ASSIGNMENT 3a

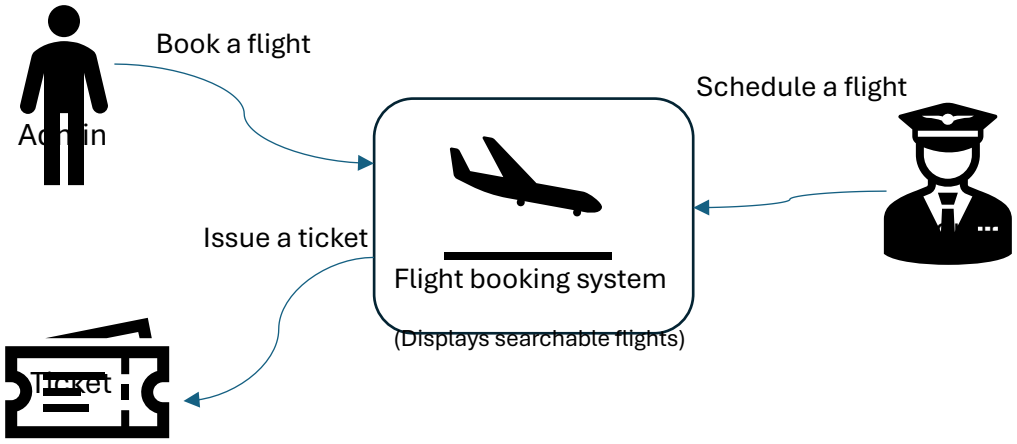

Figure 1: Context diagram

Problem Description: Flight Booking Management Application

Overview

The **Flight Booking Management Application** is designed to provide users with a seamless and intuitive platform to manage their flight bookings. The application allows users to view, create, update, and delete their flight bookings, while also providing detailed information about each booking. The system integrates with a MongoDB database to store and retrieve booking data, and it uses a Next.js-based frontend with Material-UI for a premium and responsive user interface.

Key Features

1. **User Authentication**:
    - Users can log in using their email and password.
    - The application ensures that only authenticated users can access their booking history and perform CRUD operations.

2. **Booking Management**:
    - **View Bookings**: Users can view a list of all their flight bookings, including details such as flight origin, destination, departure date, price, and payment status.
    - **View Booking Details**: Users can click on a booking to view detailed information in a modal, including flight details, fare type, seat number, and extras.
    - **Edit Bookings**: Users can update booking details such as fare type, seat number, extras, total price, and payment status.
    - **Delete Bookings**: Users can delete a booking if it is no longer needed.
3. **Database Integration**:
    - The application uses MongoDB to store data about users, flights, and bookings.
    - The database is populated with sample data, including airports, airlines, and flights, to simulate a real-world scenario.
4. **API Integration**:
    - The backend provides RESTful API endpoints for CRUD operations on bookings.
    - The frontend interacts with these endpoints to fetch, create, update, and delete bookings.
5. **Responsive and Premium UI**:
    - The frontend is built using Material-UI, providing a clean and modern user interface.
    - The application is fully responsive, ensuring a consistent experience across devices.

Problem Statement

The primary goal of this application is to address the challenges users face when managing their flight bookings. Traditional methods of managing bookings, such as contacting customer support or navigating through complex airline websites, can be time-consuming and frustrating. This application aims to simplify the process by providing a centralized platform where users can easily view, edit, and manage their bookings.

Key Challenges Addressed

1. **Centralized Booking Management**:
    - Users often have bookings with multiple airlines, making it difficult to keep track of all their flights.

- This application consolidates all bookings in one place, providing a unified view of the user's travel plans.

2. **Real-Time Updates**:
    - Users need to update their bookings frequently, such as changing seat numbers or adding extras.
    - The application allows users to make these changes in real-time, ensuring their booking details are always up-to-date.
3. **User-Friendly Interface**:
    - Many airline websites have complex and outdated interfaces, making it difficult for users to find the information they need.
    - This application provides a clean and intuitive interface, making it easy for users to manage their bookings.
4. **Data Security**:
    - Users need to trust that their personal and booking information is secure.
    - The application ensures data security by using authentication and authorization mechanisms to protect user data.

Technical Details

1. **Frontend**:
    - Built using Next.js and Material-UI.
    - Provides a responsive and interactive user interface.
    - Communicates with the backend API to fetch and update booking data.
2. **Backend**:
    - Built using Next.js API routes.
    - Provides RESTful endpoints for CRUD operations on bookings.
    - Integrates with MongoDB to store and retrieve data.
3. **Database**:
    - MongoDB is used to store data about users, flights, and bookings.
    - The database is populated with sample data to simulate a real-world scenario.
4. **Authentication**:

- Users are authenticated using NextAuth.js.
- Only authenticated users can access their booking history and perform CRUD operations.

Use Cases

1. **User Logs In and Views Bookings**:
   - A user logs in to the application and is redirected to their profile page.
   - The application fetches and displays a list of all their bookings.
2. **User Views Booking Details**:
   - The user clicks on a booking to view detailed information.
   - A modal opens, displaying the booking details, including flight information, fare type, seat number, and extras.
3. **User Edits a Booking**:
   - The user clicks the "Edit" button on a booking.
   - A dialog opens, allowing the user to update booking details such as fare type, seat number, and extras.
   - The user saves the changes, and the application updates the booking in the database.
4. **User Deletes a Booking**:
   - The user clicks the "Delete" button on a booking.
   - The application removes the booking from the database and updates the UI to reflect the change.
5. **User Creates a New Booking**:
   - The user navigates to the booking creation page.
   - The user selects a flight, enters booking details, and submits the form.
   - The application creates a new booking in the database and adds it to the user's booking history.