Flight Ticket Booking Application Using Mern Stack
Smart Internz(Naan Mudhalvan)

Project report submitted in fulfilment for the requirement of the degree of

BACHELOR OF TECHNOLOGY IN

INFORMATION TECHNOLOGY

By

Suriya Prakash S	211121205027
Yoganathan B	211121205029
Hariharan R S	211121205010
Hariharan S	211121205011
Seran E	211121205026

MADHA ENGINEERING COLLEGE
KUNDRATHUR, CHENNAI - 600069

NOV 2024

ABSTRACT

Building a flight ticket booking web application using the MERN stack involves leveraging MongoDB, Express.js, React, and Node.js to create a robust, scalable, and user-friendly platform. MongoDB serves as the database, offering a flexible schema design to store extensive details about flights, users, and bookings. Express.js handles server-side logic, RESTful API creation, and middleware integration, efficiently managing user authentication, authorization, and file uploads. React powers the frontend, providing a dynamic, responsive, and interactive user interface, while utilizing state management tools like Redux or Context API to handle complex application states. Node.js facilitates server-side execution of JavaScript, managing asynchronous operations and ensuring high performance.

Key features of the application include secure user authentication and authorization, a comprehensive flight catalogue with search and filter options (e.g., by destination, date, and airline), a seamless booking and payment system, user reviews and ratings for airlines, and an admin dashboard for managing flights, users, and bookings. The MERN stack's architecture supports scalability and flexibility, ensuring that the application can adapt to changing requirements. Its combination of fast server-side processing and responsive frontend experience enhances user engagement and satisfaction. Comprehensive security measures further protect user data and maintain the integrity of the application, making it a reliable platform for online flight ticket booking.

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

The advent of online platforms has revolutionized the way users book flights, offering unparalleled convenience, efficiency, and accessibility. These platforms empower users to explore options, compare prices, and manage travel plans seamlessly. Leveraging the MERN stack—comprising MongoDB, Express.js, React.js, and Node.js—a flight ticket booking application can deliver a robust, scalable, and user-centric solution that meets modern travel needs.

This application is designed to provide users with a dynamic and intuitive interface for browsing flights, booking tickets, and tracking their bookings. It also offers administrators an efficient dashboard for managing flights, users, and transactions. MongoDB serves as the database, offering a flexible and scalable solution for storing flight schedules, user details, and bookings. Express.js ensures secure and efficient server-side operations, handling RESTful APIs, middleware integration, and user authentication.

React.js drives the front-end experience, offering a responsive and interactive interface. Users can search and filter flights by criteria such as destinations, dates, airlines, and prices, ensuring they find the best options. State management tools like Redux enhance the handling of complex application states. Node.js powers server-side execution, enabling high performance and efficient asynchronous processing, crucial for handling real-time data and heavy traffic.

Key features include advanced flight search and filtering, secure booking and payment systems, user reviews and ratings for airlines, and real-time updates for bookings. For administrators, the dashboard facilitates flight scheduling, user management, and operational oversight, ensuring smooth management of the platform.

The MERN stack ensures scalability, enabling the platform to grow with user demands while maintaining high performance and robust security. With encrypted data, secure payments, and responsive design, this application redefines the flight booking process, making it more accessible and reliable for travelers worldwide. By combining cutting-edge technology with a user-centric approach, the platform transforms how users plan and manage their journeys.

1.1 Components of the MERN Stack in the Flight Ticket Booking Application:

MongoDB (Database):

- **Role**: MongoDB serves as the database to store flight details, user profiles, and booking records.
- **Features**:
    - Stores flight data, including departure and arrival locations, schedules, airlines, seat availability, and fares.
    - Maintains user information, such as preferences, booking history, and contact details, ensuring flexible and efficient data queries and updates.

Express.js (Backend Framework):

- **Role**: Handles backend logic and serves as a bridge between the frontend and database.
- **Features**:
    - Provides API endpoints for functionalities like user authentication, flight searches, and booking management.
    - Handles HTTP requests (GET, POST, PUT, DELETE) and processes data for seamless communication with the frontend.

React.js (Frontend Library):

- **Role**: Powers the dynamic and responsive user interface for the application.
- **Features**:
    - Users can search for flights, filter results, book tickets, and manage their itineraries.
    - Employs a modular component-based architecture for reusable UI elements.
    - Ensures consistent state management using tools like Redux or React's built-in context API for data such as user sessions and booking carts.

Node.js (Server-Side JavaScript Runtime):

- **Role**: Runs the backend server, ensuring a high-performance environment for processing multiple user requests.
- **Features**:
    - Enables JavaScript usage on both client and server sides for uniformity in development.
    - Scalable to handle concurrent user operations like flight searches and ticket bookings.

2.PRE-REQUISITES

Core Technologies:

- **HTML, CSS, and JavaScript:** Fundamental for building the application's structure, styling, and client-side interactivity.
- **React.js:** Frontend framework for creating a dynamic and responsive user interface.
- **Node.js:** Runtime environment for server-side JavaScript execution.
- **Express.js:** Backend framework for building RESTful APIs and handling server-side logic.
- **MongoDB:** NoSQL database for managing data related to flights, users, and bookings.

Database Connectivity:

- **Mongoose:** ODM (Object Data Modeling) library for MongoDB to simplify CRUD operations and schema management.
- **MongoDB Drivers:** Native MongoDB drivers for direct database interactions.

- **Version Control**

- **Git:** Version control system for collaborative development and tracking changes.
- **Hosting Platforms:** GitHub, GitLab, or Bitbucket for managing and sharing repositories.

Development Tools:

- **Code Editors:**
  - Visual Studio Code
  - Sublime Text
  - WebStorm
- **Package Managers:**
  - npm (Node Package Manager)
  - Yarn

Additional Tools:

- **Postman:** For testing and debugging RESTful APIs.
- **Browser Developer Tools:** For inspecting and debugging frontend code.
- **Testing Frameworks:**
  - Jest for unit testing.

3.PROJECT OVERVIEW

3.1 Purpose:

The flight ticket booking application serves as a comprehensive platform for modern travel needs. For users, it provides an intuitive and efficient way to search for flights, book tickets, and manage their travel plans seamlessly. The platform simplifies the often complex process of flight selection and booking, ensuring a smooth user experience. For administrators, the application offers robust tools to manage flights, update inventory, and track bookings effectively, streamlining operational workflows and enhancing overall efficiency.

3.2 Features and Functionalities:

1. **Flight Search and Filter:**
    a. Users can search and filter flights based on multiple criteria such as departure and arrival times, destinations, airlines, price ranges, and class preferences (e.g., economy, business).

    b. Real-time updates ensure that users can make informed decisions quickly.

2. **Secure Booking and Payments:**
    a. A seamless and secure checkout process integrates multiple payment gateways, offering users flexibility in payment methods (e.g., credit/debit cards, UPI, wallets).

    b. End-to-end encryption ensures user data and payment information are protected.

3. **Booking Management:**
    a. A user dashboard allows customers to view their booking history, modify travel details, or cancel reservations effortlessly.

    b. Automated email or SMS notifications keep users informed about booking confirmations, flight changes, and reminders.

4. **Personalized Recommendations:**
    a. The application utilizes user preferences and past booking history to provide tailored flight suggestions, enhancing the booking experience.

4.ARCHITECTURE

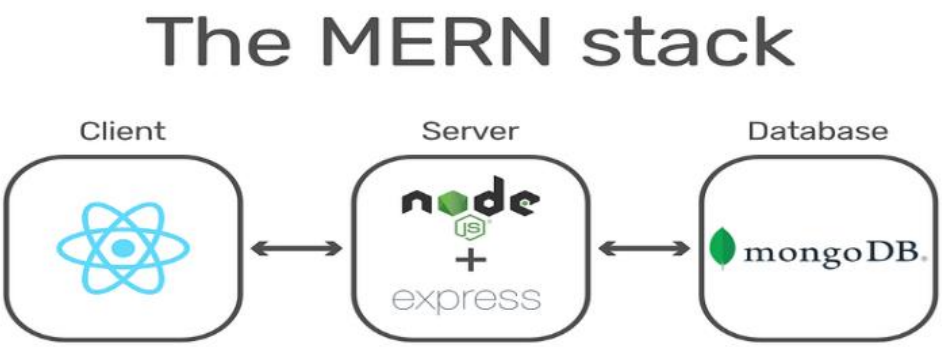

Fig 4.1 Architecture

Frontend:

The frontend is developed using **React.js**, offering an interactive, dynamic, and responsive user interface. It is designed with reusable components, such as flight search panels, booking forms, and user dashboards, ensuring a seamless and engaging user experience. State management tools like **Redux** or the **Context API** are integrated to handle dynamic interactions and maintain application-wide states efficiently.

Backend:

The backend leverages **Node.js** and **Express.js** to handle server-side operations and business logic. It efficiently manages API requests and responses through a robust set of RESTful APIs, enabling functionalities such as flight searches, bookings, and user authentication. Middleware integration ensures security, error handling, and data validation throughout the backend process.

Database:

MongoDB, a NoSQL database, is used to store and manage critical data such as flight schedules, user profiles, and booking records. Its flexible schema design allows for efficient handling of complex relationships and large datasets. Collections are structured to support fast queries, ensuring high performance and scalability.

5.SETUP INSTRUCTIONS

Prerequisites:

1. **Node.js:**
    a. Install Node.js (version 14 or higher) from the official website. This is required for running the backend server and managing JavaScript dependencies.
2. **MongoDB:**
    a. Set up MongoDB, either by installing the Community Edition for local development or using MongoDB Atlas for a cloud-based database.
    b. Ensure MongoDB is running before starting the application.
3. **npm or yarn:**
    a. npm (Node Package Manager) is included with Node.js installation. Alternatively, you can install **yarn** from the official website for managing project dependencies.
4. **Git:**
    a. Install Git for version control and to clone the project repository from GitHub.

Installation Steps:

1. **Clone the Repository:**
   Use Git to clone the project repository to your local machine.

bash
Copy code
git clone https://github.com/example/Flight-Ticket-Booking-App

2. **Navigate to the Project Folder:**
   Change the directory to the project folder:

bash
Copy code
cd Flight-Ticket-Booking-App

3. **Install Dependencies:**

    a. **Backend Dependencies**:
    Navigate to the backend directory and install the required packages using npm or yarn.

bash
Copy code
cd backend
npm install

b. **Frontend Dependencies**:

bash
Copy code
cd frontend
npm install

Start the Application:

a. **Backend**: Start the Node.js server by running the following command in the backend directory:

bash
Copy code
npmstart

The backend server will be available on the defined port (default: 5000).

b. **Frontend**: Start the React development server by running the following command in the frontend directory:

bash
Copy code
npmstart

The frontend server will be available at http://localhost:3000.

2. **Access the Application:**
   a. Open a browser and navigate to http://localhost:3000 to access the frontend.
   b. The backend APIs can be tested via tools like Postman or directly integrated with the frontend application.

6.Environment Configuration

Create a .env File:

- **Backend Directory:**
    - In the backend directory, create a .env file to securely store sensitive information and configuration-specific keys such as:

plaintext
Copy code
PORT=5000
MONGODB_URI=mongodb://localhost:27017/flight_booking
JWT_SECRET=your_jwt_secret_key
PAYMENT_API_KEY=your_payment_gateway_api_key
EMAIL_SERVICE_KEY=your_email_service_api_key

6.1 Frontend Structure:

- **Frontend Directory (src/):**
    - Contains reusable and modular components for UI.
    - Houses the following subdirectories:
        - Components/: Includes reusable components like search bars, booking forms, and cards.
        - Pages/: Contains pages for different user roles:
            - **User Pages:** Search flights, booking history, and account settings.
            - **Admin Pages:** Manage flights, users, and booking details.
            - **Seller Pages:** Manage flight listings and availability.
    - **src/app.js:**
        - Acts as the central hub for API communication with the backend.
        - Manages routing and overall state for seamless navigation.

6.2 Backend Structure:

- **Backend Directory (db/):**
    - Organizes logic into separate modules for handling API requests and responses:
        - User/: Contains logic for user-related operations like login, registration, and booking history.
        - Seller/: Includes logic for flight management, pricing, and availability updates.

        - Admin/: Manages system-wide operations like user and flight data control.

- **Models and Schemas (backend/jsfiles):**
    - Contains Mongoose schemas for database collections:
        - User.js: Defines the schema for user profiles.
        - Flight.js: Schema for flight details such as departure, arrival, price, and seat availability.
        - Booking.js: Schema for storing booking records.

6.3 Implementation:

1. Create a .env File:

- In the backend root, define environment variables for:
    - **Database Connection String**: MONGODB_URI
    - **Server Port**: PORT
    - **JWT Secret Key**: JWT_SECRET
    - **Payment Gateway API Key**: PAYMENT_API_KEY
    - **Email Service API Key**: EMAIL_SERVICE_KEY

2. Install dotenv Package:

- Install the package to enable loading environment variables:

bash
Copy code
npm install dotenv

3. Load Environment Variables:

- At the start of the backend server (index.js or app.js), configure dotenv:

javascript
Copy code
require('dotenv').config();

7.Folder Structure

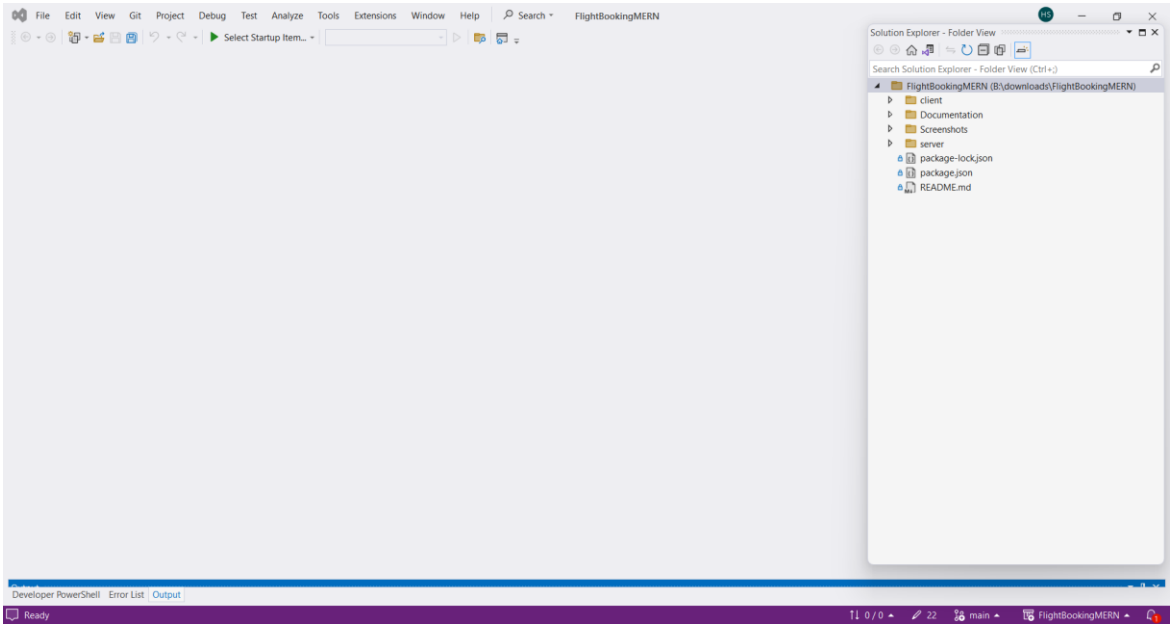

Fig 7.1 Folder Structure

Frontend:

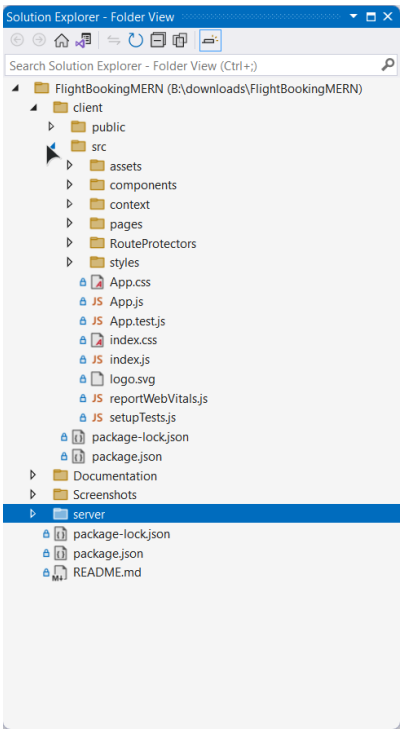

Fig 7.2 Frontend

Backend:

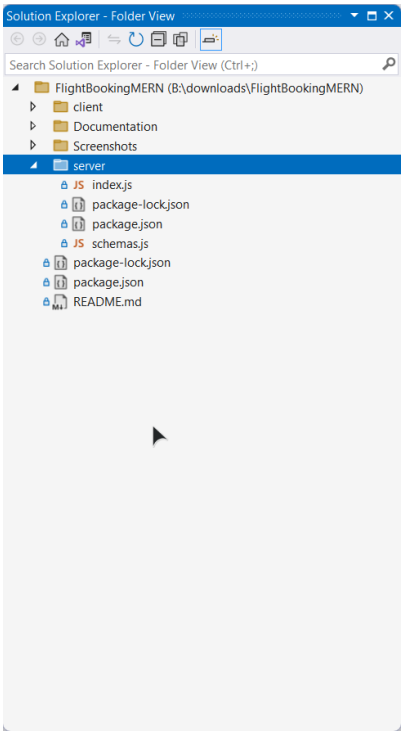

Fig 7.3 Backend

8.Running the Application

➢ Frontend:

• To navigate to frontend, use the command `cd frontend` in terminal.

• Navigate to the frontend directory and run `npm run dev` to start the React application.

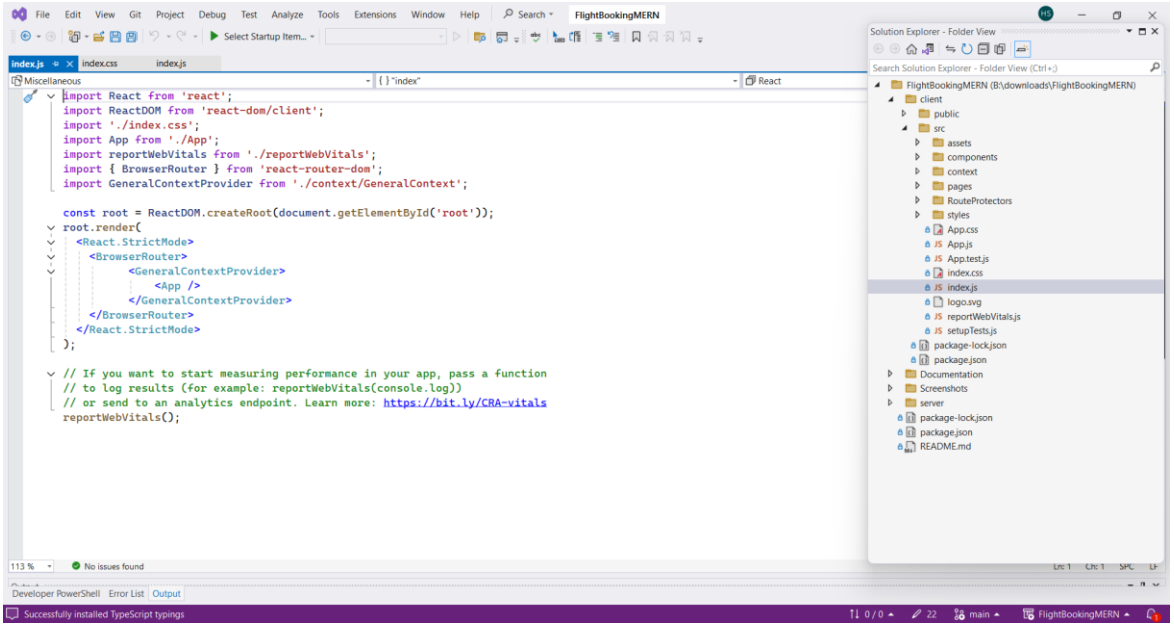

Fig 8.1 Frontend Application running

Backend:

• To navigate to backend, use the command `cd backend` in terminal.

• Navigate to the backend directory and run `npm start` to start the Express.js backend.

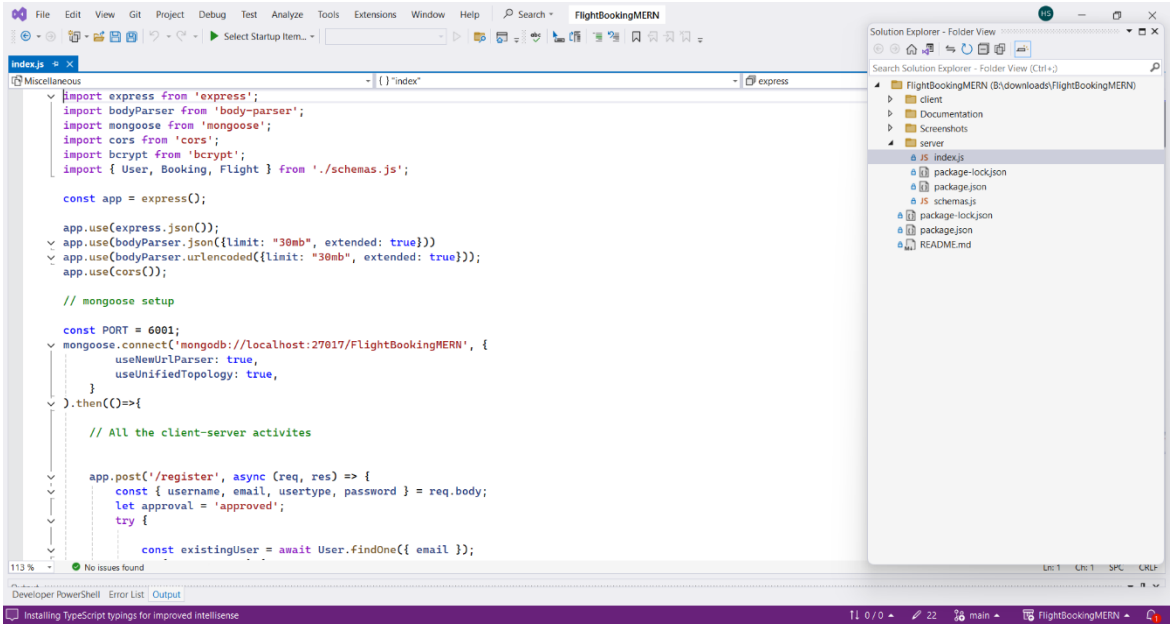

Fig 8.2 Backend Application running

9.Extension & Packages

9.1 Backend Packages:

Package. Json:

```
{
    "dependencies": {
        "axios": "^1.4.0",
        "bootstrap": "^5.3.1",
        "react": "^18.2.0",
        "react-dom": "^18.2.0",
        "react-router-dom": "^6.14.2",
        "react-scripts": "^3.0.1",
        "web-vitals": "^2.1.4"
    }


}
```

• This `package. Json` file defines the configuration for a Node.js backend project. The project, named "backend," is at version 1.0.0. It specifies the main entry point as ‘index.js’ and includes a script to start the server using ‘nodemon’, a tool that automatically restarts the server upon changes in the code.

• Dependencies include ‘cors’ for enabling cross-origin resource sharing, ‘express’ for building the web server, ‘mongoose’ for interacting with MongoDB, and ‘multer’ for handling file uploads. Additionally, it lists ‘nodemon’ and ‘nodeman’ (likely a typo or misconfigured dependency) for development purposes.

• The `license` is set to ISC, indicating the open-source license under which the project is distributed. This setup ensures that all necessary packages are installed, and the server runs efficiently with automatic restarts during development.

Package-lock-Json:

```
{
  "name": "Flight-Booking-App-MERN-main",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "dependencies": {
        "axios": "^1.4.0",
        "bootstrap": "^5.3.1",
        "react": "^18.2.0",
        "react-dom": "^18.2.0",
```

```
        "react-router-dom": "^6.14.2",
        "react-scripts": "^3.0.1",
        "web-vitals": "^2.1.4"
      }
    },
    "node_modules/@ampproject/remapping": {
      "version": "2.3.0",
      "resolved": "https://registry.npmjs.org/@ampproject/remapping/-
/remapping-2.3.0.tgz",
                                          "integrity":           "sha512-
30iZtAPgz+LTIYoeivqYo853f02jBYSd5uGnGpkFV0M3xOt9aN73erkgYAmZU43x4Vfqc
nLxW9Kpg3R5LC4YYw==",
      "license": "Apache-2.0",
      "dependencies": {
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.24"
      },
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@babel/code-frame": {
      "version": "7.26.2",
         "resolved":  "https://registry.npmjs.org/@babel/code-frame/-
/code-frame-7.26.2.tgz",
                                          "integrity":           "sha512-
RJlIHRueQgwWitWgF8OdFYGZX328Ax5BCemNGlqHfplnRT9ESi8JkFlvaVYbS+UubVY6d
pv87Fs2u5M29iNFVQ==",
      "license": "MIT",
      "dependencies": {
        "@babel/helper-validator-identifier": "^7.25.9",
        "js-tokens": "^4.0.0",
        "picocolors": "^1.0.0"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/compat-data": {
      "version": "7.26.2",
        "resolved":  "https://registry.npmjs.org/@babel/compat-data/-
/compat-data-7.26.2.tgz",
                                          "integrity":           "sha512-
Z0WgzSEa+aUcdiJuCIqgujCshpMWgUpgOxXotrYPSA53hA3qopNaqcJpyr0hVb1FeWdnq
FA35/fUtXgBK8srQg==",
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
```

```
      }
    },
    "node_modules/@babel/core": {
      "version": "7.26.0",
         "resolved":  "https://registry.npmjs.org/@babel/core/-/core-
7.26.0.tgz",
                                        "integrity":           "sha512-
i1SLeK+DzNnQ3LL/CswPCa/E5u4lh1k6IAEphON8F+cXt0t9euTshDru0q7/IqMa1PMPz
5RnHuHscF8/ZJsStg==",
      "license": "MIT",
      "dependencies": {
        "@ampproject/remapping": "^2.2.0",
        "@babel/code-frame": "^7.26.0",
        "@babel/generator": "^7.26.0",
        "@babel/helper-compilation-targets": "^7.25.9",
        "@babel/helper-module-transforms": "^7.26.0",
        "@babel/helpers": "^7.26.0",
        "@babel/parser": "^7.26.0",
        "@babel/template": "^7.25.9",
        "@babel/traverse": "^7.25.9",
        "@babel/types": "^7.26.0",
        "convert-source-map": "^2.0.0",
        "debug": "^4.1.0",
        "gensync": "^1.0.0-beta.2",
        "json5": "^2.2.3",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/babel"
      }
    },
    "node_modules/@babel/core/node_modules/semver": {
      "version": "6.3.1",
           "resolved":   "https://registry.npmjs.org/semver/-/semver-
6.3.1.tgz",
                                        "integrity":           "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9
yArRkyrQxTO6XjMzA==",
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }
    },
    "node_modules/@babel/generator": {
```

```
      "version": "7.26.2",
          "resolved":  "https://registry.npmjs.org/@babel/generator/-
/generator-7.26.2.tgz",
                                        "integrity":           "sha512-
zevQbhbau95nkoxSq3f/DC/SC+EEOUZd3DYqfSkMhY2/wfSeaHV1Ew4vk8e+x8lja31Ib
yuUa2uQ3JONqKbysw==",
      "license": "MIT",
      "dependencies": {
        "@babel/parser": "^7.26.2",
        "@babel/types": "^7.26.0",
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.25",
        "jsesc": "^3.0.2"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-annotate-as-pure": {
      "version": "7.25.9",
       "resolved": "https://registry.npmjs.org/@babel/helper-annotate-
as-pure/-/helper-annotate-as-pure-7.25.9.tgz",
                                        "integrity":           "sha512-
gv7320KBUFJz1RnylIg5WWYPRXKZ884AGkYpgpWW02TH66Dl+HaC1t1CKd0z3R4b6hdYE
cmrNZHUmfCP+1u3/g==",
      "license": "MIT",
      "dependencies": {
        "@babel/types": "^7.25.9"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
       "node_modules/@babel/helper-builder-binary-assignment-operator-
visitor": {
      "version": "7.25.9",
       "resolved":  "https://registry.npmjs.org/@babel/helper-builder-
binary-assignment-operator-visitor/-/helper-builder-binary-
assignment-operator-visitor-7.25.9.tgz",
                                        "integrity":           "sha512-
C47lC7LIDCnz0h4vai/tpNOI95tCd5ZT3iBt/DBH5lXKHZsyNQv18yf1wIIg2ntiQNgmA
vA+DgZ82iW8Qdym8g==",
      "license": "MIT",
      "dependencies": {
        "@babel/traverse": "^7.25.9",
        "@babel/types": "^7.25.9"
      },
      "engines": {
```

```
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-compilation-targets": {
      "version": "7.25.9",
            "resolved":   "https://registry.npmjs.org/@babel/helper-
compilation-targets/-/helper-compilation-targets-7.25.9.tgz",
                                        "integrity":           "sha512-
j9Db8Suy6yV/VHa4qzrj9yZfZxhLWQdVnRlXxmKLYlhWUVB1sB2G5sxuWYXk/whHD9iW7
6PmNzxZ4UCnTQTVEQ==",
      "license": "MIT",
      "dependencies": {
        "@babel/compat-data": "^7.25.9",
        "@babel/helper-validator-option": "^7.25.9",
        "browserslist": "^4.24.0",
        "lru-cache": "^5.1.1",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
                              "node_modules/@babel/helper-compilation-
targets/node_modules/semver": {
      "version": "6.3.1",
           "resolved":   "https://registry.npmjs.org/semver/-/semver-
6.3.1.tgz",
                                        "integrity":           "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9
yArRkyrQxTO6XjMzA==",
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }
    },
    "node_modules/@babel/helper-create-class-features-plugin": {
      "version": "7.25.9",
        "resolved":  "https://registry.npmjs.org/@babel/helper-create-
class-features-plugin/-/helper-create-class-features-plugin-
7.25.9.tgz",
                                        "integrity":           "sha512-
UTZQMvt0d/rSz6KI+qdu7GQze5TIajwTS++GUozlw8VBJDEOAqSXwm1WvmYEZwqdqSGQs
hRocPDqrt4HBZB3fQ==",
      "license": "MIT",
      "dependencies": {
        "@babel/helper-annotate-as-pure": "^7.25.9",
        "@babel/helper-member-expression-to-functions": "^7.25.9",
        "@babel/helper-optimise-call-expression": "^7.25.9",
```

```
        "@babel/helper-replace-supers": "^7.25.9",
                "@babel/helper-skip-transparent-expression-wrappers":
"^7.25.9",
        "@babel/traverse": "^7.25.9",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0"
      }
    },
                  "node_modules/@babel/helper-create-class-features-
plugin/node_modules/semver": {
      "version": "6.3.1",
           "resolved":   "https://registry.npmjs.org/semver/-/semver-
6.3.1.tgz",
                                        "integrity":           "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9
yArRkyrQxTO6XjMzA==",
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }
    },
    "node_modules/@babel/helper-create-regexp-features-plugin": {
      "version": "7.25.9",
        "resolved":  "https://registry.npmjs.org/@babel/helper-create-
regexp-features-plugin/-/helper-create-regexp-features-plugin-
7.25.9.tgz",
                                        "integrity":           "sha512-
ORPNZ3h6ZRkOyAa/SaHU+XsLZr0UQzRwuDQ0cczIA17nAzZ+85G5cVkOJIj7QavLZGSe8
QXUmNFxSZzjcZF9bw==",
      "license": "MIT",
      "dependencies": {
        "@babel/helper-annotate-as-pure": "^7.25.9",
        "regexpu-core": "^6.1.1",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0"
      }
    },
```

```
                  "node_modules/@babel/helper-create-regexp-features-
plugin/node_modules/semver": {
      "version": "6.3.1",
           "resolved":   "https://registry.npmjs.org/semver/-/semver-
6.3.1.tgz",
                                        "integrity":           "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9
yArRkyrQxTO6XjMzA==",
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }



    "node_modules/workbox-google-analytics": {
      "version": "4.3.1",
            "resolved":   "https://registry.npmjs.org/workbox-google-
analytics/-/workbox-google-analytics-4.3.1.tgz",
                                        "integrity":           "sha512-
xzCjAoKuOb55CBSwQrbyWBKqp35yg1vw9ohIlU2wTy06ZrYfJ8rKochb1MSGlnoBfXGWs
s3UPzxR5QL5guIFdg==",
       "deprecated": "It is not compatible with newer versions of GA
starting with v4, as long as you are using GAv3 it should be ok, but
the package is not longer being maintained",
      "license": "MIT",
      "dependencies": {
        "workbox-background-sync": "^4.3.1",
        "workbox-core": "^4.3.1",
        "workbox-routing": "^4.3.1",
        "workbox-strategies": "^4.3.1"
      }
    },
    "node_modules/workbox-navigation-preload": {
      "version": "4.3.1",
         "resolved":  "https://registry.npmjs.org/workbox-navigation-
preload/-/workbox-navigation-preload-4.3.1.tgz",
                                        "integrity":           "sha512-
K076n3oFHYp16/C+F8CwrRqD25GitA6Rkd6+qAmLmMv1QHPI2jfDwYqrytOfKfYq42bYt
W8Pr21ejZX7GvALOw==",
      "license": "MIT",
      "dependencies": {
        "workbox-core": "^4.3.1"
      }
    },
    "node_modules/workbox-precaching": {
      "version": "4.3.1",
```

```
        "resolved":  "https://registry.npmjs.org/workbox-precaching/-
/workbox-precaching-4.3.1.tgz",
                                        "integrity":           "sha512-
piSg/2csPoIi/vPpp48t1q5JLYjMkmg5gsXBQkh/QYapCdVwwmKlU9mHdmy52KsDGIjVa
qEUMFvEzn2LRaigqQ==",
      "license": "MIT",
      "dependencies": {
        "workbox-core": "^4.3.1"
      }
    },
    "node_modules/workbox-range-requests": {
      "version": "4.3.1",
     "resolved": "https://registry.npmjs.org/workbox-range-requests/-
/workbox-range-requests-4.3.1.tgz",
                                        "integrity":           "sha512-
S+HhL9+iTFypJZ/yQSl/x2Bf5pWnbXdd3j57xnb0V60FW1LVn9LRZkPtneODklzYuFZv7
qK6riZ5BNyc0R0jZA==",
      "license": "MIT",
      "dependencies": {
        "workbox-core": "^4.3.1"
      }
    },
    "node_modules/workbox-routing": {
      "version": "4.3.1",
          "resolved":   "https://registry.npmjs.org/workbox-routing/-
/workbox-routing-4.3.1.tgz",
                                        "integrity":           "sha512-
FkbtrODA4Imsi0p7TW9u9MXuQ5P4pVs1sWHK4dJMMChVROsbEltuE79fBoIk/BCztvOJ7
yUpErMKa4z3uQLX+g==",
      "license": "MIT",
      "dependencies": {
        "workbox-core": "^4.3.1"
      }
    },
    "node_modules/workbox-strategies": {
      "version": "4.3.1",
        "resolved":  "https://registry.npmjs.org/workbox-strategies/-
/workbox-strategies-4.3.1.tgz",
                                        "integrity":           "sha512-
F/+E57BmVG8dX6dCCopBlkDvvhg/zj6VDs0PigYwSN23L8hseSRwljrceU2WzTvk/+BSY
ICsWmRq5qHS2UYzhw==",
      "license": "MIT",
      "dependencies": {
        "workbox-core": "^4.3.1"
      }
    },
    "node_modules/workbox-streams": {
      "version": "4.3.1",
```

```
          "resolved":   "https://registry.npmjs.org/workbox-streams/-
/workbox-streams-4.3.1.tgz",
                                        "integrity":           "sha512-
4Kisis1f/y0ihf4l3u/+ndMkJkIT4/6UOacU3A4BwZSAC9pQ9vSvJpIi/WFGQRH/uPXvu
VjF5c2RfIPQFSS2uA==",
      "license": "MIT",
      "dependencies": {
        "workbox-core": "^4.3.1"
      }
    },
    "node_modules/workbox-sw": {
      "version": "4.3.1",
     "resolved": "https://registry.npmjs.org/workbox-sw/-/workbox-sw-
4.3.1.tgz",
                                        "integrity":           "sha512-
0jXdusCL2uC5gM3yYFT6QMBzKfBr2XTk0g5TPAV4y8IZDyVNDyj1a8uSXy3/XrvkVTmQv
LN4O5k3JawGReXr9w==",
      "license": "MIT"
    },
    "node_modules/workbox-webpack-plugin": {
      "version": "4.3.1",
     "resolved": "https://registry.npmjs.org/workbox-webpack-plugin/-
/workbox-webpack-plugin-4.3.1.tgz",
                                        "integrity":           "sha512-
gJ9jd8Mb8wHLbRz9ZvGN57IAmknOipD3W4XNE/Lk/4lqs5Htw4WOQgakQy/o/4CoXQlMC
YldaqUg+EJ35l9MEQ==",
      "license": "MIT",
      "dependencies": {
        "@babel/runtime": "^7.0.0",
        "json-stable-stringify": "^1.0.1",
        "workbox-build": "^4.3.1"
      },
      "engines": {
        "node": ">=4.0.0"
      },
      "peerDependencies": {
        "webpack": "^2.0.0 || ^3.0.0 || ^4.0.0"
      }
    },
    "node_modules/workbox-window": {
      "version": "4.3.1",
           "resolved":   "https://registry.npmjs.org/workbox-window/-
/workbox-window-4.3.1.tgz",
                                        "integrity":           "sha512-
C5gWKh6I58w3GeSc0wp2Ne+rqVw8qwcmZnQGpjiek8A2wpbxSJb1FdCoQVO+jDJs35bFg
o/WETgl1fqgsxN0Hg==",
      "license": "MIT",
      "dependencies": {
```

```
        "workbox-core": "^4.3.1"
      }
    },
    "node_modules/worker-farm": {
      "version": "1.7.0",
        "resolved":  "https://registry.npmjs.org/worker-farm/-/worker-
farm-1.7.0.tgz",
                                          "integrity":           "sha512-
rvw3QTZc8lAxyVrqcSGVm5yP/IJ2UcB3U0graE3LCFoZ0Yn2x4EoVSqJKdB/T5M+FLcRP
jz4TDacRf3OCfNUzw==",
      "license": "MIT",
      "dependencies": {
        "errno": "~0.1.7"
      }
    },
    "node_modules/worker-rpc": {
      "version": "0.1.1",
     "resolved": "https://registry.npmjs.org/worker-rpc/-/worker-rpc-
0.1.1.tgz",
                                          "integrity":           "sha512-
P1WjMrUB3qgJNI9jfmpZ/htmBEjFh//6l/5y8SD9hg1Ef5zTTVVoRjTrTEzPrNBQvmhMx
koTsjOXN10GWU7aCg==",
      "license": "MIT",
      "dependencies": {
        "microevent.ts": "~0.1.1"
      }
    },
    "node_modules/wrap-ansi": {
      "version": "5.1.0",
       "resolved":  "https://registry.npmjs.org/wrap-ansi/-/wrap-ansi-
5.1.0.tgz",
                                          "integrity":           "sha512-
QC1/iN/2/RPVJ5jYK8BGttj5z83LmSKmvbvrXPNCLZSEb32KKVDJDl/MOt2N01qU2H/Fk
zEa9PKto1BqDjtd7Q==",
      "license": "MIT",
      "dependencies": {
        "ansi-styles": "^3.2.0",
        "string-width": "^3.0.0",
        "strip-ansi": "^5.0.0"
      },
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/wrap-ansi/node_modules/is-fullwidth-code-point": {
      "version": "2.0.0",
          "resolved":  "https://registry.npmjs.org/is-fullwidth-code-
point/-/is-fullwidth-code-point-2.0.0.tgz",
```

```
                                        "integrity":           "sha512-
VHskAKYM8RfSFXwee5t5cbN5PZeq1Wrh6qd5bkyiXIf6UQcN6w/A0eXM9r6t8d+GYOh+o
6ZhiEnb88LN/Y8m2w==",
      "license": "MIT",
      "engines": {
        "node": ">=4"
      }
    },
    "node_modules/wrap-ansi/node_modules/string-width": {
      "version": "3.1.0",
       "resolved": "https://registry.npmjs.org/string-width/-/string-
width-3.1.0.tgz",
                                        "integrity":           "sha512-
vafcv6KjVZKSgz06oM/H6GDBrAtz8vdhQakGjFIvNrHA6y3HCF1CInLy+QLq8dTJPQ1b+
KDUqDFctkdRW44e1w==",
      "license": "MIT",
      "dependencies": {
        "emoji-regex": "^7.0.1",
        "is-fullwidth-code-point": "^2.0.0",
        "strip-ansi": "^5.1.0"
      },
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/wrappy": {
      "version": "1.0.2",
           "resolved":   "https://registry.npmjs.org/wrappy/-/wrappy-
1.0.2.tgz",
                                        "integrity":           "sha512-
l4Sp/DRseor9wL6EvV2+TuQn63dMkPjZ/sp9XkghTEbV9KlPS1xUsZ3u7/IQO4wxtcFB4
bgpQPRcR3QCvezPcQ==",
      "license": "ISC"
    },
    "node_modules/write": {
      "version": "1.0.3",
            "resolved":   "https://registry.npmjs.org/write/-/write-
1.0.3.tgz",
                                        "integrity":           "sha512-
/lg70HAjtkUgWPVZhZcm+T4hkL8Zbtp1nFNOn3lRrxnlv50SRBv7cR7RqR+GMsd3hUXy9
hWBo4CHTbFTcOYwig==",
      "license": "MIT",
      "dependencies": {
        "mkdirp": "^0.5.1"
      },
      "engines": {
        "node": ">=4"
      }
```

```
    },
    "node_modules/write-file-atomic": {
      "version": "2.4.1",
         "resolved":  "https://registry.npmjs.org/write-file-atomic/-
/write-file-atomic-2.4.1.tgz",
                                        "integrity":           "sha512-
TGHFeZEZMnv+gBFRfjAcxL5bPHrsGKtnb4qsFAws7/vlh+QfwAaySIw4AXP9ZskTTh5GW
u3FLuJhsWVdiJPGvg==",
      "license": "ISC",
      "dependencies": {
        "graceful-fs": "^4.1.11",
        "imurmurhash": "^0.1.4",
        "signal-exit": "^3.0.2"
      }
    },
    "node_modules/ws": {
      "version": "5.2.4",
      "resolved": "https://registry.npmjs.org/ws/-/ws-5.2.4.tgz",
                                        "integrity":           "sha512-
fFCejsuC8f9kOSu9FYaOw8CdO68O3h5v0lg4p74o8JqWpwTf9tniOD+nOB78aWoVSS6Wp
tVUmDrp/KPsMVBWFQ==",
      "license": "MIT",
      "dependencies": {
        "async-limiter": "~1.0.0"
      }
    },
    "node_modules/xml-name-validator": {
      "version": "3.0.0",
         "resolved":  "https://registry.npmjs.org/xml-name-validator/-
/xml-name-validator-3.0.0.tgz",
                                        "integrity":           "sha512-
A5CUptxDsvxKJEU3yO6DuWBSJz/qizqzJKOMIfUJHETbBw/sFaDxgd6fxm1ewUaM0jZ44
4Fc5vC5ROYurg/4Pw==",
      "license": "Apache-2.0"
    },
    "node_modules/xmlchars": {
      "version": "2.2.0",
         "resolved":  "https://registry.npmjs.org/xmlchars/-/xmlchars-
2.2.0.tgz",
                                        "integrity":           "sha512-
JZnDKK8B0RCDw84FNdDAIpZK+JuJw+s7Lz8nksI7SIuU3UXJJslUthsi+uWBUYOwPFwW7
W7PRLRfUKpxjtjFCw==",
      "license": "MIT"
    },
    "node_modules/xregexp": {
      "version": "4.4.1",
          "resolved":  "https://registry.npmjs.org/xregexp/-/xregexp-
4.4.1.tgz",
```

```
                                        "integrity":           "sha512-
2u9HwfadaJaY9zHtRRnH6BY6CQVNQKkYm3oLtC9gJXXzfsbACg5X5e4EZZGVAH+YIfa+Q
A9lsFQTTe3HURF3ag==",
      "license": "MIT",
      "dependencies": {
        "@babel/runtime-corejs3": "^7.12.1"
      }
    },
    "node_modules/xtend": {
      "version": "4.0.2",
            "resolved":   "https://registry.npmjs.org/xtend/-/xtend-
4.0.2.tgz",
                                        "integrity":           "sha512-
LKYU1iAXJXUgAXn9URjiu+MWhyUXHsvfp7mcuYm9dSUKK0/CjtrUwFAxD82/mCWbtLsGj
FIad0wIsod4zrTAEQ==",
      "license": "MIT",
      "engines": {
        "node": ">=0.4"
      }
    },
    "node_modules/y18n": {
      "version": "4.0.3",
      "resolved": "https://registry.npmjs.org/y18n/-/y18n-4.0.3.tgz",
                                        "integrity":           "sha512-
JKhqTOwSrqNA1NY5lSztJ1GrBiUodLMmIZuLiDaMRJ+itFd+ABVE8XBjOvIWL+rSqNDC7
4LCSFmlb/U4UZ4hJQ==",
      "license": "ISC"
    },
    "node_modules/yallist": {
      "version": "3.1.1",
          "resolved":  "https://registry.npmjs.org/yallist/-/yallist-
3.1.1.tgz",
                                        "integrity":           "sha512-
a4UGQaWPH59mOXUYnAG2ewncQS4i4F43Tv3JoAM+s2VDAmS9NsK8GpDMLrCHPksFT7h3K
6TOoUNn2pb7RoXx4g==",
      "license": "ISC"
    },
    "node_modules/yaml": {
      "version": "1.10.2",
     "resolved": "https://registry.npmjs.org/yaml/-/yaml-1.10.2.tgz",
                                        "integrity":           "sha512-
r3vXyErRCYJ7wg28yvBY5VSoAF8ZvlcW9/BwUzEtUsjvX/DKs24dIkuwjtuprwJJHsbyU
bLApepYTR1BN4uHrg==",
      "license": "ISC",
      "engines": {
        "node": ">= 6"
      }
    },
```

```
    "node_modules/yargs": {
      "version": "13.3.2",
            "resolved":   "https://registry.npmjs.org/yargs/-/yargs-
13.3.2.tgz",
                                        "integrity":          "sha512-
AX3Zw5iPruN5ie6xGRIDgqkT+ZhnRlZMLMHAs8tg7nRruy2Nb+i5o9bwghAogtM08q1dp
r2LVoS8KSTMYpWXUw==",
      "license": "MIT",
      "dependencies": {
        "cliui": "^5.0.0",
        "find-up": "^3.0.0",
        "get-caller-file": "^2.0.1",
        "require-directory": "^2.1.1",
        "require-main-filename": "^2.0.0",
        "set-blocking": "^2.0.0",
        "string-width": "^3.0.0",
        "which-module": "^2.0.0",
        "y18n": "^4.0.0",
        "yargs-parser": "^13.1.2"
      }
    },
    "node_modules/yargs-parser": {
      "version": "13.1.2",
        "resolved":  "https://registry.npmjs.org/yargs-parser/-/yargs-
parser-13.1.2.tgz",
                                        "integrity":          "sha512-
3lbsNRf/j+A4QuSZfDRA7HRSfWrzO0YjqTJd5kjAq37Zep1CEgaYmrH9Q3GwPiB9cHyd1
Y1UwggGhJGoxipbzg==",
      "license": "ISC",
      "dependencies": {
        "camelcase": "^5.0.0",
        "decamelize": "^1.2.0"
      }
    },
    "node_modules/yargs/node_modules/is-fullwidth-code-point": {
      "version": "2.0.0",
          "resolved":  "https://registry.npmjs.org/is-fullwidth-code-
point/-/is-fullwidth-code-point-2.0.0.tgz",
                                        "integrity":          "sha512-
VHskAKYM8RfSFXwee5t5cbN5PZeq1Wrh6qd5bkyiXIf6UQcN6w/A0eXM9r6t8d+GYOh+o
6ZhiEnb88LN/Y8m2w==",
      "license": "MIT",
      "engines": {
        "node": ">=4"
      }
    },
    "node_modules/yargs/node_modules/string-width": {
      "version": "3.1.0",
```

```
      "resolved": "https://registry.npmjs.org/string-width/-/string-
width-3.1.0.tgz",
                                        "integrity":          "sha512-
vafcv6KjVZKSgz06oM/H6GDBrAtz8vdhQakGjFIvNrHA6y3HCF1CInLy+QLq8dTJPQ1b+
KDUqDFctkdRW44e1w==",
      "license": "MIT",
      "dependencies": {
        "emoji-regex": "^7.0.1",
        "is-fullwidth-code-point": "^2.0.0",
        "strip-ansi": "^5.1.0"
      },
      "engines": {
        "node": ">=6"
      }
    }
  }
}
```

Node modules:

In a Node.js project, the node_modules directory is crucial because it contains all the dependencies (packages) your project requires to run. These dependencies are specified in the package.json file, and when you run npm install, Node.js downloads and installs these dependencies into the node_modules folder.

Index.js:

```
import express from 'express';
import bodyParser from 'body-parser';
import mongoose from 'mongoose';
import cors from 'cors';
import bcrypt from 'bcrypt';
import { User, Booking, Flight } from './schemas.js';

const app = express();

app.use(express.json());
app.use(bodyParser.json({limit: "30mb", extended: true}))
app.use(bodyParser.urlencoded({limit: "30mb", extended: true}));
app.use(cors());

// mongoose setup

const PORT = 6001;
mongoose.connect('mongodb://localhost:27017/FlightBookingMERN', {
        useNewUrlParser: true,
```

```
        useUnifiedTopology: true,
    }
).then(()=>{

    // All the client-server activites


    app.post('/register', async (req, res) => {
        const { username, email, usertype, password } = req.body;
        let approval = 'approved';
        try {

            const existingUser = await User.findOne({ email });
            if (existingUser) {
                 return res.status(400).json({ message: 'User already
exists' });
            }

            if(usertype === 'flight-operator'){
                approval = 'not-approved'
            }

            const hashedPassword = await bcrypt.hash(password, 10);
            const newUser = new User({
                 username, email, usertype, password: hashedPassword,
approval
            });
            const userCreated = await newUser.save();
            return res.status(201).json(userCreated);

        } catch (error) {
          console.log(error);
          return res.status(500).json({ message: 'Server Error' });
        }
    });

    app.post('/login', async (req, res) => {
        const { email, password } = req.body;
        try {

            const user = await User.findOne({ email });

            if (!user) {
                return res.status(401).json({ message: 'Invalid email
or password' });
            }
                   const  isMatch  =  await  bcrypt.compare(password,
user.password);
            if (!isMatch) {
```

```
                return res.status(401).json({ message: 'Invalid email
or password' });
            } else{

                return res.json(user);
            }

        } catch (error) {
          console.log(error);
          return res.status(500).json({ message: 'Server Error' });
        }
    });


    // Approve flight operator

    app.post('/approve-operator', async(req, res)=>{
        const {id} = req.body;
        try{

            const user = await User.findById(id);
            user.approval = 'approved';
            await user.save();
            res.json({message: 'approved!'})
        }catch(err){
            res.status(500).json({ message: 'Server Error' });
        }
    })

    // reject flight operator

    app.post('/reject-operator', async(req, res)=>{
        const {id} = req.body;
        try{

            const user = await User.findById(id);
            user.approval = 'rejected';
            await user.save();
            res.json({message: 'rejected!'})
        }catch(err){
            res.status(500).json({ message: 'Server Error' });
        }
    })


    // fetch user

    app.get('/fetch-user/:id', async (req, res)=>{
        const id = await req.params.id;
```

```
        console.log(req.params.id)
        try{
            const user = await User.findById(req.params.id);
            console.log(user);
            res.json(user);

        }catch(err){
            console.log(err);
        }
    })

    // fetch all users

    app.get('/fetch-users', async (req, res)=>{

        try{
            const users = await User.find();
            res.json(users);

        }catch(err){
            res.status(500).json({message: 'error occured'});
        }
    })


    // Add flight

    app.post('/add-flight', async (req, res)=>{
              const  {flightName,  flightId,  origin,  destination,
departureTime,
                                  arrivalTime, basePrice, totalSeats} =
req.body;
        try{

             const flight = new Flight({flightName, flightId, origin,
destination,
                                              departureTime, arrivalTime,
basePrice, totalSeats});
            const newFlight = flight.save();

            res.json({message: 'flight added'});

        }catch(err){
            console.log(err);
        }
    })

    // update flight
```

```
    app.put('/update-flight', async (req, res)=>{
        const {_id, flightName, flightId, origin, destination,
                    departureTime, arrivalTime, basePrice, totalSeats}
= req.body;
        try{

            const flight = await Flight.findById(_id)

            flight.flightName = flightName;
            flight.flightId = flightId;
            flight.origin = origin;
            flight.destination = destination;
            flight.departureTime = departureTime;
            flight.arrivalTime = arrivalTime;
            flight.basePrice = basePrice;
            flight.totalSeats = totalSeats;

            const newFlight = flight.save();

            res.json({message: 'flight updated'});

        }catch(err){
            console.log(err);
        }
    })

    // fetch flights

    app.get('/fetch-flights', async (req, res)=>{

        try{
            const flights = await Flight.find();
            res.json(flights);

        }catch(err){
            console.log(err);
        }
    })


    // fetch flight

    app.get('/fetch-flight/:id', async (req, res)=>{
        const id = await req.params.id;
        console.log(req.params.id)
        try{
            const flight = await Flight.findById(req.params.id);
            console.log(flight);
            res.json(flight);
```

```
        }catch(err){
            console.log(err);
        }
    })

    // fetch all bookings

    app.get('/fetch-bookings', async (req, res)=>{

        try{
            const bookings = await Booking.find();
            res.json(bookings);

        }catch(err){
            console.log(err);
        }
    })

    // Book ticket

    app.post('/book-ticket', async (req, res)=>{
            const {user, flight, flightName, flightId,  departure,
destination,
                    email, mobile, passengers, totalPrice, journeyDate,
journeyTime, seatClass} = req.body;
        try{
                const bookings = await Booking.find({flight: flight,
journeyDate: journeyDate, seatClass: seatClass});
             const numBookedSeats = bookings.reduce((acc, booking) =>
acc + booking.passengers.length, 0);

            let seats = "";
            const seatCode = {'economy': 'E', 'premium-economy': 'P',
'business': 'B', 'first-class': 'A'};
            let coach = seatCode[seatClass];
               for(let i = numBookedSeats + 1; i< numBookedSeats +
passengers.length+1; i++){
                if(seats === ""){
                    seats = seats.concat(coach, '-', i);
                }else{
                    seats = seats.concat(", ", coach, '-', i);
                }
            }
              const booking = new Booking({user, flight, flightName,
flightId, departure, destination,
                                            email, mobile, passengers,
totalPrice, journeyDate, journeyTime, seatClass, seats});
```

```
            await booking.save();

            res.json({message: 'Booking successful!!'});
        }catch(err){
            console.log(err);
        }
    })
```

Explanation:

The server runs on port 4000 and employs middleware like cors for cross-origin requests and express. Json for handling JSON payloads. It also configures multer for file uploads, storing documents like user IDs and travel proofs in the uploads directory. For user authentication, the app provides endpoints for admin and customer login and registration, checking credentials against MongoDB records.

It includes routes for managing users, Admin, Flights, and Bookings, enabling CRUD operations such as fetching flight schedules, adding new flights, and cancelling bookings. Each route handles specific tasks, such as logging in an admin, registering a customer, or uploading passenger details and travel documents. The code ensures seamless integration with the MongoDB database for data storage and retrieval, aiming to provide a robust backend infrastructure for the flight booking application. Enhancements for better security, such as password hashing, JWT-based authentication, and improved error handling, are advisable to ensure scalability and reliability

9.2 Frontend Pages:

Login.jsx:

```
import React, { useContext } from 'react'
import { GeneralContext } from '../context/GeneralContext';

const Login = ({setIsLogin}) => {

  const {setEmail, setPassword, login} = useContext(GeneralContext);

  const handleLogin = async (e) =>{
    e.preventDefault();
    await login();
  }
  return (
```

```
    <form className="authForm">
        <h2>Login</h2>
        <div className="form-floating mb-3 authFormInputs">
                        <input   type="email"   className="form-control"
id="floatingInput" placeholder="name@example.com"
                                                                    onC
hange={(e) => setEmail(e.target.value)} />
            <label htmlFor="floatingInput">Email address</label>
        </div>
            <div className="form-floating mb-3 authFormInputs">
                    <input  type="password"  className="form-control"
id="floatingPassword" placeholder="Password"
                                                                    onC
hange={(e) => setPassword(e.target.value)} />
            <label htmlFor="floatingPassword">Password</label>
        </div>
               <button   type="submit"   className="btn   btn-primary"
onClick={handleLogin}>Sign in</button>

                      <p>Not    registered?    <span    onClick={()=>
setIsLogin(false)}>Register</span></p>
    </form>
  )
}
export default Login
```

NavBar.jsx:

```
import React, { useContext } from 'react'
import '../styles/Navbar.css';
import { useNavigate } from 'react-router-dom';
import { GeneralContext } from '../context/GeneralContext';

const Navbar = () => {

    const navigate = useNavigate();
    const usertype = localStorage.getItem('userType');

    const {logout} = useContext(GeneralContext);

  return (
    <>
        <div className="navbar">

        {!usertype ?
```

```
            <>
                <h3 >Best Flights</h3>

                <div className="nav-options" >
                    <p onClick={()=>navigate('/')}>Home</p>
                    <p onClick={()=>navigate('/auth')}>Login</p>
                </div>

            </>
        :

        <>
        {usertype === 'customer' ?

        <>
            <h3 >Best Flights</h3>

            <div className="nav-options" >

                <p onClick={()=>navigate('/')}>Home</p>
                <p onClick={()=>navigate('/bookings')}>Bookings</p>
                <p onClick={logout}>Logout</p>

            </div>
        </>
            :  usertype === 'admin' ?

                    <>
                        <h3 >Best Flights (Admin)</h3>
                        <div className="nav-options" >

                          <p onClick={()=>navigate('/admin')}>Home</p>
                                      <p  onClick={()=>navigate('/all-
users')}>Users</p>
                                      <p  onClick={()=>navigate('/all-
bookings')}>Bookings</p>
                                      <p  onClick={()=>navigate('/all-
flights')}>Flights</p>
                             <p onClick={logout}>Logout</p>
                        </div>
                    </>

                : usertype === 'flight-operator' ?
                    <>
                        <h3 >Best Flights (Operator)</h3>
                        <div className="nav-options" >
```

```
                                    <p  onClick={()=>navigate('/flight-
admin')}>Home</p>
                                    <p  onClick={()=>navigate('/flight-
bookings')}>Bookings</p>
                                                                        <p
onClick={()=>navigate('/flights')}>Flights</p>
                            <p onClick={()=>navigate('/new-flight')}>Add
Flight</p>
                               <p onClick={logout}>Logout</p>
                           </div>
                     </>

                :

                     ""


        }
        </>
        }
        </div>

    </>
  )
}

export default Navbar
```

Register.jsx:

```
import React, { useContext } from 'react'
import { GeneralContext } from '../context/GeneralContext';

const Register = ({setIsLogin}) => {

  const {setUsername, setEmail, setPassword, usertype, setUsertype,
register, setHomeBranch} = useContext(GeneralContext);

  const handleRegister = async (e) =>{
    e.preventDefault();
    await register()
  }
  return (
    <form className="authForm">
        <h2>Register</h2>
        <div className="form-floating mb-3 authFormInputs">
```

```
                        <input   type="text"   className="form-control"
id="floatingInput" placeholder="username"
                                                              onChange={(e)=>
setUsername(e.target.value)} />
            <label htmlFor="floatingInput">Username</label>
        </div>
        <div className="form-floating mb-3 authFormInputs">
                      <input   type="email"   className="form-control"
id="floatingEmail" placeholder="name@example.com"
                                                              onChange={(e)=>
setEmail(e.target.value)} />
            <label htmlFor="floatingInput">Email address</label>
        </div>
        <div className="form-floating mb-3 authFormInputs">
                    <input  type="password"  className="form-control"
id="floatingPassword" placeholder="Password"
                                                              onChange={(e)=>
setPassword(e.target.value)} />
            <label htmlFor="floatingPassword">Password</label>
        </div>
          <select  className="form-select  form-select-lg  mb-3"  aria-
label=".form-select-lg example"
                                                              onChange={(e)=>
setUsertype(e.target.value)}>
          <option value="">User type</option>
          <option value="admin">Admin</option>
          <option value="customer">Customer</option>
          <option value="flight-operator">Flight Operator</option>
        </select>

                          <button     className="btn     btn-primary"
onClick={handleRegister}>Sign up</button>
                  <p>Already    registered?    <span    onClick={()=>
setIsLogin(true)}>Login</span></p>
    </form>
  )}
export default Register;
```

Admin.jsx:

```
import React, { useEffect, useState } from 'react'
import '../styles/Admin.css'
import { useNavigate } from 'react-router-dom'
import axios from 'axios'

const Admin = () => {
```

```
  const navigate = useNavigate();
  const [users, setUsers] = useState([]);
  const [userCount, setUserCount] = useState(0);
  const [bookingCount, setbookingCount] = useState(0);
  const [flightsCount, setFlightsCount] = useState(0);


  useEffect(()=>{

    fetchData();
  }, [])

  const fetchData = async () =>{
    await axios.get('http://localhost:6001/fetch-users').then(
      (response)=>{

        setUserCount(response.data.length -1);
        setUsers(response.data.filter(user => user.approval === 'not-
approved'));
      }
    );
    await axios.get('http://localhost:6001/fetch-bookings').then(
      (response)=>{
        setbookingCount(response.data.length);
      }
    );
    await axios.get('http://localhost:6001/fetch-flights').then(
      (response)=>{
        setFlightsCount(response.data.length);
      }
    );
  }



  const approveRequest = async (id) =>{
      try{

           await axios.post('http://localhost:6001/approve-operator',
{id}).then(
             (response)=>{
               alert("Operator approved!!");
               fetchData();
             }
           )

      }catch(err){
```

```
      }
  }

  const rejectRequest = async (id) =>{
    try{

           await   axios.post('http://localhost:6001/reject-operator',
{id}).then(
        (response)=>{
          alert("Operator rejected!!");
          fetchData();
        }
      )

    }catch(err){

    }
  }

  return (
    <>

      <div className="admin-page">

        <div className="admin-page-cards">

            <div className="card admin-card users-card">
                <h4>Users</h4>
                <p> {userCount} </p>
                                <button   className="btn   btn-primary"
onClick={()=>navigate('/all-users')}>View all</button>
            </div>

            <div className="card admin-card transactions-card">
                <h4>Bookings</h4>
                <p> {bookingCount} </p>
                                <button   className="btn   btn-primary"
onClick={()=>navigate('/all-bookings')}>View all</button>
            </div>

            <div className="card admin-card deposits-card">
                <h4>Flights</h4>
                <p> {flightsCount} </p>
                                <button   className="btn   btn-primary"
onClick={()=>navigate('/all-flights')}>View all</button>
            </div>


        </div>
```

```
        <div className="admin-requests-container">

            <h3>New Operator Applications</h3>

            <div className="admin-requests">

              {
                users.length === 0 ?
                  <p>No new requests..</p>
                :
                  <>
                  {users.map((user)=>{
                    return(
                      <div className="admin-request" key={user._id}>
                                    <span><b>Operator  name:  </b>
{user.username}</span>
                       <span><b>Operator email: </b> {user.email}</span>
                          <div className="admin-request-actions">
                                   <button  className='btn  btn-primary'
onClick={()=> approveRequest(user._id)}>Approve</button>
                                    <button  className='btn  btn-danger'
onClick={()=> rejectRequest(user._id)}>Reject</button>
                          </div>
                        </div>
                    )
                  })}
                  </>

              }


            </div>

        </div>

    </div>

    </>
  )
}

export default Admin
```

All Booking.jsx:

```
import axios from 'axios';
import React, { useEffect, useState } from 'react'

const AllBookings = () => {

  const [bookings, setBookings] = useState([]);

  const userId = localStorage.getItem('userId');

  useEffect(()=>{
    fetchBookings();
  }, [])

  const fetchBookings = async () =>{
    await axios.get('http://localhost:6001/fetch-bookings').then(
      (response)=>{
        setBookings(response.data.reverse());
      }
    )
  }

  const cancelTicket = async (id) =>{
    await axios.put(`http://localhost:6001/cancel-ticket/${id}`).then(
      (response)=>{
        alert("Ticket cancelled!!");
        fetchBookings();
      }
    )
  }

  return (
    <div className="user-bookingsPage">
      <h1>Bookings</h1>

      <div className="user-bookings">

        {bookings.map((booking)=>{
          return(
            <div className="user-booking" key={booking._id}>
            <p><b>Booking ID:</b> {booking._id}</p>
            <span>
              <p><b>Mobile:</b> {booking.mobile}</p>
              <p><b>Email:</b> {booking.email}</p>
            </span>
            <span>
              <p><b>Flight Id:</b> {booking.flightId}</p>
              <p><b>Flight name:</b> {booking.flightName}</p>
            </span>
```

```
            <span>
              <p><b>On-boarding:</b> {booking.departure}</p>
              <p><b>Destination:</b> {booking.destination}</p>
            </span>
            <span>
              <div>
                <p><b>Passengers:</b></p>
                <ol>
                  {booking.passengers.map((passenger, i)=>{
                    return(
                                          <li   key={i}><p><b>Name:</b>
{passenger.name},  <b>Age:</b> {passenger.age}</p></li>
                    )
                  })}
                </ol>
              </div>
             {booking.bookingStatus === 'confirmed' ? <p><b>Seats:</b>
{booking.seats}</p> : ""}
            </span>
            <span>
                                            <p><b>Booking     date:</b>
{booking.bookingDate.slice(0,10)}</p>
                                            <p><b>Journey     date:</b>
{booking.journeyDate.slice(0,10)}</p>
            </span>
            <span>
              <p><b>Journey Time:</b> {booking.journeyTime}</p>
              <p><b>Total price:</b> {booking.totalPrice}</p>
            </span>
              {booking.bookingStatus === 'cancelled' ?
                    <p  style={{color:  "red"}}><b>Booking  status:</b>
{booking.bookingStatus}</p>
                :
                <p><b>Booking status:</b> {booking.bookingStatus}</p>
              }
            {booking.bookingStatus === 'confirmed' ?
              <div>
                    <button  className="btn  btn-danger"  onClick={()=>
cancelTicket(booking._id)}>Cancel Ticket</button>
              </div>

            :
            <></>}
          </div>
          )
        })}
```

```
      </div>
    </div>
  )
}

export default AllBookings
```

All Flights.jsx:

```
import axios from 'axios';
import React, { useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom';
import '../styles/AllFlights.css';

const AllFlights = () => {
    const [flights, setFlights] = useState([]);
    const navigate = useNavigate();


    const fetchFlights = async () =>{
      await axios.get('http://localhost:6001/fetch-flights').then(
        (response)=>{
          setFlights(response.data);
          console.log(response.data)
        }
        )
      }

      useEffect(()=>{
        fetchFlights();
      }, [])

    return (
      <div className="allFlightsPage">
        <h1>All Flights</h1>

        <div className="allFlights">

          {flights.map((Flight)=>{
            return(

                <div className="allFlights-Flight" key={Flight._id}>
                  <p><b>_id:</b> {Flight._id}</p>
                  <span>
                    <p><b>Flight Id:</b> {Flight.flightId}</p>
```

```
                  <p><b>Flight name:</b> {Flight.flightName}</p>
                </span>
                <span>
                  <p><b>Starting station:</b> {Flight.origin}</p>
                <p><b>Departure time:</b> {Flight.departureTime}</p>
                </span>
                <span>
                  <p><b>Destination:</b> {Flight.destination}</p>
                  <p><b>Arrival time:</b> {Flight.arrivalTime}</p>
                </span>
                <span>
                  <p><b>Base price:</b> {Flight.basePrice}</p>
                  <p><b>Total seats:</b> {Flight.totalSeats}</p>
                </span>
              </div>
            )
          })}




        </div>
      </div>
    )
  }export default AllFlights
```

All Users.jsx:

```
import React, { useEffect, useState } from 'react'
import Navbar from '../components/Navbar'
import '../styles/allUsers.css'
import axios from 'axios';

const AllUsers = () => {

  const [users, setUsers] = useState([]);

  useEffect(()=>{
    fetchUsers();
  },[]);

  const fetchUsers = async () =>{
    await axios.get('http://localhost:6001/fetch-users').then(
      (response) =>{
```

```
        setUsers(response.data);
      }
    )
  }

  return (
    <>
      <Navbar />

      <div class="all-users-page">
        <h2>All Users</h2>
        <div class="all-users">

                        {users.filter(user=>     user.usertype     ===
'customer').map((user)=>{
            return(

              <div class="user" key={user._id}>
                  <p><b>UserId </b>{user._id}</p>
                  <p><b>Username </b>{user.username}</p>
                  <p><b>Email </b>{user.email}</p>
              </div>
            )
          })}

        </div>


        <h2>Flight Operators</h2>
        <div class="all-users">

                {users.filter(user=>   user.usertype   ===   'flight-
operator').map((user)=>{
            return(

              <div class="user" key={user._id}>
                  <p><b>Id </b>{user._id}</p>
                  <p><b>Flight Name </b>{user.username}</p>
                  <p><b>Email </b>{user.email}</p>
              </div>
            )
          })}

        </div>

    </div>
    </>
  )
}
```

```
export default AllUsers
```

Authenticate.jsx:

```
import React, { useState } from 'react';
import '../styles/Authenticate.css'
import Login from '../components/Login';
import Register from '../components/Register';

const Authenticate = () => {

  const [isLogin, setIsLogin] = useState(true);

  return (
    <div className="AuthenticatePage">

      {isLogin ?

      <Login  setIsLogin = {setIsLogin} />

      :

      <Register setIsLogin = {setIsLogin} />
      }

    </div>
  )
}

export default Authenticate
```

Booking Fligfht.jsx:

```
import React, { useContext, useEffect, useState } from 'react'
import '../styles/BookFlight.css'
import { GeneralContext } from '../context/GeneralContext';
import axios from 'axios';
import { useParams, useNavigate } from 'react-router-dom';

const BookFlight = () => {
    const {id} = useParams();

    const [flightName, setFlightName] = useState('');
```

```
    const [flightId, setFlightId] = useState('');
    const [basePrice, setBasePrice] = useState(0);
    const [StartCity, setStartCity] = useState('');
    const [destinationCity, setDestinationCity] = useState('');
    const [startTime, setStartTime] = useState();


    useEffect(()=>{
      fetchFlightData();
    }, [])

    const fetchFlightData = async () =>{
                    await      axios.get(`http://localhost:6001/fetch-
flight/${id}`).then(
        (response) =>{
          setFlightName(response.data.flightName);
          setFlightId(response.data.flightId);
          setBasePrice(response.data.basePrice);
          setStartCity(response.data.origin);
          setDestinationCity(response.data.destination);
          setStartTime(response.data.departureTime);
        }
      )
    }


    const [email, setEmail] = useState('');
    const [mobile, setMobile] = useState('');
    const [coachType, setCoachType] = useState('');
    const {ticketBookingDate} = useContext(GeneralContext);
    const [journeyDate, setJourneyDate] = useState(ticketBookingDate);

    const [numberOfPassengers, setNumberOfPassengers] = useState(0);
    const [passengerDetails, setPassengerDetails] = useState([]);

    const [totalPrice, setTotalPrice] = useState(0);
    const price = {'economy': 1, 'premium-economy': 2, 'business': 3,
'first-class': 4}


    const handlePassengerChange = (event) => {
      const value = parseInt(event.target.value);
      setNumberOfPassengers(value);
      };

    const handlePassengerDetailsChange = (index, key, value) => {
      setPassengerDetails((prevDetails) => {
        const updatedDetails = [...prevDetails];
```

```
          updatedDetails[index] = { ...updatedDetails[index], [key]:
value };
        return updatedDetails;
      });

    };

    useEffect(()=>{
      if(price[coachType] * basePrice * numberOfPassengers){
                  setTotalPrice(price[coachType]    *    basePrice    *
numberOfPassengers);
      }
    },[numberOfPassengers, coachType])


    const navigate = useNavigate();


    const bookFlight = async ()=>{

       const inputs = {user: localStorage.getItem('userId'), flight:
id, flightName,
                                                    flightId, departure:
StartCity, journeyTime: startTime, destination: destinationCity,
                                                          email, mobile,
passengers: passengerDetails, totalPrice,
                                                 journeyDate, seatClass:
coachType}

              await    axios.post('http://localhost:6001/book-ticket',
inputs).then(
        (response)=>{
          alert("booking successful");
          navigate('/bookings');
        }
      ).catch((err)=>{
        alert("Booking failed!!")
      })
    }






    return (
      <div className='BookFlightPage'>

        <div className="BookingFlightPageContainer">
```

```
          <h2>Book ticket</h2>
        <span>
          <p><b>Flight Name: </b> {flightName}</p>
          <p><b>Flight No: </b> {flightId}</p>
        </span>
        <span>
          <p><b>Base price: </b> {basePrice}</p>
        </span>

        <span>
          <div className="form-floating mb-3">
                        <input  type="email"  className="form-control"
id="floatingInputemail"         value={email}         onChange={(e)=>
setEmail(e.target.value)} />
                  <label htmlFor="floatingInputemail">Email</label>
            </div>
            <div className="form-floating mb-3">
                         <input  type="text"  className="form-control"
id="floatingInputmobile"        value={mobile}        onChange={(e)=>
setMobile(e.target.value)} />
                  <label htmlFor="floatingInputmobile">Mobile</label>
            </div>
        </span>
        <span className='span3'>
          <div className="no-of-passengers">
            <div className="form-floating mb-3">
                        <input  type="number"  className="form-control"
id="floatingInputreturnDate"               value={numberOfPassengers}
onChange={handlePassengerChange} />
                       <label  htmlFor="floatingInputreturnDate">No  of
passengers</label>
            </div>
          </div>
          <div className="form-floating mb-3">
                         <input  type="date"  className="form-control"
id="floatingInputreturnDate"                        value={journeyDate}
onChange={(e)=>setJourneyDate(e.target.value)} />
                      <label  htmlFor="floatingInputreturnDate">Journey
date</label>
          </div>
          <div className="form-floating">
                         <select className="form-select form-select-sm
mb-3"     defaultValue=""     aria-label=".form-select-sm     example"
value={coachType} onChange={(e) => setCoachType(e.target.value) }>
                           <option value="" disabled>Select</option>
                        <option value="economy">Economy class</option>
                              <option  value="premium-economy">Premium
Economy</option>
```

```
                                    <option  value="business">Business
class</option>
                                    <option  value="first-class">First
class</option>
                        </select>
                                <label  htmlFor="floatingSelect">Seat
Class</label>
                    </div>

        </span>

        <div className="new-passengers">
            {Array.from({ length: numberOfPassengers }).map((_, index)
=> (
              <div className='new-passenger' key={index}>
                <h4>Passenger {index + 1}</h4>
                <div className="new-passenger-inputs">

                    <div className="form-floating mb-3">
                         <input  type="text"  className="form-control"
id="floatingInputpassengerName"   value={passengerDetails[index]?.name
||   ''}   onChange={(event)   =>   handlePassengerDetailsChange(index,
'name', event.target.value) } />
                                                                    <label
htmlFor="floatingInputpassengerName">Name</label>
                    </div>
                    <div className="form-floating mb-3">
                        <input type="number" className="form-control"
id="floatingInputpassengerAge" value={passengerDetails[index]?.age ||
''} onChange={(event) => handlePassengerDetailsChange(index, 'age',
event.target.value) } />
                                                                    <label
htmlFor="floatingInputpassengerAge">Age</label>
                    </div>


                </div>
              </div>
            ))}

        </div>

        <h6><b>Total price</b>: {totalPrice}</h6>
        <button className='btn btn-primary' onClick={bookFlight}>Book
now</button>
      </div>
      </div>
    )
```

10.API DOCUMENTATION

10.1 Flight Management Endpoints:

- **Add a Flight:**
  - **Method:** POST /flights
  - **Description:** Adds a new flight to the inventory.

- **View All Flights:**
  - **Method:** GET /flights
  - **Description:** Retrieves a list of all available flights.

- **View a Specific Flight:**
  - **Method:** GET /flights/{id}
  - **Description:** Retrieves the details of a specific flight by its ID.

- **Update a Flight:**
  - **Method:** PUT /flights/{id}
  - **Description:** Updates the details of a specific flight.

- **Delete a Flight:**
  - **Method:** DELETE /flights/{id}
  - **Description:** Removes a flight from the inventory.

10.2 User Management Endpoints:

- **Register a User:**
  - **Method:** POST /users/register
  - **Description:** Registers a new user in the system.

- **Login a User:**
  - **Method:** POST /users/login
  - **Description:** Authenticates a user and generates an authentication token.

- **Get User Details:**
  - **Method:** GET /users/{id}
  - **Description:** Retrieves details of a specific user by ID.

- **Update User Details:**
  - **Method:** PUT /users/{id}
  - **Description:** Updates the profile details of a specific user.

- **Delete a User:**
  - **Method:** DELETE /users/{id}
  - **Description:** Removes a user from the system.

10.3 Booking Management Endpoints:

- **Create a Booking:**
  - **Method:** POST /bookings
  - **Description:** Creates a new booking for a flight.

- **View All Bookings:**
  - **Method:** GET /bookings
  - **Description:** Retrieves a list of all bookings made by all users (admin access).

- **View a Single Booking:**
  - **Method:** GET /bookings/{id}
  - **Description:** Retrieves the details of a specific booking by its ID.

- **Update a Booking:**
  - **Method:** PUT /bookings/{id}
  - **Description:** Updates the details of a specific booking.

- **Delete a Booking:**
  - **Method:** DELETE /bookings/{id}
  - **Description:** Removes a booking from the system.

10.4 Payment Management Endpoints

- **Initiate a Payment:**
  - **Method:** POST /payments
  - **Description:** Initiates a payment process for a booking.

- **Verify Payment Status:**
  - **Method:** GET /payments/{id}
  - **Description:** Verifies the status of a specific payment by its ID.

11.AUTHENTICATION

11.1User Registration:

- User sends a request to register with their details (name, email, password, etc.).
- The system validates the input data.
- If validation passes, a new user account is created.
- The system responds with a success message and user ID.

11.2 User Login:

- User sends login credentials (email and password).
- The system verifies the credentials.
- If valid, a token (JWT or similar) is generated and returned.
- User uses the token for authenticated requests.

11.3 Get User Profile:

- User sends a request with the authentication token.
- The system verifies the token and retrieves user details.
- The system returns the user's profile information.

11.4 Admin API Steps:

1. Admin Login:

- Admin provides credentials to log in.
- The system verifies the credentials.
- If valid, a token is issued for admin-level access.

2. View All Users:

- Admin sends a request with their authentication token.
- The system verifies the token and fetches all user records.
- The system returns the list of users.

3. Delete a User:

- Admin sends a request to delete a specific user, providing the user ID.
- The system verifies the admin token and checks if the user exists.
- If valid, the user is deleted, and the system responds with a confirmation.

12.AUTHENTICATION MECHANISM

Authentication for the flight ticket booking application is implemented using **JSON Web Tokens (JWT)** to ensure secure, scalable, and efficient access control.

User Registration:

- **Endpoint:** POST /register
- **Process:** Users provide necessary details, such as username, email, and password, to create an account.
- **Security Measures:** Passwords are hashed using a robust algorithm (e.g., bcrypt) before storage to enhance security and safeguard sensitive data in the event of a database breach.

User Login:

- **Endpoint:** POST /login
- **Process:** Users submit their email and password for authentication.
- **Verification:** The server validates the submitted email against the database and verifies the hashed password using secure comparison methods.
- **Token Generation:** Upon successful verification, the server issues a JWT containing:
    - **User ID:** Identifies the authenticated user.
    - **Role:** Specifies user access levels (e.g., admin, customer).
    - **Expiration Time:** Defines the token's validity period for enhanced security.

Token-Based Authentication

- **JWT Details:** The token encapsulates user information and is cryptographically signed using a secret key, ensuring integrity and preventing tampering.
- **Storage Options:**
    - **Secure Cookies:** Recommended for added protection (e.g., HTTP Only, Secure flags).
    - **Local Storage:** An alternative for less sensitive implementations.
- **Authorization Mechanism:** The client includes the JWT in the Authorization header for all protected endpoints:

Middleware Validation

- **Token Validation:** Middleware intercepts requests to protected resources and verifies the token's authenticity using the secret key.
    - Expired or malformed tokens are rejected with a 401 Unauthorized response.

13.AUTHORIZATION

Authorization in the flight ticket booking application is implemented using **Role-Based Access Control (RBAC)** to assign permissions based on user roles. This approach ensures secure and structured access to resources and actions, improving data management and overall security.

13.1 Key Components:

1. User Roles: Roles define the scope of access and operations that a user can perform:

- **Common Roles:**
  - **Admin:**
    - Full access to all resources and management capabilities.
    - Examples: Managing users, flights, and bookings.
  - **Seller (e.g., Airline Partner):**
    - Limited access to manage their own flight inventory.
    - Examples: Adding or updating flight details, viewing bookings for their flights.
  - **User (Customer):**
    - Access to view and book flights, as well as manage their own bookings.

- **Role-Based Access Control (RBAC):**
  - Administered permissions restrict actions to specific roles to maintain order and security.
  - Example:
    - Admin can delete users or flights. Sellers can manage flights they own but cannot access user data.
    - Users can only book flights or view their booking history.

13.2 Implementation Overview:

1. **Defining Roles and Permissions:**
   - Clearly outline the different roles (Admin, Operator, User) and their respective permissions.
2. **Assigning Roles to Users:**
   - When a user registers or is created, assign them a role based on their function within the application.
   - Store this role information securely in the user's profile.

14.DEMO LINK

Drive Link:
https://drive.google.com/file/d/1AWJwnbxvy4RvajaYTbVqtQIndOXz5lFf/view?usp=sharing

GitHub Link:
https://github.com/Suriyaprak/Flight-Ticket-Booking---Project.git

15.USER INTERFACE AND SCREENSHOT

Home Page

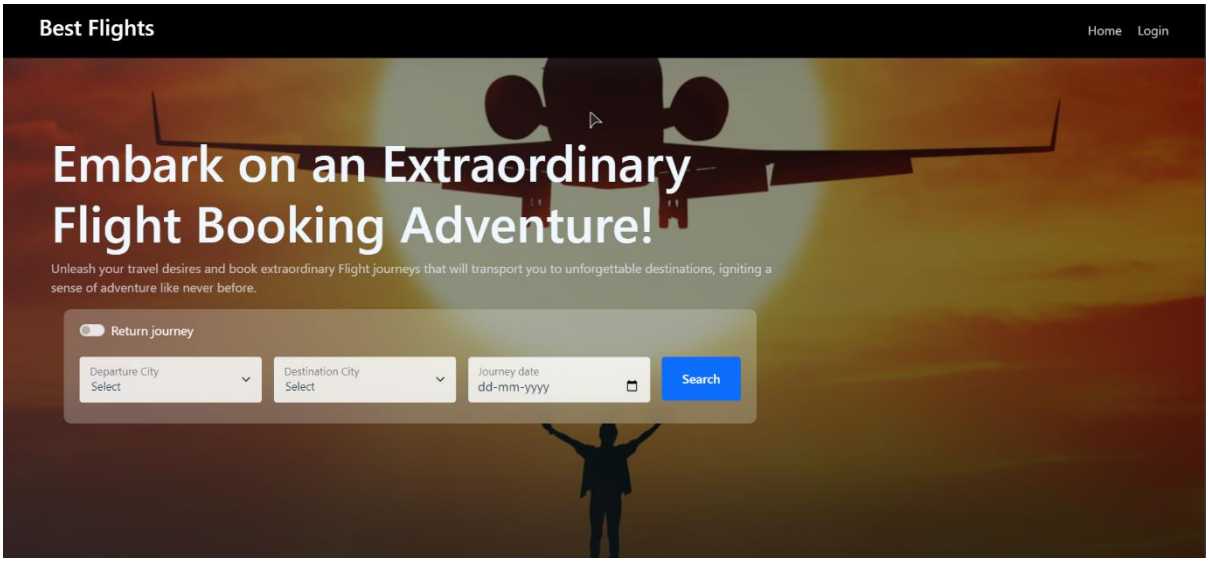

Fig 15.1 Home page

User Login Page

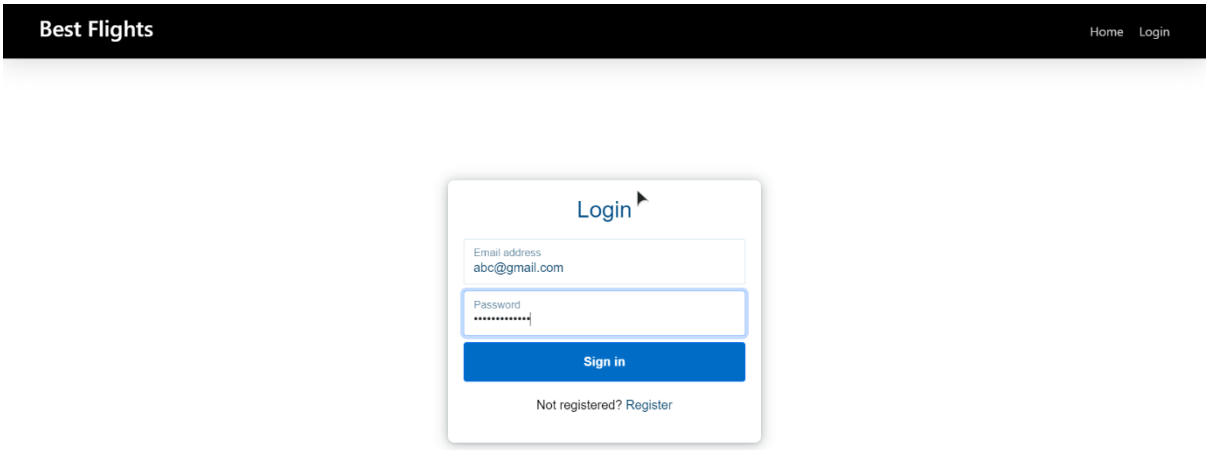

Fig 15.2 User Login page

User Register Page

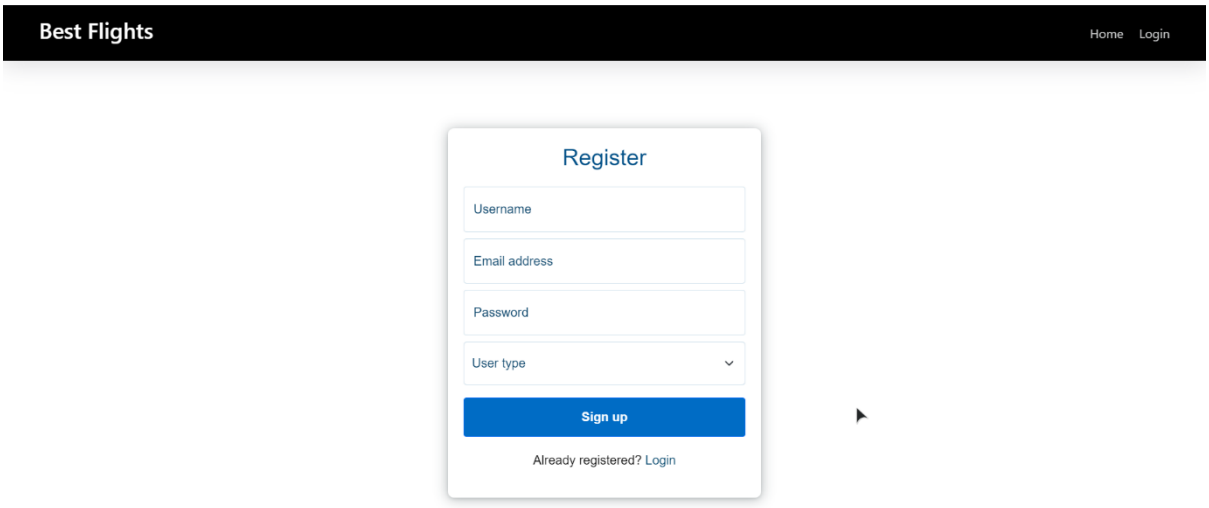

Fig 15.3 Registration page

Available Flights Page

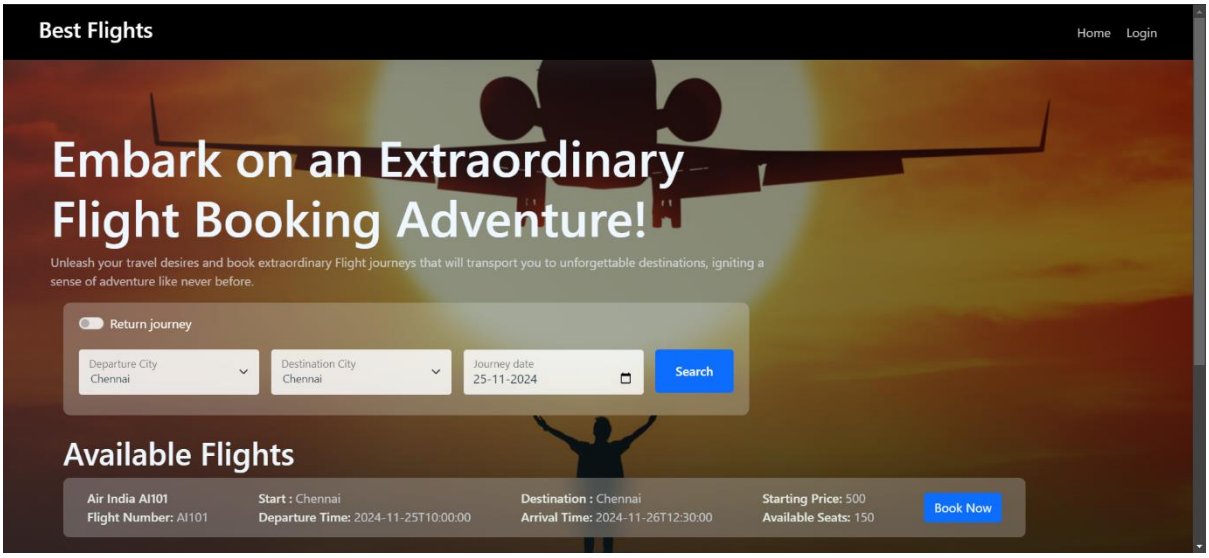

Fig 15.4 Flights page

Ticket Booking Page

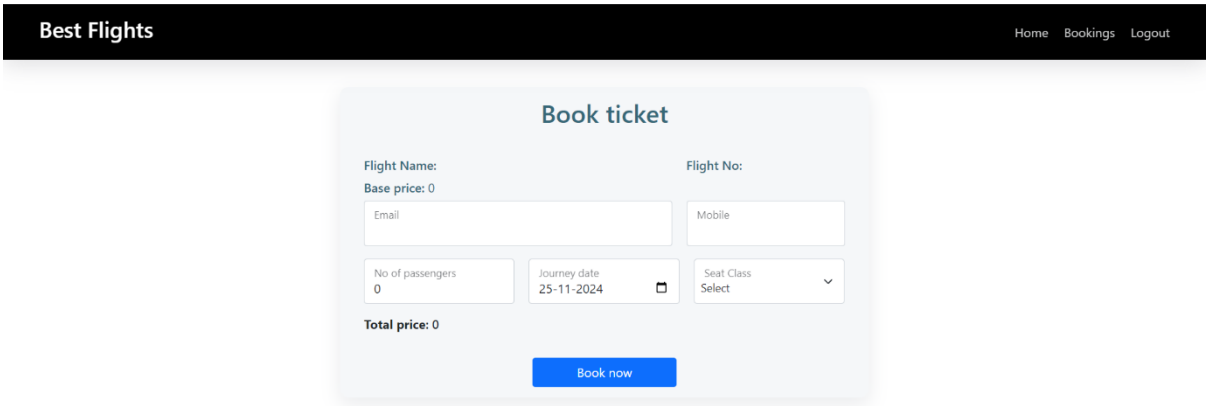

Fig 15.5 Booking page

Flight Operator Home Page

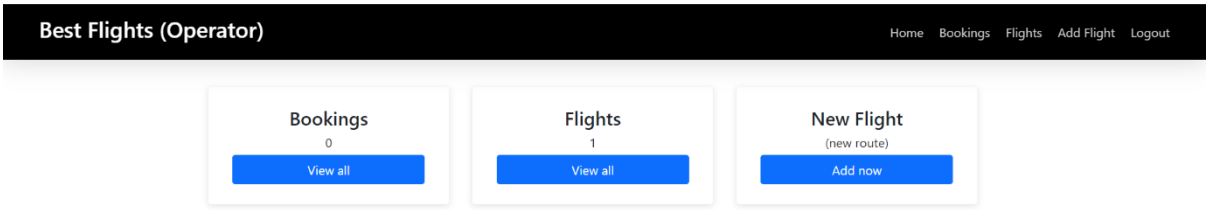

Fig 15.6 Operator Home page

Add Flights Page

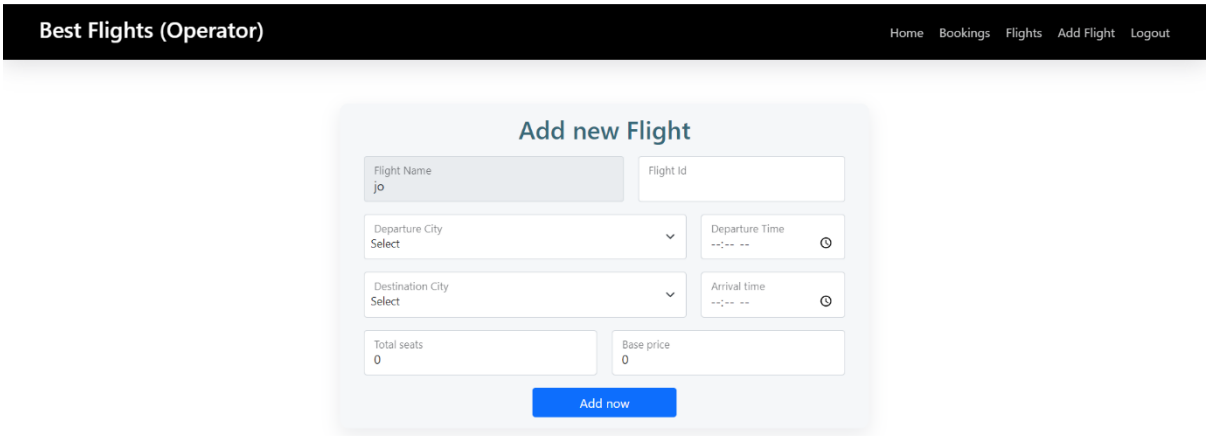

Fig 15.7 Add Flights Information

Admin Home Page

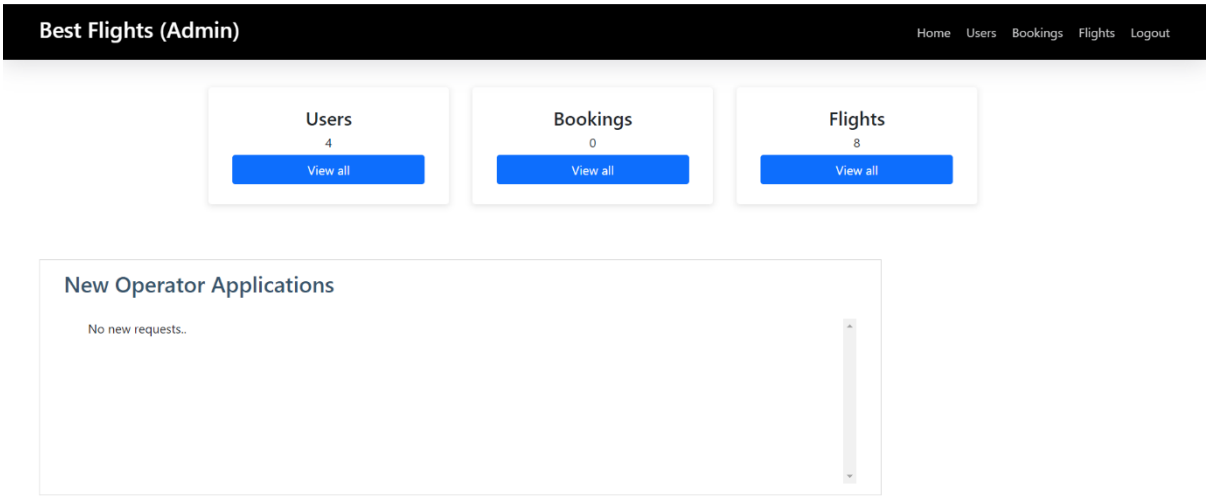

Fig 15.8 Admin Home page

Flights Info Page

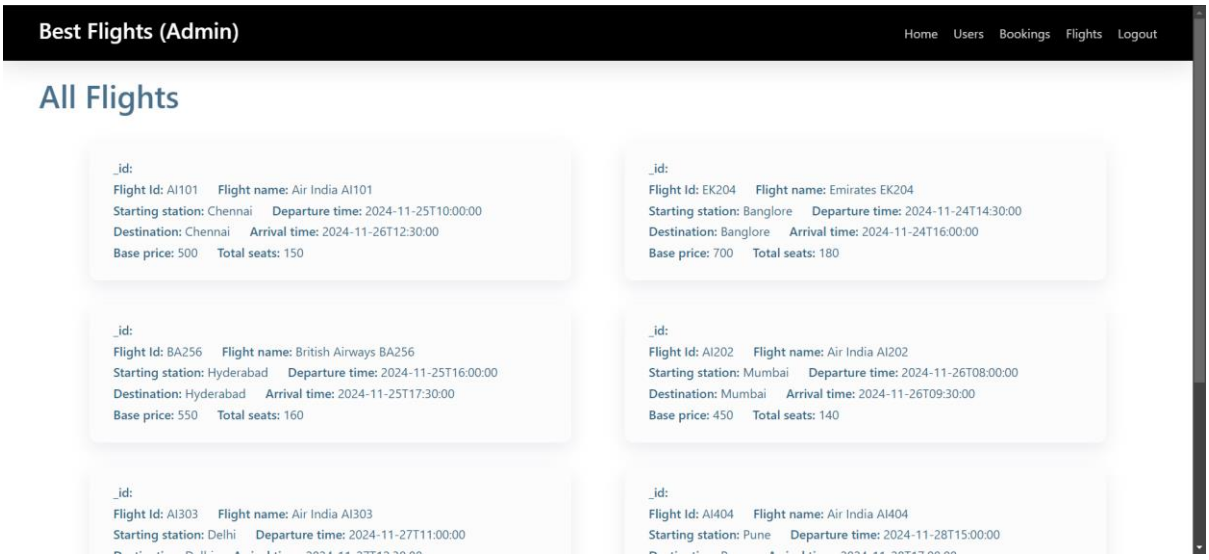

Fig 15.9 Flights Information

16. TESTING STRATEGY

Functional Testing:

- **Purpose**: Ensure core features like flight search, ticket booking, payment processing, and seat selection function as expected.

- **Tools**: Selenium, Postman (for API testing), Cypress.

Security Testing:

- **Purpose**: Identify vulnerabilities such as SQL injection, XSS, and ensure secure storage of sensitive data (e.g., user credentials and payment details).

- **Tools**: OWASP ZAP, Burp Suite, Acunetix.

Performance Testing:

- **Purpose**: Evaluate the system's performance under peak traffic conditions to identify bottlenecks or crashes.

- **Tools**: Apache JMeter, LoadRunner, Gatling.

Usability Testing:

- **Purpose**: Assess the user interface and navigation for ease of use, ensuring users can easily search and book flights.

- **Tools**: User Testing, Crazy Egg.

Integration Testing:

- **Purpose**: Verify that external services like payment gateways and third-party APIs (e.g., airline APIs) are seamlessly integrated.

- **Tools**: Postman, SoapUI.

Regression Testing:

- **Purpose**: Ensure that newly implemented features do not break existing functionalities.
- **Tools**: Selenium, TestNG.

17. KNOWN ISSUES

- **Slow Loading Times:** Slow server response times or inefficient code can lead to delays in page loading, affecting user experience.
- **Security Vulnerabilities:** Inadequate security measures can expose the application to risks such as data breaches and unauthorized access.
- **Inventory Management:** Keeping the inventory updated in real-time can be challenging, leading to discrepancies between actual stock and displayed stock.
- **Payment Processing Issues:** Integration with payment gateways can sometimes result in failed transactions or delays in payment processing.
- **User Experience Problems:** Poorly designed user interfaces or navigation can make it difficult for users to find and purchase books.
- **Database Performance**: As your bookstore grows, managing large datasets and ensuring efficient queries can become challenging. Proper indexing and optimizing queries are essential to maintain performance.
- **Authentication and Security**: Implementing secure authentication and protecting sensitive data (like user credentials and payment information) is crucial. Using JWT for authentication and ensuring HTTPS is used for data transmission are good practices.
- **Frontend-Backend Integration**: Ensuring smooth communication between the frontend (React.js) and backend (Node.js/Express.js) can sometimes be tricky, especially when handling asynchronous operations and managing state.
- **Scalability**: Designing the application to handle increased traffic and data is important. Using load balancing, caching, and considering cloud services can help in scaling the application.
- **User Experience (UX)**: Creating a user-friendly interface that is intuitive and responsive across different devices can be challenging. Regular user testing and feedback are essential to improve UX.
- **Error Handling**: Properly handling errors and providing meaningful error messages to users can improve the overall user experience and help in debugging issues.
- **Deployment and Maintenance**: Setting up continuous integration/continuous deployment (CI/CD) pipelines and maintaining the application can be time-consuming. Using tools like Docker for containerization and deployment platforms like Heroku or AWS can simplify this process.

18. FUTURE ENHANCEMENTS

- **Advanced Search and Filtering:** Implement advanced search functionality to allow users to filter flights by criteria such as departure and arrival times, airlines, flight duration, price range, and travel class. This will enable users to find the most suitable flights quickly and easily.

- **Personalized Recommendations:** Develop a recommendation engine that suggests flights based on user preferences, past bookings, and travel history. Machine learning algorithms can be utilized to analyse user behaviour and offer tailored flight options for an enhanced user experience.

- **Social Features:** Integrate social features like the ability to share travel experiences, itineraries, or reviews on social media platforms. Users can engage in travel forums or join travel groups, fostering a sense of community and enhancing the platform's user engagement.

- **Mobile App Development:** Create a cross-platform mobile application for booking flights on the go. Using React Native or a similar framework, you can offer users a seamless and optimized booking experience, making it convenient for them to check flights, manage bookings, and receive real-time notifications.

- **Subscription Service:** Introduce a subscription-based model where users can access exclusive flight deals or receive a certain number of discounted tickets per month. This model can improve user loyalty and offer consistent revenue for the application.

- **Enhanced Payment Options:** Integrate multiple payment gateways to provide users with a variety of payment options, including credit/debit cards, digital wallets (like PayPal or Google Pay), and bank transfers. This will enhance the checkout experience and cater to a wider audience.

- **Search and Assistants:** Implement voice search capabilities, allowing users to search for flights using voice commands. Integrating with virtual assistants such as Amazon Alexa or Google Assistant can make the user experience more interactive and accessible.

- **Analytics and Insights:** Build an analytics dashboard for administrators to track user behaviour, booking trends, popular destinations, and other key metrics. This will help in making data-driven decisions to optimize the platform and improve user satisfaction.

- **Localization and Multilingual Support:** Expand your platform's reach by adding support for multiple languages and localizing content based on the user's region.

19. CONCLUSION

The flight booking application is designed to provide a user-centric, secure, and efficient platform for simplifying the travel booking process. With an intuitive interface, users can effortlessly search for flights, book tickets, and enjoy secure payment integration, all while receiving real-time updates on their bookings. The application is built to address common challenges like security vulnerabilities, performance issues, and gaps in user experience, ensuring reliability and trustworthiness throughout the process.

In its current state, the application offers a streamlined and hassle-free experience, allowing users to easily compare flight options, select preferences, and manage their bookings with minimal effort. Enhanced security measures are in place to protect sensitive data, while a fast and responsive backend ensures high performance even under heavy traffic.

Looking ahead, the platform plans to introduce future enhancements like mobile app development, personalized recommendations, and advanced flight search functionalities. These updates will ensure the application remains adaptable to evolving customer needs and industry trends, keeping it competitive and relevant in the fast-changing travel sector. Additionally, features such as social sharing, a loyalty program, and eco-friendly travel options will add further value, making the platform a more comprehensive travel companion.

With its robust and scalable infrastructure, the flight booking application is poised to redefine convenience in flight ticket booking, positioning itself as a leading choice for travelers worldwide.

20. REFERENCES

React.js Documentation (2024):

• React – A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/

MongoDB Documentation (2024):

• MongoDB: The Database for Modern Applications. Retrieved from https://www.mongodb.com/

Node.js Documentation (2024):

• Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Retrieved from https://nodejs.org/

Libraries and Frameworks:

Node.js (2024):

• Node.js Documentation. Retrieved from https://nodejs.org/

Express.js (2024):

• Express - Fast, unopinionated, minimalist web framework for Node.js. Retrieved from https://expressjs.com/

JWT.io (2024):

• JWT: JSON Web Token. Retrieved from https://jwt.io/

Bootstrap (2024):

• Bootstrap: The most popular HTML, CSS, and JS library in the world. Retrieved from https://getbootstrap.com