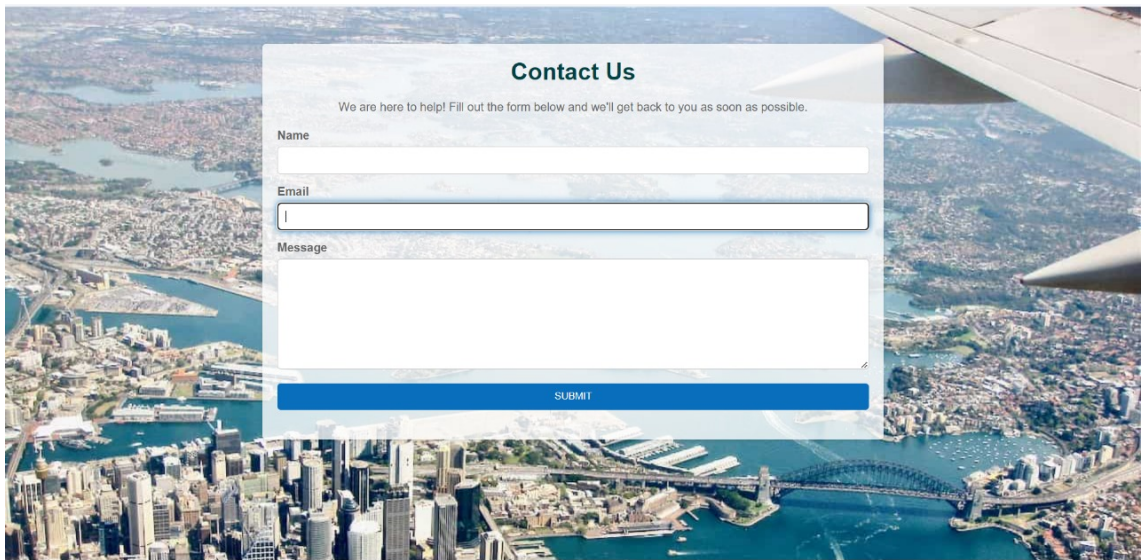


Figure 15: Flight Search Filter

Flight Search Result		
<input type="radio"/> Flight ID: 101	Departure: BOM Destination: DEL	Departure Time: 08:00:00 Arrival Time: 10:30:00
<input checked="" type="radio"/> Flight ID: 111	Departure: BOM Destination: DEL	Departure Time: 09:00:00 Arrival Time: 11:30:00
<input type="radio"/> Flight ID: 112	Departure: BOM Destination: DEL	Departure Time: 11:30:00 Arrival Time: 14:00:00
<input type="radio"/> Flight ID: 113	Departure: BOM Destination: DEL	Departure Time: 12:00:00 Arrival Time: 14:30:00
<input type="radio"/> Flight ID: 114	Departure: BOM Destination: DEL	Departure Time: 14:00:00 Arrival Time: 16:30:00
<input type="radio"/> Flight ID: 115	Departure: BOM Destination: DEL	Departure Time: 15:30:00 Arrival Time: 18:00:00
Select Flight		

Figure 16: Result of Flight Search Query

An aerial photograph of a city, likely Sydney, Australia, featuring the Sydney Harbour Bridge and the Sydney Opera House. Overlaid on the image is a semi-transparent white contact form. The form has a blue header with the text "Contact Us". Below the header, there is a line of text: "We are here to help! Fill out the form below and we'll get back to you as soon as possible." The form contains three input fields: "Name", "Email", and "Message". The "Message" field is a larger text area. At the bottom of the form is a blue button with the text "SUBMIT".

Contact Us

We are here to help! Fill out the form below and we'll get back to you as soon as possible.

Name

Email

Message

SUBMIT

Figure 17: Customer Support Page