Full Stack Development with MERN

Project Documentation format

1. Introduction

- **Project Title:** Flight_booking
- **Team Members:** Sarvesh T D

2. Project Overview

Purpose: The flight booking system allows users to search, select, and reserve flights easily. It helps travelers compare prices, choose preferred schedules, and confirm tickets conveniently.

Features:

Users can search flights by date, destination, and price.

Provides seat selection, meal preference, and baggage options.

Enables secure online ticket booking and payment.

Shows real-time flight status, fare comparison, and booking history.

Supports cancellation, rescheduling, and refund tracking.

3. Architecture

- **3.1 Frontend Architecture – React**

The frontend of Flight booking is developed using React, following a modular and component-based architecture. Each UI feature is split into reusable components to ensure scalability and easy maintenance.

- **3.2 Backend Architecture – Node.js & Express.js**

The backend uses Node.js with Express.js to build a lightweight and scalable REST API.

.3.3Database Architecture – MongoDB

MongoDB is used as the database for its flexibility and document-based data structure. Mongoose ODM is used for schema management and queries.

4. Setup Instructions

- **Prerequisites:** List software dependencies.

Requirement	Version (recommended)
Node.js	v16+
npm	v8+ (comes with Node.js)

Requirement	Version (recommended)
MongoDB	v5+ (Local or Cloud — e.g., MongoDB Atlas)
Git	Latest version
Code Editor	Visual Studio Code (recommended)

Installation: Step-by-step guide to clone, install dependencies, and set up the environment variables.

Step 1: Install Frontend Dependencies

Cd frontend

npm install

Step 2: Install Backend Dependencies

cd backend

npm install

5. Folder Structure

· **Client:**

client/

├── node_modules/

├── public/

├── src/

│ ├── assets/

│ ├── components/

│ ├── context/

│ ├── pages/

│ ├── RouteProtectors/

│ ├── styles/

│ ├── App.css

│ ├── App.js

│ ├── App.test.js

│ ├── index.css

│ ├── index.js

│ ├── logo.svg

│ ├── reportWebVitals.js

│ └── setupTests.js

├── .gitignore

├── package-lock.json

├── package.json

- **Server:**

backend/
├── config/
├── controllers/
├── middlewares/
├── node_modules/
├── routes/
├── schemas/
├── uploads/
├── .env
├── .gitignore
├── index.js
├── package-lock.json
├── package.json

6. Running the Application

- commands to start the frontend and backend servers locally.
  - **Frontend:** `npm start` in the client directory.
  - **Backend:** nodemon index.js in the server directory.

7. API Documentation

POST /api/admin/approve-operator → Approves operator

POST /api/admin/reject-operator → Rejects operator

GET /api/admin/fetch-user/:id → Fetch single user by ID

GET /api/admin/fetch-users → Fetch all users

8. Authentication

Flight booking uses JWT (JSON Web Token) for authentication.

When a user logs in, the backend generates a token.

The token contains the user ID and user role.

The frontend stores the token in localStorage/sessionStorage.

For protected APIs, the token is sent in the Authorization header.

Format: "Authorization: Bearer <token>".

A middleware checks if the token is valid.

If valid, the request is allowed to continue.

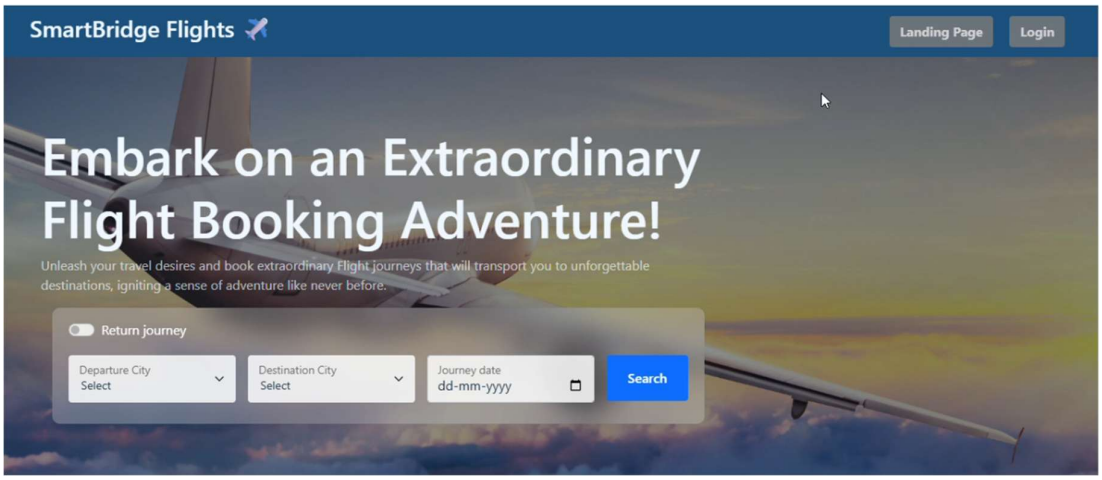
If not valid or missing, access is denied.

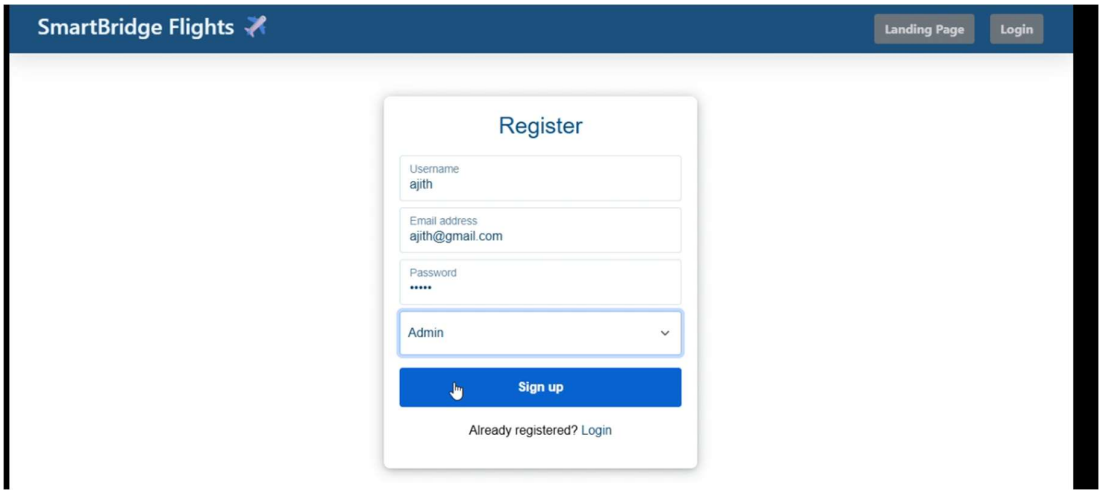
Authorization is role-based (user / admin / operator).

Users can book appointments, doctors manage requests, admins handle approvals.

No sessions are stored on the server — authentication is completely token-based.

9. User Interface





10. Testing

• Manual testing was carried out on both the frontend and backend.

Postman was used to test backend APIs like authentication, admin approval, and appointments.

Each API was tested with valid, invalid, and missing token scenarios to verify authorization.

Frontend pages were tested by navigating through all user roles (user, operator, admin) to ensure proper access control.

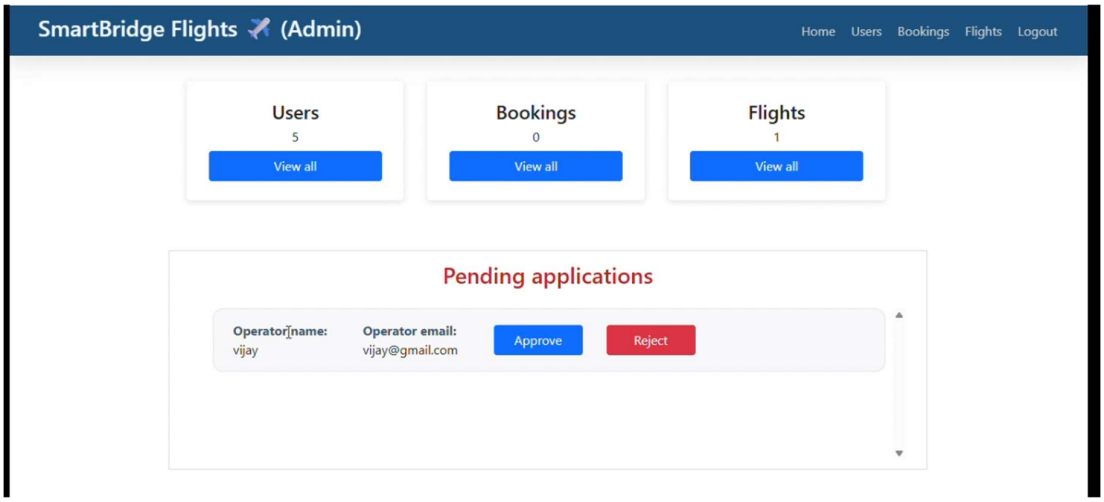
Form validation was checked by entering correct and incorrect inputs.

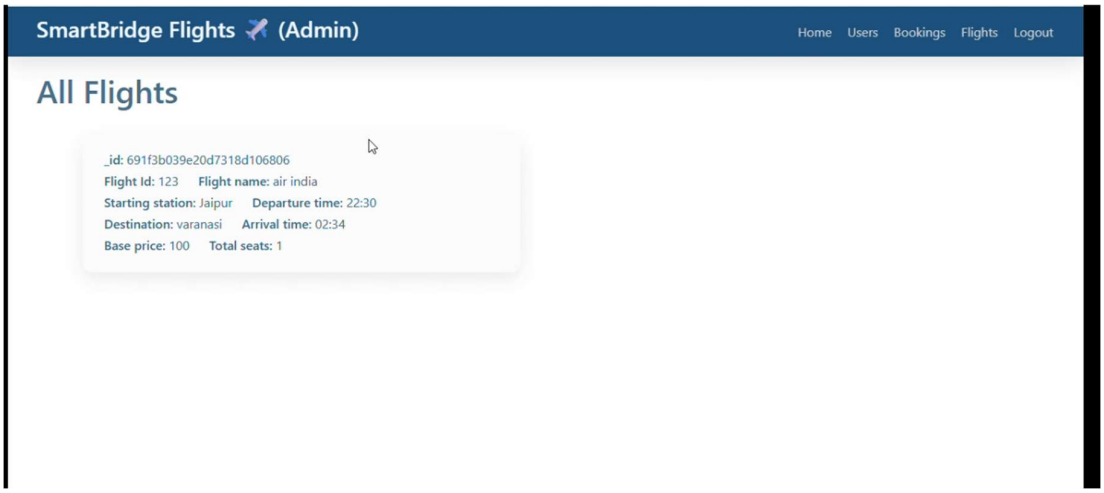
All routes and buttons were tested to confirm smooth navigation and error-free UI flow.

MongoDB was monitored to verify database updates after each action.

Testing focused on functionality, performance, and error handling rather than automation.

11. Screenshots or Demo





12. Known Issues

Sometimes API requests take longer depending on network speed.

Email notifications may not be sent if SMTP credentials are not configured correctly.

Admin dashboard may require page refresh to reflect updated doctor status.

If MongoDB connection is unstable, the server may restart automatically.

Token expiration can log out users unexpectedly if they remain inactive for a long time.

Uploading large files in the uploads folder may increase backend memory usage.

UI alignment may vary slightly on different screen sizes.

13. Future Enhancements

Add loyalty points and rewards program.Enable live flight tracking with notifications.
Introduce dynamic seat pricing based on demand.
Support multi-city and round-the-world booking.
Integrate hotel and cab booking for full travel packages.
Provide AI-based cheapest fare prediction.
Launch mobile app for Android & iOS.