

# **Flight Ticket Booking Application Using Mern Stack**

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

## BONAFIDE CERTIFICATE

Certified that the project report titled “**FLIGHT TICKET BOOKING APPLICATION USING MERN STACK**” is Bonafide work of “**Suriya Prakash S (211121205027)**”, “**Yoganathan B (211121205029)**”, “**Hariharan R S (211121205010)**”, “**Hariharan S (211121205011)**”, “**Seran E (211121205026)**” who carried out the project. Work under my supervision.

Signature

**Mrs. Er. B Kalpana Sattu**, B.Tech(IT).,  
BS Psychology., MS Criminology.,  
ME CSE

HEAD OF THE DEPARTMENT

Information Technology,

Madha Engineering College,

Kundrathur, Chennai – 600069.

Signature

Mr. M. Navin Bharathi,  
M.Tech(IT)

ASSOCIATE PROFESSOR

Information Technology,

Madha Engineering College,

Kundrathur, Chennai - 600069.

Submitted for the examination held on \_\_\_\_\_

**Internal Examiner**

**External Examiner**

## **Acknowledgement**

First of all we pay our grateful thanks to the chairman Ln. Dr. S. Peter for introducing the Engineering College in Kundrathur.

We would like to thank the Director Er. A. Prakash, for giving us support and valuable suggestion for our project.

It is with great pleasure and privilege we express our sincere thanks and gratitude to Dr. Venugopalakrishnan, M.E., M.B.A., Ph.D., Principal, for the spontaneous help rend to us during our study in this college.

We express our sincere thanks to Mrs. Er. B Kalpana Sattu., B.Tech(IT).. BS Psychology... MS Criminology.. ME CSE, Head of the Computer Science Department and our project Co-ordinator Mr. M. Navin Bharathi, M.Tech(IT), for the Good will fostered towards and for their guidance during the execution of this project.

It is a great privilege to express our sincere thanks to our Internal Guide Mr. M.Navin Bharathi, M.Tech(IT), and we acknowledge our indebtedness to her for the encouragement valuable suggestions and clear tireless guidance given to us on the preparation and execution of this project.

We would like to thank all the teaching and non-teaching Staff Members & Friends of the Computer Science Engineering Department for giving the support and valuable suggestions for our Project work.

## **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

<b>S. No.</b>	<b>Topic</b>	<b>Page No.</b>
1	Introduction	7
1.1	Components of the MERN Stack in the Flight Ticket Booking Application	8
2	Pre-Requirement	9
3	Project Overview	9
3.1	Purpose	9
3.2	Features and Functionalities	10
4	Architecture	11
5	Setup Instruction	11
6	Configure environment variables	12
6.1	Frontend	12
6.2	Backend	12
7	Folder Structure	13
8	Running the Application	14
9	Extension & Packages	15
9.1	Backend Packages	15
9.2	Frontend Packages	34
10	API Documentation	52
11	Authentication	53
11.1	Admin API Steps	54
12	Authentication Mechanism	55
13	Authorization	56
13.1	Key Components	56
13.2	Implementation Overview	56
14	Demo Link	57
15	User Interface and Screenshot	57
16	Testing Strategy	64
17	Known Issues	65

18	Future Enhancements	65
19	Conclusion	66

## List of Figures:

<b>Fig</b>	<b>Title of the Figure</b>	<b>Page No.</b>
Fig 4.1	Architecture	11
Fig 7.1	Folder Structure	13
Fig 7.2	Frontend	13
Fig 7.3	Backend	13
Fig 8.1	Frontend Application running	14
Fig 8.2	Backend Application running	14
Fig 15.1	Home page	57
Fig 15.2	User Login page	58
Fig 15.3	Registration page	58
Fig 15.4	Flights page	59
Fig 15.5	Ticket Booking	59
Fig 15.6	Operator Home page	60
Fig 15.7	Add Flights Information	60
Fig 15.8	Admin Home page	61
Fig 15.9	Flights Information	61

## **1.Introduction:**

The emergence of online platforms has transformed how users book flights, enabling them to compare options, purchase tickets, and manage travel plans with ease. A flight ticket booking application built using the MERN stack—comprising MongoDB, Express.js, React.js, and Node.js—delivers a robust and scalable solution for modern travel needs. This application aims to provide users with an intuitive interface for browsing flights, booking tickets, and tracking bookings, while administrators can efficiently manage flights, customers, and bookings.

### **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:

- **HTML, CSS, and JavaScript:** Fundamental knowledge for building the application's structure, styling, and client-side interactivity.
- **Database Connectivity:** Use MongoDB drivers or Mongoose (ODM library) to enable database CRUD (Create, Read, Update, Delete) operations.
- **Version Control:** Utilize Git for collaborative development and tracking changes. Platforms like GitHub or Bitbucket are suitable for hosting repositories.
- **Tools:** Visual Studio Code, Sublime Text, or WebStorm for code editing.
- **Setup:**
  - **Git:** [Download Git](#).
  - **Node.js:** [Download Node.js](#).



- **MongoDB:** Use Community Edition or MongoDB Atlas for database setup.

### 3. Project Overview:

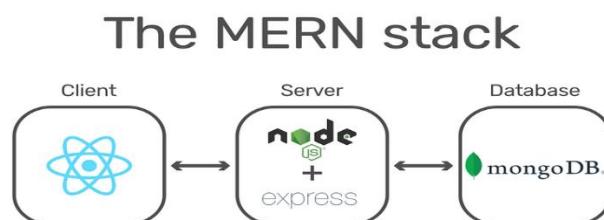
#### 3.1. Purpose:

A flight ticket booking application offers users an intuitive online platform to search, book, and manage their flight tickets. For administrators, it simplifies flight management, inventory updates, and booking tracking.

#### 3.2. Features and Functionalities:

1. **Flight Search and Filter:** Users can browse and filter flights based on criteria like departure time, destination, and price.
2. **Secure Booking and Payments:** A secure checkout process with multiple payment options ensures smooth ticket purchases.
3. **Booking Management:** Provides users with a dashboard to view, modify, or cancel bookings.
4. **Personalized Recommendations:** Suggests flights based on user preferences and past bookings.

### 4. Architecture:



*Fig 4.1 Architecture*

#### Frontend

- Built using **React.js** for an interactive and responsive UI.

- Includes components for flight search, booking forms, and user dashboards.

## Backend

- Powered by **Node.js** and **Express.js**, which manage API requests and implement business logic.
- Implements RESTful APIs to handle flight data, bookings, and user authentication.

## Database

- **MongoDB** stores flight schedules, user profiles, and booking records in collections.

## 5.Setup Instructions:

### Prerequisites

- **Node.js**: Version 14+
- **MongoDB**: Community Edition or MongoDB Atlas
- **npm/yarn**: For managing packages

### Installation Steps

1. Clone the repository:

bash

Copy code

git clone <https://github.com/example/Flight-Ticket-Booking-App>

2. Navigate to the project folder:

bash

Copy code

cd Flight-Ticket-Booking-App

3. Install dependencies:

a. **Backend**:

bash

Copy code

```
cd backend && npm install
```

**b. Frontend:**

bash

Copy code

```
cd frontend && npm install
```

## **6.Environment Configuration:**

- ◆ Create a `.env`` file in the backend directory with required keys (e.g., database URI, JWT secret).
- ◆ Create a file named `.env` in the root directory of your project. This file will store your environment variables. Here, you can define variables like `PORT`, `MONGODB_URI`, `JWT_SECRET`, and any other sensitive or configuration-specific information.

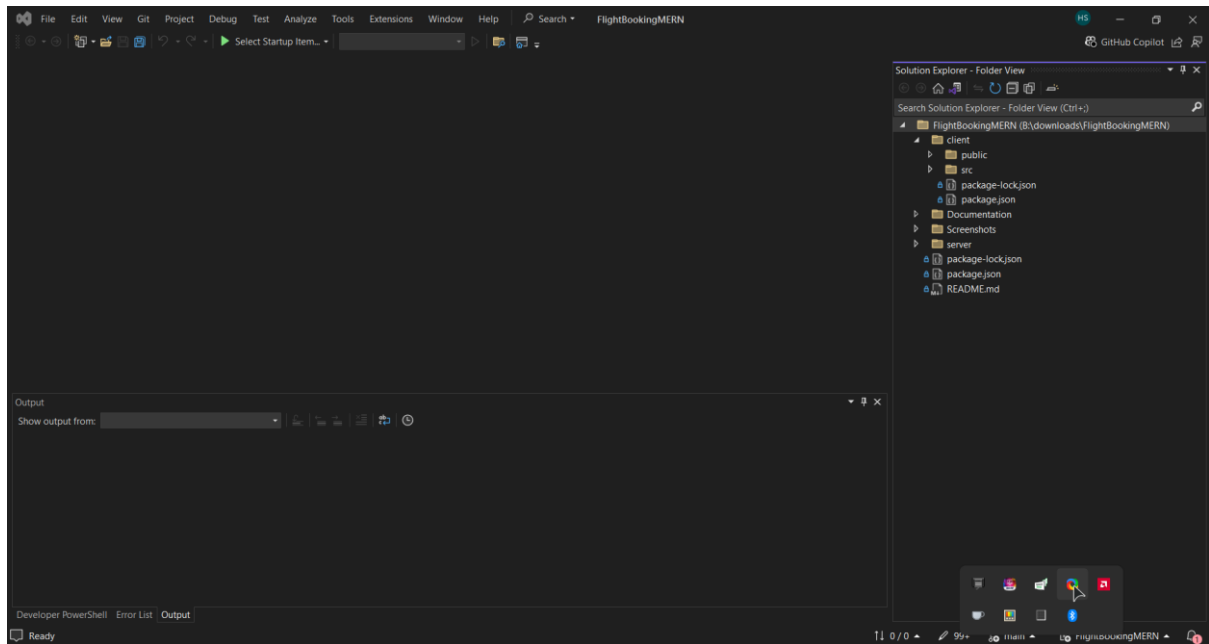
### **6.1 Frontend:**

- ``src/``: Contains reusable UI components.
- ``src``: Houses different pages (User, Seller, Admin, Components).
- ``src/app.js``: Manages API calls and handles communication with the backend.

### **6.2 Backend:**

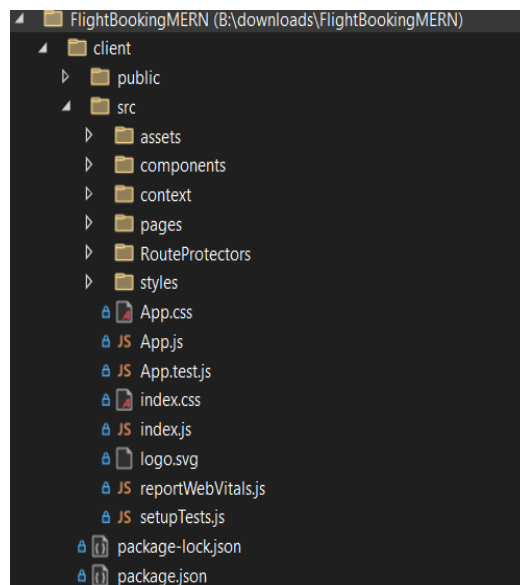
- ``db/User``: Holds logic for handling various API requests and response for user components.
- ``db/Seller``: Holds logic for handling various API requests and response for Seller components.
- ``db/Admin``: Holds logic for handling various API requests and response for Admin components.
- ``backend/jsfiles``: Contains Mongoose schemas and models for MongoDB collections.

## 7.Folder Structure:



*Fig 7.1 Folder Structure*

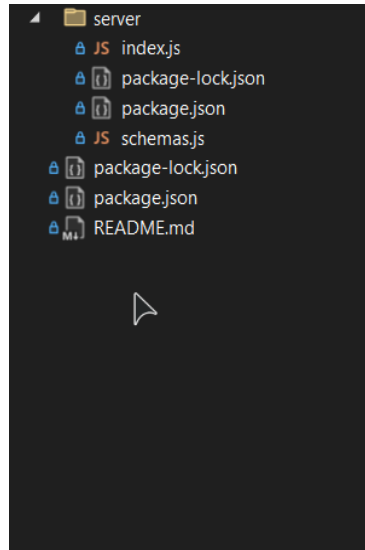
## Frontend:



*Fig 7.2 Frontend*

- **src/components:** Contains reusable UI components.
- **src/pages:** Houses flight search, booking, and user dashboard pages.
- **src/app.js:** Manages frontend API calls and state handling.

## Backend:



*Fig 7.3 Backend*

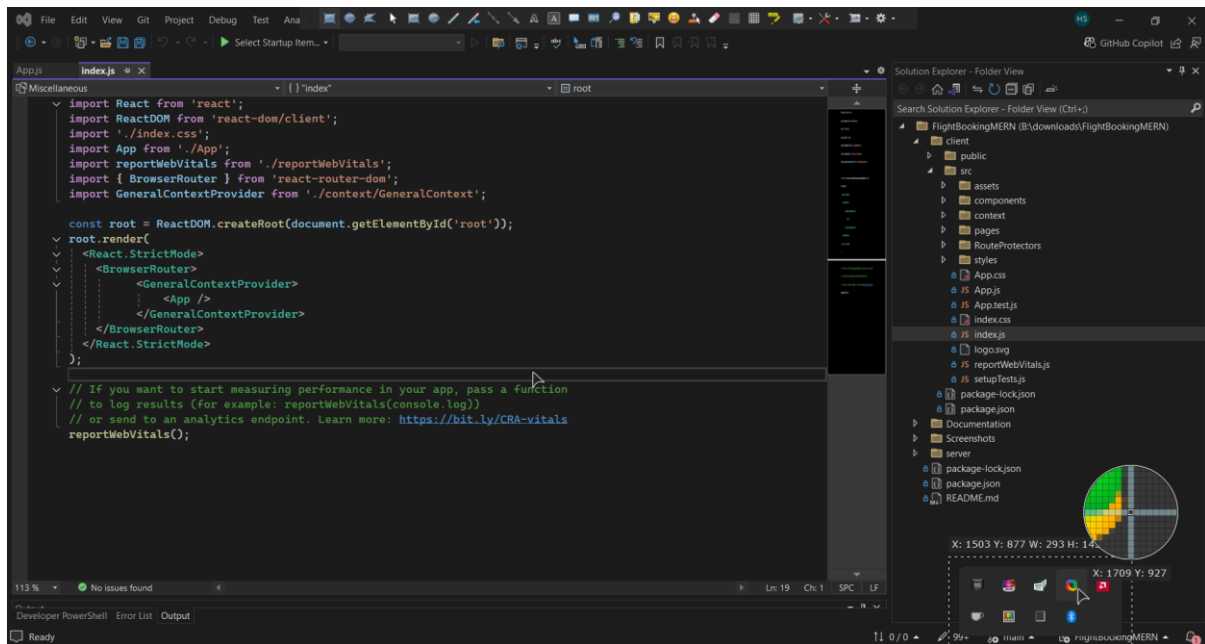
- **routes/**: Defines API endpoints for flights, users, and bookings.
- **models/**: Contains Mongoose schemas for MongoDB collections.
- **controllers/**: Includes logic for handling user requests and responses.

This MERN stack-based flight ticket booking application ensures a robust and user-friendly experience while allowing for scalability and efficient management of travel services.

## 8. Running the Application:

### ➤ Frontend:

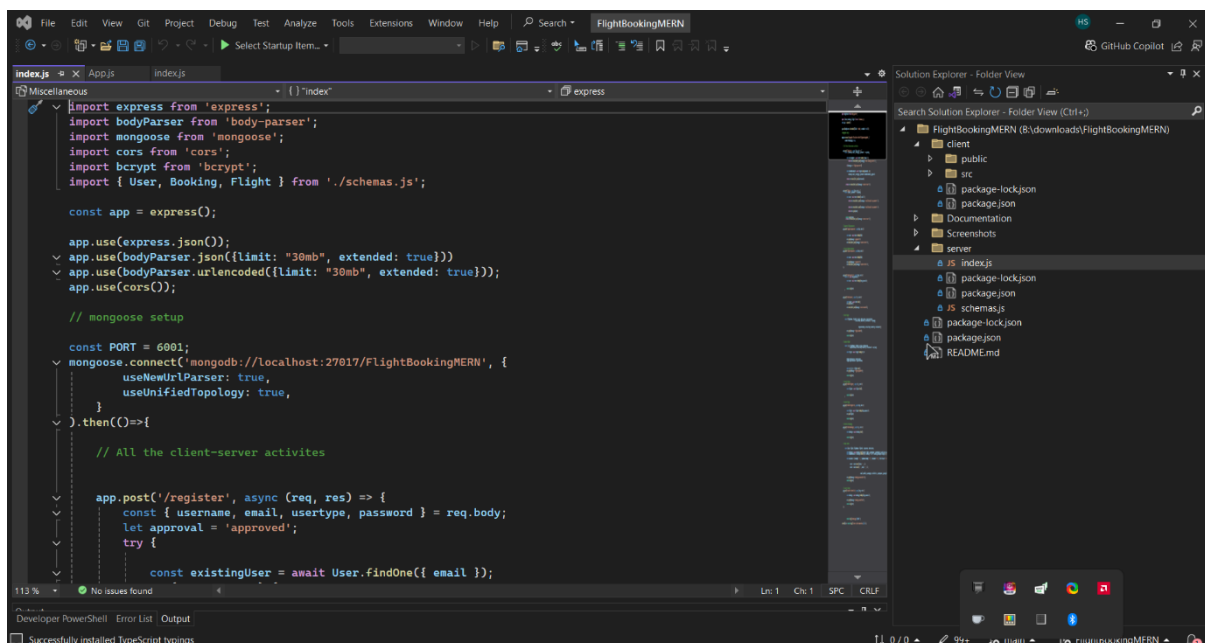
- To navigate to frontend, use the command ``cd frontend`` in terminal.
- Navigate to the frontend directory and run ``npm run dev`` to start the React application.



*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:

```
{
  "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+LTIYoeivqYo853f02jBYSd5uGnGpkFV0M3xOt9aN73erkgYAmZU43x4VfqcnL
xW9Kpg3R5LC4YYw==",
  "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-RJlIHrueQgwWitWgF80dFYGZX328Ax5BCemNGlqHfpInRT9ESi8JkFlvaVYbS+UubVY6dpv
87Fs2u5M29iNFVQ==",
  "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+aUcdiJuCIqgujCshpMWgUpG0xXotrYPSA53hA3qopNaqcJpyr0hVb1FeWdnqFA
35/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/IqMa1PMPz5RnHuHscF8/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/MprG9yArRkyrQxT06XjMzA==",
    "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+EE0UZd3DYqfSkMhY2/wfSeaHV1Ew4vk8e+x8lja31Ibyu
Ua2uQ3JONqKbysw==",
      "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+HaC1t1CKd0z3R4b6hdYEcm
rNZHUmfCP+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-
C471C7LIDCnz0h4vai/tpNOI95tCd5ZT3iBt/DBH51XKHZsyNQv18yf1wIIg2ntiQNgmAvA
+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/whHD9iW76PmNzxZ4UCnTQTVEQ==",
    "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/MprG9yArRkyrQxT06XjMzA==",
    "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++GUozlw8VBJDEOAqSXwm1WvmYEZwqdqSGQshRocPDqrt4HBZB3fQ==",
    "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/MprG9yA
rRkyrQxT06XjMzA==",
  "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-
ORPNZ3h6ZRk0yAa/SaHU+XsLZr0UQzRwuDQ0cczIA17nAzZ+85G5cVk0JIj7QavLZGSe8QX
UmNFxSZzjcZF9bw==",
  "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-
BR7VvDCVH0+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9yA
rRkyrQxT06XjMzA==",
      "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-
xzCjAoKu0b55CSwQrbyWBKqp35yg1vw9ohIlU2wTy06ZrYfJ8rKochb1MSGlnoBfXGWss3
UPzxR5QL5guIFdg==",
      "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+qAmLmMv1QHPI2jfDwYqrytOfKfYq42bYtW8
Pr21ejZX7GvAL0w==",
      "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/QYapCdVwwmKlU9mHdmy52KsDGIjVaqE
UMFvEzn2LRaigqQ==",
    "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/x2Bf5pWnbXdd3j57xnb0V60FW1LVn9LRZkPtneODklzYuFZv7qK
6riZ5BNyc0R0jZA==",
    "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-
FkbtrODA4Imsi0p7TW9u9MXuQ5P4pVs1sWHK4dJMMChVR0sbEltuE79fBoIk/BCztv0J7yU
pErMKa4z3uQLX+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/zj6VDs0PigYwSN23L8hseSRw1jrceU2WzTvK/+BSYIC
sWmRq5qHS2UYzhw==",
    "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/+ndMkJkIT4/6U0acU3A4BwZSAC9pQ9vSvJpIi/WFGQRH/uPXvuVj
F5c2RfIPQFSS2uA==",
    "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-
0jXdusCL2uC5gM3yYFT6QMBzKfBr2XTk0g5TPAV4y8IZDyVNDyjl1a8uSXy3/XrvkVTmQvLN
405k3JawGReXr9w==",
    "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-
gJ9jd8Mb8wHLbRz9ZvGN57IAmkn0ipD3W4XNE/Lk/4lqs5Htw4W0QgakQy/o/4CoXQlMCYl
daqUg+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+rqVw8qwcmZnQGpjjiek8A2wpbxSJB1FdCoQV0+jDJs35bFgo/
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-rvw3Q7Zc81AxyVr9cSGVm5yP/IJ2UcB3U0graE3LCFoZ0Yn2x4EoVSqJKdB/T5M+FLcRPjz4TDacRf30CfNUzw==",
    "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-P1WjMrUB3qgJN19jfmPZ/htmBEjFh//6l/5y8SD9hg1Ef5zTTVVoRjTrTEzPrNBQvmhMxkoTsj0XN10GWU7aCg==",
    "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/iN2/RPVJ5jYK8BGttj5z83LmSKmbvvrXPNC LZSEb32KKVDJD1/M0t2N01qU2H/FkzEa9PKto1BqDjtd7Q==",
    "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-
VHskAKYM8RfSFxwee5t5cbN5PZeq1Wrh6qd5bkyiXI f6UQcN6w/A0eXM9r6t8d+GYOh+o6Z
hiEnb88LN/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+KD
UqDFctkdRW44e1w==",
    "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/IQ04wxtcFB4bg
pQPRcR3QCvezPcQ==",
    "license": "ISC"
  },
  "node_modules/write": {
    "version": "1.0.3",
    "resolved": "https://registry.npmjs.org/write/-/write-1.0.3.tgz",
    "integrity": "sha512-
/lg70HAjtkUgWPVZhZcm+T4hkL8Zbtp1nFN0n3lRrxnlv50SRBv7cR7RqR+GMsd3hUXy9hW
Bo4CHTbFTcOYwig==",
    "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+QfwAaySIw4AXP9ZskTTh5GWu3
FLuJhsWVdiJPGvg==",
      "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-FfCejsuC8f9k0SU9FYaOw8Cd06803h5v0lg4p74o8JqWpwTf9tniOD+n0B78aWoVSS6WptV
UmDrp/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/sFaDxgd6fxm1ewUaM0jZ444F
c5vC5R0Yurg/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+uWBUYOWPFwW7W7
PRLRfUKpxjtjFCw==",
      "license": "MIT"
    },
    "node_modules/xregexp": {
      "version": "4.4.1",
      "resolved": "https://registry.npmjs.org/xregexp/-/xregexp-
4.4.1.tgz",

```

```

    "integrity": "sha512-
2u9HwfadaJaY9zHtRRnH6BY6CQVNQKkYm3oLtC9gJXXzfsbACg5X5e4EZZGVAH+YIfa+QA9
1sFQTTe3HURF3ag==",
    "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/mCWbtLsGjFI
ad0wIsod4zrTAEQ==",
    "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+rSqNDC74L
CSFmlb/U4UZ4hJQ==",
    "license": "ISC"
  },
  "node_modules/yallist": {
    "version": "3.1.1",
    "resolved": "https://registry.npmjs.org/yallist/-/yallist-
3.1.1.tgz",
    "integrity": "sha512-
a4UGQaWPH59mOXUYnAG2ewncQS4i4F43Tv3JJoAM+s2VDAmS9NsK8GpDMLrCHPkSFT7h3K6T
OoUNn2pb7RoXx4g==",
    "license": "ISC"
  },
  "node_modules/yaml": {
    "version": "1.10.2",
    "resolved": "https://registry.npmjs.org/yaml/-/yaml-1.10.2.tgz",
    "integrity": "sha512-
r3vXyErRCYJ7wg28yvBY5VSoAF8Zv1cW9/BwUzEtUsjvX/DKs24dIkuwjtuprwJJHsbyUbl
ApepYTR1BN4uHrg==",
    "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+i5o9bwghAogtM08q1dpr2
LVoS8KSTMYpWUw==",
    "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+A4QuSZfDRA7HRSfWrz00YjqTJd5kjAq37Zep1CEgaYmrH9Q3GwPiB9cHyd1Y1
UwggGhJGoxipbzb==",
    "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-
VHskAKYM8RfSFXwee5t5cbN5PZeq1Wrh6qd5bkyiXIIf6UQcN6w/A0eXM9r6t8d+GY0h+o6Z
hiEnb88LN/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+KD
UqDFctkdRW44e1w==",
    "license": "MIT",
    "dependencies": {
      "emoji-regex": "^7.0.1",
      "is-fullwidth-code-point": "^2.0.0",
      "strip-ansi": "^5.1.0"
    },
    "engines": {
      "node": ">=6"
    }
  }
}
}

```

The package-lock. Json file is automatically generated when you run npm install in a Node.js project. This file ensures that the exact versions of dependencies are installed, maintaining consistency across different environments. Here is a brief overview of its primary functions:

### **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"
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>
      <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>
    </form>
  )
}
```

```

        <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>

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

```

```

        </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>
    )
  }
}

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 Flight.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>
  )
}
export default BookFlight

```

## 10.API Documentation:

### Add a Flight

- **Method:** POST /flights
- **Description:** Adds a new flight to the database.

### View All Flights

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

### Update Flight Details

- **Method:** PUT /flights/{id}
- **Description:** Updates the details of a specific flight by its ID.

### Delete a Flight

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

## Search Flights

- **Method:** GET /flights/search
- **Description:** Searches for flights by departure city, destination, date, or airline.

## 11.Authentication:

### 1.User Registration:

- **Request:**
  - User provides details (e.g., name, email, password, phone, etc.).
- **Process:**
  - System validates the input data (e.g., checks for duplicate email or invalid formats).
  - If validation passes, a new user account is created and stored in the database (password is hashed).
- **Response:**
  - Success message with the user ID.

### 2. User Login:

- **Request:**
  - User submits login credentials (email and password).
- **Process:**
  - System verifies credentials against database records.
  - If valid, generates a token (JWT or similar) for session authentication.
- **Response:**
  - Token (used for subsequent authenticated requests).

### 3. Get User Profile:

- **Request:**
  - User sends a request with the authentication token.
- **Process:**
  - System validates the token and retrieves user details from the database.
- **Response:**
  - Returns the user's profile information (name, email, booking history, etc.).

### 11.1.Admin API Steps:

#### 1. Admin Login:

- **Request:**
  - Admin submits credentials for login.
- **Process:**
  - System verifies the credentials.
  - If valid, generates a token for admin-level access.
- **Response:**
  - Token for making authorized admin requests.

#### 2. View All Users:

- **Request:**
  - Admin sends a request with the admin authentication token.
- **Process:**
  - System verifies the token and fetches all user records from the database.
- **Response:**
  - Returns a list of all registered users (ID, name, email, roles, etc.).

### 3. Delete a User:

- **Request:**
  - Admin provides the user ID to delete in the request.
- **Process:**
  - System verifies the admin token and checks if the user ID exists in the database.
  - If valid, deletes the user record.
- **Response:**
  - Confirmation message indicating successful deletion.

### 4. Update User Role:

- **Request:**
  - Admin specifies the user ID and the new role (e.g., "customer," "agent").
- **Process:**
  - System verifies the admin token and validates the new role.
  - If valid, updates the user's role in the database.
- **Response:**
  - Success message confirming the role update.

## 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.
  - The JWT is returned to the client for use in future requests.

## 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:**
  - The JWT can be stored:
    - **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:

make file

Copy code

Authorization: Bearer <token>

## 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.
- **Access Control:**
  - If valid, the middleware grants access to the requested resource.
  - Access levels (e.g., admin vs. customer) are determined by token claims.

## 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:

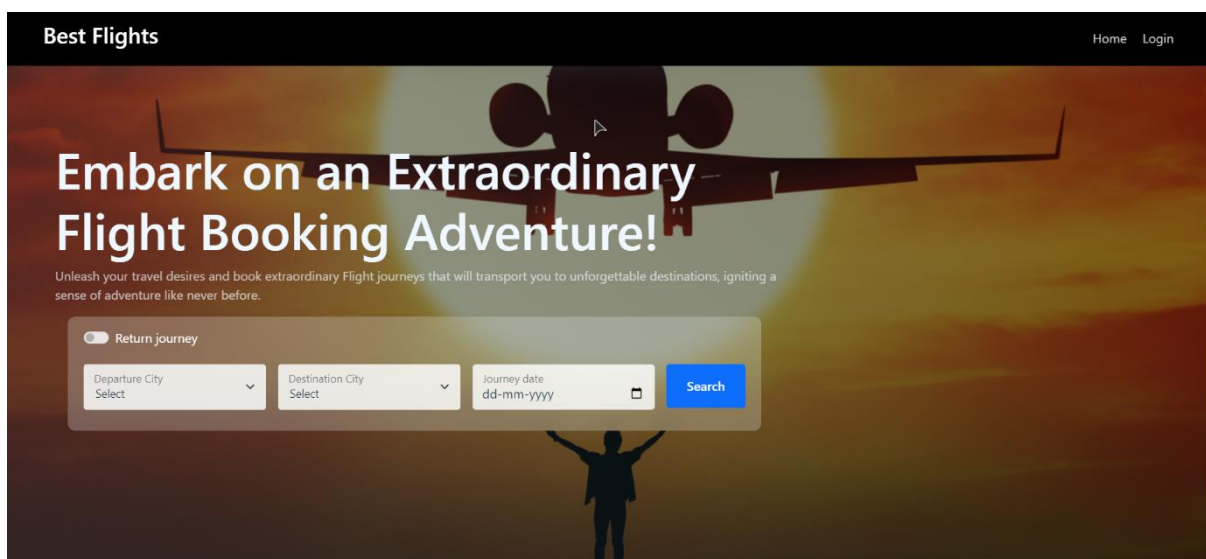
- **Common Roles:**
  - **Admin:**
    - Full access to all resources and management capabilities.
    - Examples: Managing users, flights, and bookings.
  - **Seller (e.g., Airline Partner):**

- 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.

**14.Demo Link:** [Flight Booking App](#)

## 15.User Interface and Screenshot:

### Home Page



*Fig 15.1 Home page*

## User Login Page

Best Flights

Home Login

Login

Email address  
abc@gmail.com

Password  
\*\*\*\*\*

Sign in

Not registered? Register

Fig 15.2 User Login page

## User Register Page

Best Flights

Home Login

Register

Username

Email address

Password

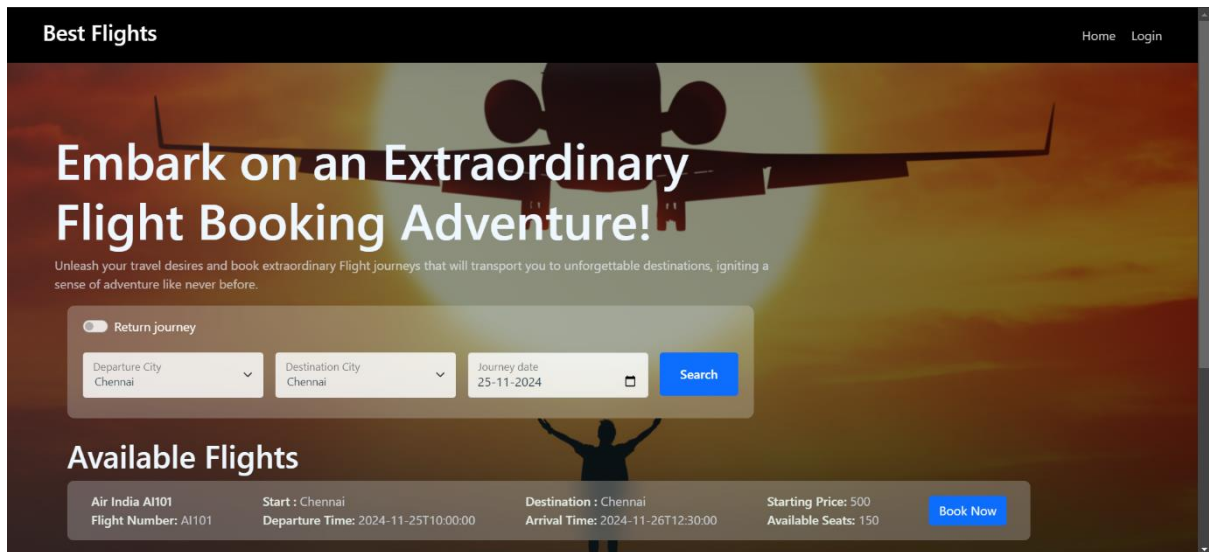
User type

Sign up

Already registered? Login

Fig 15.3 Registration page

## Available Flights Page



The image shows a web page titled "Best Flights" with a dark header. The main content area has a background image of an airplane flying over a sunset. The text "Embark on an Extraordinary Flight Booking Adventure!" is prominently displayed. Below this, a sub-header reads "Available Flights". A search form is present with fields for "Departure City" (Chennai), "Destination City" (Chennai), and "Journey date" (25-11-2024). A "Search" button is next to the date field. Below the search form, a table lists flight details for "Air India AI101". The table includes columns for flight name, start location, destination, starting price, and available seats. A "Book Now" button is located at the end of the row.

Best Flights Home Login

# Embark on an Extraordinary Flight Booking Adventure!

Unleash your travel desires and book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.

☐ Return journey

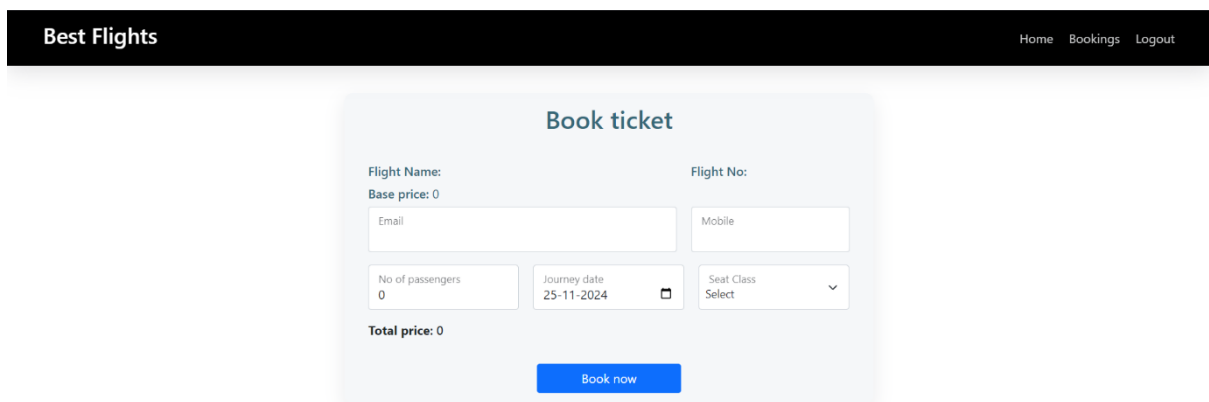
Departure City: Chennai | Destination City: Chennai | Journey date: 25-11-2024 Search

## Available Flights

Air India AI101 Flight Number: AI101	Start : Chennai Departure Time: 2024-11-25T10:00:00	Destination : Chennai Arrival Time: 2024-11-26T12:30:00	Starting Price: 500 Available Seats: 150	<span>Book Now</span>
---	--	--	---	-----------------------

Fig 15.4 Flights page

## Ticket Booking Page



The image shows a web page titled "Best Flights" with a dark header. The main content area has a light gray background. A central form titled "Book ticket" contains several input fields. The form is divided into two columns. The left column has fields for "Flight Name:", "Base price: 0", "Email", "No of passengers" (0), and "Total price: 0". The right column has fields for "Flight No:", "Mobile", "Journey date" (25-11-2024), and "Seat Class" (Select). A "Book now" button is located at the bottom of the form.

Best Flights Home Bookings Logout

### Book ticket

Flight Name:  Flight No:

Base price: 0

Email:  Mobile:

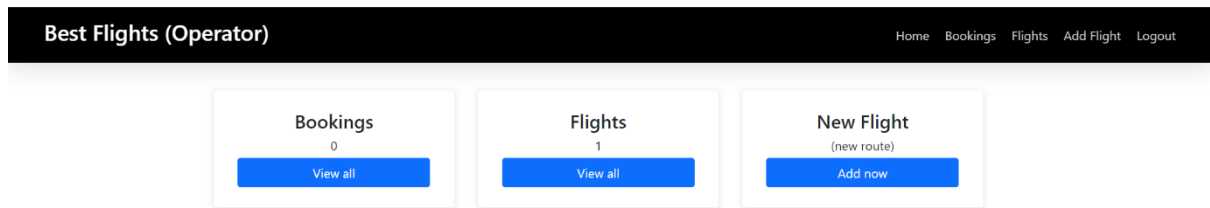
No of passengers:  Journey date: 25-11-2024  Seat Class:

Total price: 0

Book now

Fig 15.5 Booking page

## Flight Operator Home Page



*Fig 15.6 Operator Home page*

## Add Flights Page

Best Flights (Operator) Home Bookings Flights Add Flight Logout

Add new Flight

Flight Name  
jo

Flight Id

Departure City  
Select

Departure Time  
--:-- --

Destination City  
Select

Arrival time  
--:-- --

Total seats  
0

Base price  
0

Add now

*Fig 15.7 Add Flights Information*

## Admin Home Page

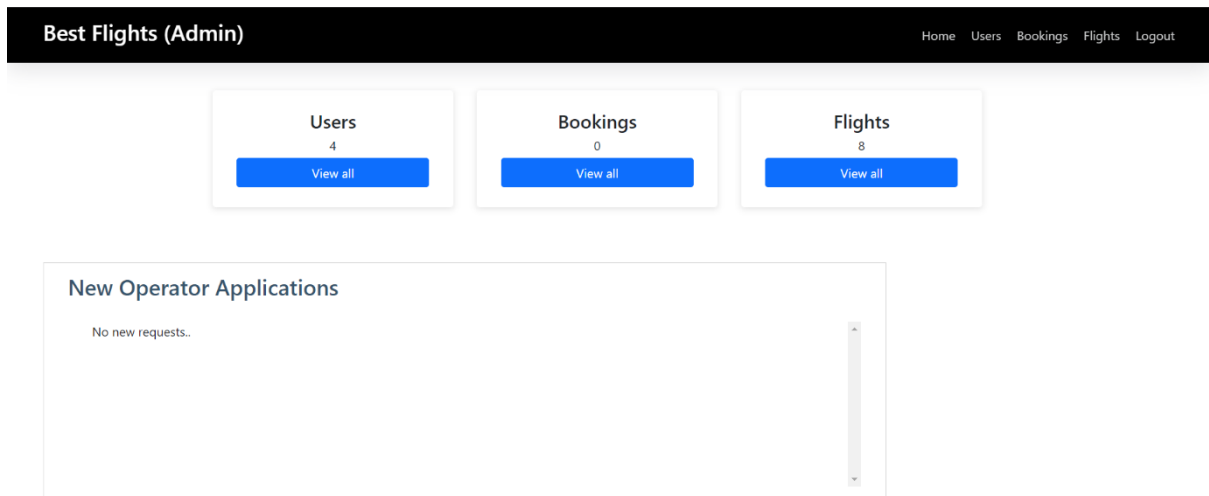


Fig 15.8 Admin Home page

## Flights Info Page

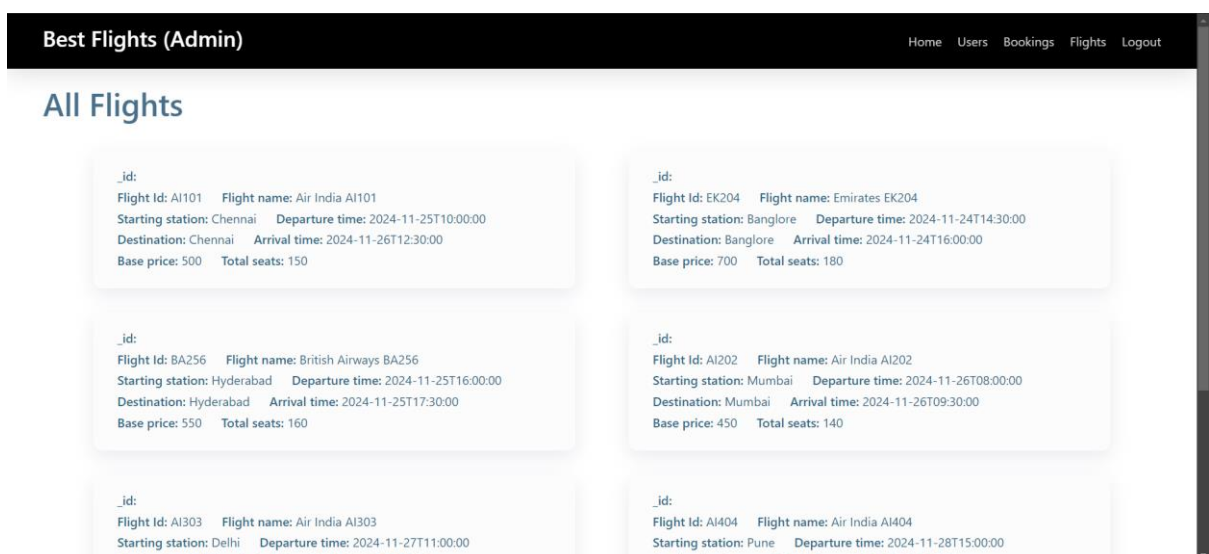


Fig 15.9 Flights Information

## 16. Testing Strategy:

### 1. Functional Testing:

- 2. ➤ **Purpose:** Ensure core features like flight search, ticket booking, payment processing, and seat selection function as expected.
- 3. ➤ **Tools:** Selenium, Postman (for API testing), Cypress.

### 4. 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.

### 5. Performance Testing:

- **Purpose:** Evaluate the system's performance under peak traffic conditions to identify bottlenecks or crashes.
- **Tools:** Apache JMeter, LoadRunner, Gatling.

### 6. 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.

### 7. Integration Testing:

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

### 8. Regression Testing:

- **Purpose:** Ensure that newly implemented features do not break existing functionalities.



➤ **Tools:** Selenium, TestNG.

## **17. Known Issues:**

### **1. Slow Loading Times:**

➤ Prolonged server response times, especially during peak traffic, can impact user experience.

### **2. Security Vulnerabilities:**

➤ Risks such as unencrypted data transmission or poor authentication mechanisms may lead to data breaches or unauthorized access.

### **3. Seat Availability Sync:**

➤ Delays in syncing real-time seat availability with airline servers may cause booking discrepancies.

### **4. Payment Processing Errors:**

➤ Issues with payment gateway integration may result in transaction failures or delayed confirmations.

### **5. User Experience Challenges:**

➤ Inconsistent UI elements, unclear navigation, or lack of accessibility features can hinder user satisfaction.

## **18. Future Enhancements:**

### **1. Mobile App Development:**

➤ Launch dedicated iOS and Android applications for a more accessible and convenient user experience on mobile devices.

### **2. Personalized User Recommendations:**

➤ Integrate machine learning to provide tailored flight suggestions based on user preferences, travel history, and search patterns.

### **3. Improved Search Functionality:**

➤ Leverage advanced search algorithms and AI-driven natural language processing to enable flexible and accurate flight searches.

### **4. Dynamic Pricing Notifications:**

➤ Offer real-time alerts for price drops, flash sales, or seat availability changes to enhance customer engagement.

### **5. Subscription-Based Offers:**

➤ Introduce loyalty programs or subscription plans with benefits like discounted fares, priority boarding, or exclusive deals.

### **6. Multi-Currency Payment Support:**

➤ Enable seamless transactions for international travellers with support for multiple currencies and payment gateways.

### **7. Enhanced Analytics:**

➤ Implement tools to track user behaviour, sales trends, and booking patterns to improve business decision-making and customer targeting.

## **19. Conclusion:**

The flight booking application is a user-centric, secure, and efficient platform designed to simplify the travel booking process. With features like intuitive flight searches, seamless ticket booking, secure payment integration, and real-time updates, the application offers a streamlined and hassle-free experience for users. By addressing known challenges such as security vulnerabilities, performance issues, and user experience gaps, the app ensures reliability and trustworthiness. Future enhancements like mobile app development, personalized recommendations, and advanced search functionalities will further elevate the user experience, making the application adaptable to evolving

customer needs and industry trends. This robust and scalable solution is set to redefine convenience in flight booking, positioning itself as a leading choice in the travel industry.