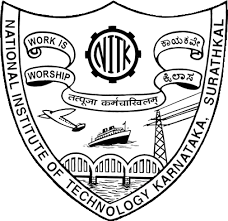
**National Institute of Technology Karnataka Surathkal,**

**Mangalore, India**

**Department of Computer Science and Engineering**



**Database Management System Project**

**Airline Reservation System**

**D. R. Vikas - 221CS215**

**Harsha Guandapalli - 221CS222**

**Manohar Rohit Vijay - 221CS230**

**Rayala Abhyuday - 221CS239**

**Regulagadda Rohan Chandra - 221CS241**

**Content page**

* Introduction 3
* Literature Survey 3
* Objectives 4
* Features 5
* Requirement specification

Front End 6

Back End 6

* System Design

Database Schema Diagram 7

**Introduction**

In the competitive aviation industry, effective airline management is crucial for operational efficiency, customer satisfaction, and profitability. Modern airlines utilize online booking systems to revolutionize the passenger experience, offering convenience and accessibility. These systems streamline reservations, minimize errors, and handle high transaction volumes efficiently. Online bookings also provide valuable data for tailoring services and marketing efforts. However, successful implementation requires careful planning and investment in user-friendly, secure technology infrastructure. Integration with other operational functions is essential for a seamless passenger experience. Airlines that embrace innovative technologies and prioritize customer-centric strategies can thrive by leveraging online booking systems to enhance satisfaction, drive revenue growth, and maintain competitiveness.

**Literature Survey**

The system enables users to register for an account, providing personal details such as name, address, contact information, gender, email, age, and creating a unique user ID and password. Registered users can directly book flights, while new users must register before booking. Users can check flight availability by inputting origin and destination cities along with the date of travel. Upon selecting a flight, the system prompts for confirmation before proceeding to reserve seats. For canceling or rescheduling tickets, users must provide ticket and flight details.

The system facilitates automated data entry as passengers input their details directly, minimizing manual entry errors. Real-time validation checks ensure accuracy and completeness of passenger information while automated processes prevent double bookings by updating seat availability instantly. Security measures include SSL encryption for data transmission and integration with secure payment gateways. Optional two-factor authentication adds an extra layer of security. Passengers can filter flights and compare fares across different airlines, aided by detailed flight listings and route maps. Flexible booking options cater to diverse travel needs, with features like seat selection and instant seat availability status. Automatic waitlist management notifies passengers of available seats, reducing manual work and streamlining the booking process.

The survey begins by outlining the fundamental objectives of ARSs, which include providing seamless ticket reservation, cancellation, and rescheduling functionalities, all while centralizing information to enable easy updates and minimize manual work. However, existing systems face notable challenges, such as the lack of real-time flight tracking, discrepancies in ticket pricing across different platforms, limitations in seat selection for connecting flights, and issues related to luggage tracking. To address these challenges, proposed solutions include integrated real-time flight tracking, centralized pricing information, advanced seat selection for connecting flights, and enhanced luggage tracking features. Additionally, future enhancements could encompass ticket editing capabilities, Aadhaar integration for ID verification, and health/Covid status verification for passengers, all aimed at further improving system functionality and user experience. The survey concludes by highlighting the significance of ongoing enhancements and technological advancements in meeting the evolving needs of customers and the airline industry, ultimately underlining the critical role of Java-based ARSs in modern air travel management.

**Objectives**

* User Convenience: Provide a user-friendly interface for easy flight searches based on preferences like dates, destinations, and budget considerations.

* Comprehensive Search: Offer a comprehensive search functionality enabling users to compare flights from different airlines based on factors like price, duration, and layovers.

* Competitive Pricing: Ensure competitive pricing options, enabling users to find affordable and value-for-money flight options.

* Canceling or Rescheduling: Provide a straightforward process for canceling flights, allowing users to easily navigate and manage their bookings.

* Booking and Payment: Facilitate a seamless booking process with secure online payment options for quick and efficient flight reservations.

* Real-time Updates: Deliver real-time updates on flight availability, schedule changes, and relevant information for users' travel plans.

* Customer Support: Offer effective customer support services to assist users with any issues, changes, or queries related to their flight bookings.

* Integration with Other Services: Integrate the flight booking system with other travel-related services like hotel reservations, bus bookings, and travel packages, providing users with a one-stop solution.

* Catering Services: If applicable, allow users to customize in-flight meals by selecting specific dishes, dietary restrictions, or special requests during the booking process.

**Features**

**User Management**

* User Profiles: Store personal information like name, contact details, and preferences.
* Authentication and Authorization: Manage login, registration, and permissions.
* User Preferences: Save preferred airlines, meal choices, seating preferences, etc.

**Flight Information**

* Flights: Details include flight number, airline, departure and arrival times, duration, and available classes (economy, business, first class).
* Airports: Information on airport names, locations (city, country), and facilities.

**Booking Management**

* Flight Bookings: Records of user bookings, including flight details, class, seat selection, and fare.
* Payment Information: Securely store payment transactions, including payment method, transaction ID, and status.
* Booking Status: Track booking status (confirmed, canceled, pending).
* Meals and Amenities: Details on available meal options and additional services (Wi-Fi, entertainment).

**Search and Filters**

* Flight Search: Functionality to search for flights based on parameters like date, destination, and number of passengers.
* Filters and Sorting: Options to filter and sort search results by price, duration, airline, departure time, etc.

**Required Specifications**

**Front end:**

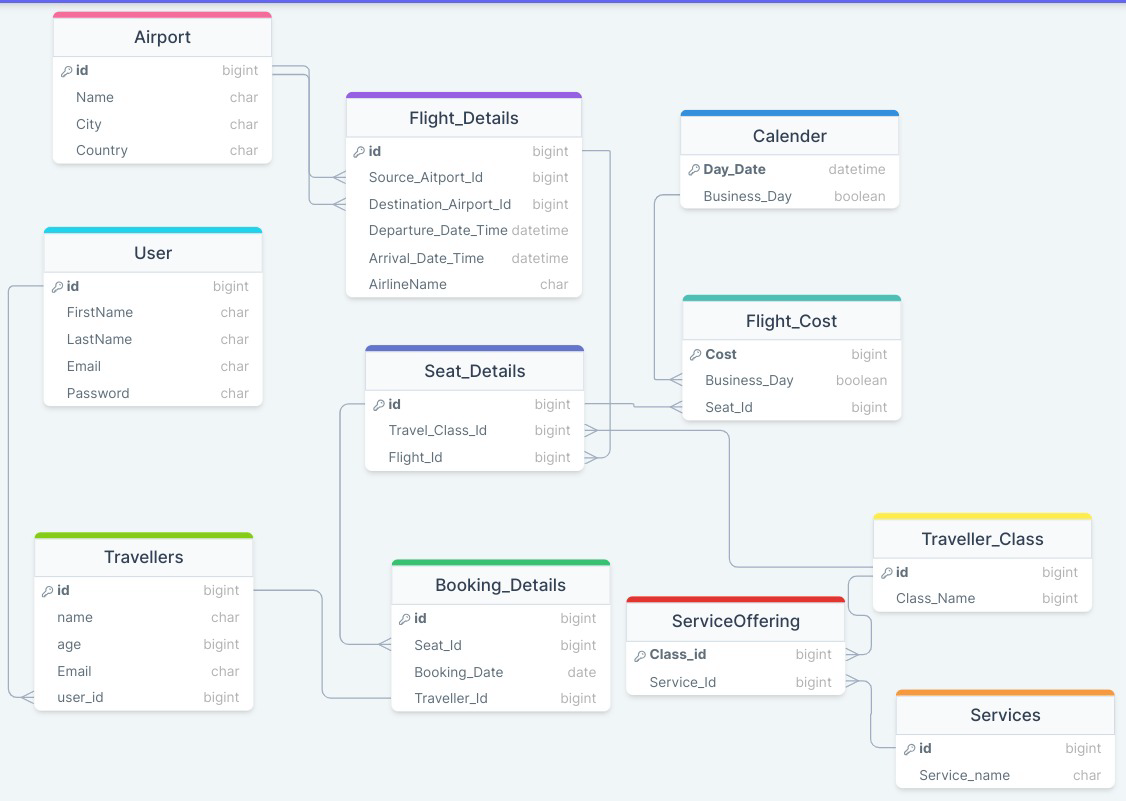
integrating React JS into the frontend framework enhances the UI/UX of the airline reservation system by enabling the creation of dynamic, responsive, and interactive user interfaces, ultimately improving the booking experience for users.

**Back end:**

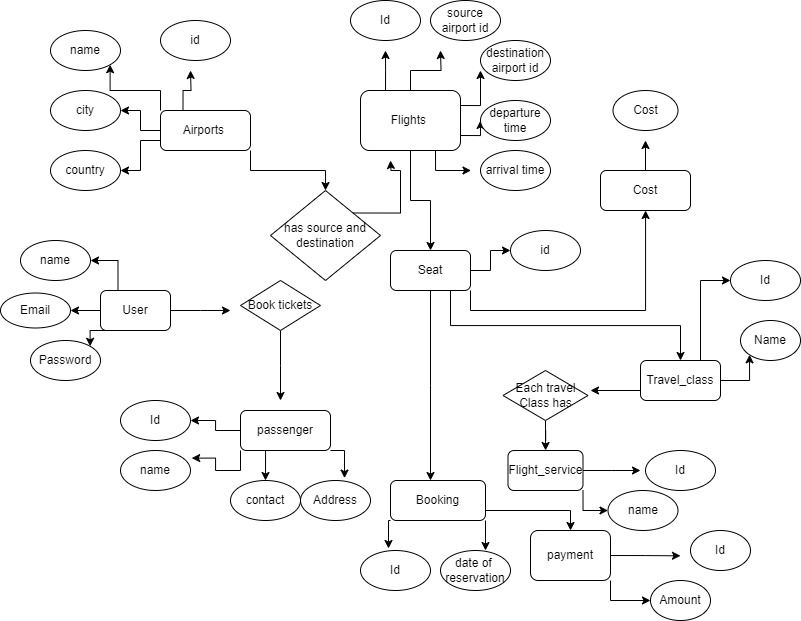
We will be using Node JS runtime to run our backend express server which enhances the communication between the frontend queries and database of our application. As informed, we will be using MySQL as our database.

**System Design**

**Database Schema Design:**

****

**ER diagram:**

****

**Normalization Process**

A relation is said to be in 1 normal form in DBMS (or 1NF) when it consists of an atomic value. In simpler words, 1NF states that a table's attribute would not be able to hold various values- it will only be able to hold an attribute of a single value. So, we normalized our tables in such a way that every attribute is single valued. For example, each traveler class has multiple services, so we created a separate table called service\_offering which maps each travel\_class with multiple services listed in the services table.

A relation is said to be in the 2nd Normal Form in DBMS (or 2NF) when it is in the First Normal Form but has no non-prime attribute functionally dependent on any candidate key's proper subset in a relation. In the same way, we have tables where each non-prime attribute is dependent only on primary key, but not other non-prime attributes.For example, In Airport Table, every attribute like city, airport name are all dependent only on Airport\_Id.