

**TO DEVELOP A PROJECT ON**

**Flight Booking Application (MERN stack by MongoDB)**

A PROJECT REPORT

*Submitted by*

**PAVITHRA M - 110521104027**

**PONSURESH L - 110521104029**

**PRASANNA B - 110521104030**

**PAVITHRA MS - 110521104028**

*In partial fulfilment for the award of the degree*

*Of*

BACHELOR OF ENGINEERING

IN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY

ANNA UNIVERSITY, CHENNAI-600 025.

ABSTRACT

* FlyEasy is a full-stack flight booking application designed to simplify travel booking and improve user experience. Built on the MERN (MongoDB, Express.js, React, and Node.js) stack, the platform enables travelers to search flights, view details, and book with ease. It also features personalized flight recommendations and secure payment processing.
* For administrators, FlyEasy provides a dashboard for managing flight schedules, tracking bookings, and viewing analytics. The backend uses token-based authentication to protect user data, and the modular architecture supports scalability for future enhancements like AI-driven recommendations and real-time flight updates.
* This report covers the technical architecture, key features, and solutions implemented in FlyEasy, showcasing it as a scalable solution tailored to the needs of the travel industry.

Abstract

Table of content

**Executive Summary**

1. Introduction

- 1.1 Project Title

- 1.2 Team Members and Roles

2. Project Overview

- 2.1 Purpose

- 2.2 Key Features

3. Architecture

- 3.1 Front-end (React)

- 3.2 Back-end (Node.js and Express.js)

- 3.3 Database (MongoDB)

4. Setup Instructions

- 4.1 Prerequisites

- 4.2 Installation Steps

- 4.3 Database Configuration

5. Folder Structure

- 5.1 Client (React Front-end)

- 5.2 Server (Node.js Back-end)

6. Running the Application

- 6.1 Starting the Front-end Server

- 6.2 Starting the Back-end Server

- 6.3 Environment Configuration

7. API Documentation

- 7.1 API Endpoints

- 7.2 Request Methods and Parameters

- 7.3 Response Examples

8. Authentication

- 8.1 JWT Authentication

- 8.2 Role-Based Access Control

- 8.3 Token Storage

9. User Interface

- 9.1 Search and Booking Work-flow

- 9.2 User Dashboard

- 9.3 Responsive Design Overview

10. Testing

- 10.1 Front-end Testing

- 10.2 Back-end Testing

- 10.3 Integration Testing

11. Screenshots or Demo

- 11.1 Key Feature Screenshots

- 11.2 Demo Link or Video Walkthrough

12. Known Issues

- 12.1 List of Bugs and Limitations

- 12.2 Workarounds and Fixes

13. Future Enhancements

- 13.1 Additional Filtering Options

- 13.2 Internationalization Support

- 13.4 Loyalty Program

- 13.4 AI-Based Recommendations

Exclusive summary

Flight Booking System is a full-stack web application developed using the MERN stack (MongoDB, Express.js, React, and Node.js).

* Objectives: To provide travelers with a seamless flight booking experience and administrators with efficient management tools.
* Main Features: Includes flight search, personalized recommendations, secure booking, and an admin dashboard with analytic.
* Scope: This report covers the technical architecture, application flow, back-end and front-end development, testing strategies, and future enhancements.

### **INTRODUCTION**

**1.Project Title**: **Flight Booking System:**

* A digital platform designed to simplify the process of finding, booking, and managing flights for users around the world.

**2.Team Members:**

· PONSURESH L: Role (e.g., Lead Back-end Developer) – responsible for API development, server management, and data handling.

· PAVITHRA M: Role (e.g., Front-end Developer) – focuses on building the user interface, including booking forms, search filters, and responsiveness.

· PRASANNA B: Role (e.g., Database Manager) – oversees database schema design, optimization, and data security.

· PAVITHRA MS: Role (e.g., UI/UX Designer) – responsible for designing user-friendly and accessible interfaces.

PROJECT OVERVIEW

1. **Purpose**:

* The flight booking system serves as an online solution for booking flights, allowing users to search for flights based on destinations, dates, and airlines, and to complete bookings securely. The system aims to make travel planning seamless by providing instant availability, clear pricing, and easy access to booking details.

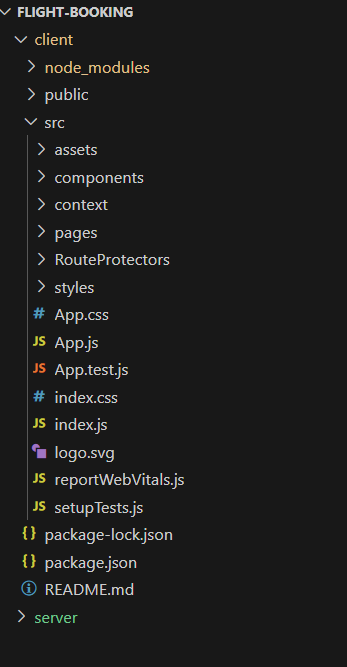
2. **Features**:

* **User Registration and Login:** Users can create an account, login, and manage their profiles.
* **Flight Search**: Search flights by date, destination, price, and other filters for customized experience.
* **Booking Management**: Allows users to view, modify, or cancel their bookings.
* **Payment Integration**: Secure payment gateway integration to handle transactions.
* **Admin Dashboard**: Provides administrative access to manage flights, monitor bookings, and handle customer issues.
* **Notifications**: Email or SMS notifications for booking confirmations, reminders, and flight changes.

ARCHITECTURE

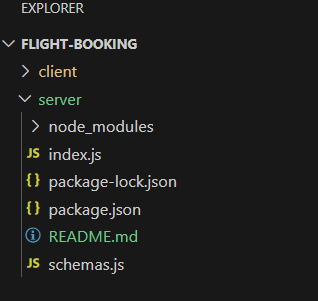
1. **Frontend**:

* Built with **React**, featuring modular components for each function (search, booking, user profile, etc.).
* Implements **React Router** for navigation across pages (e.g., Home, Flight Search, Booking, Profile).
* Uses **Redux** for state management to handle data across components, especially for the booking cart, user data, and search filters.
* **Responsive Design**: Uses CSS frameworks like **Bootstrap** or **Material-UI** for consistent styling and responsive layouts across different devices.



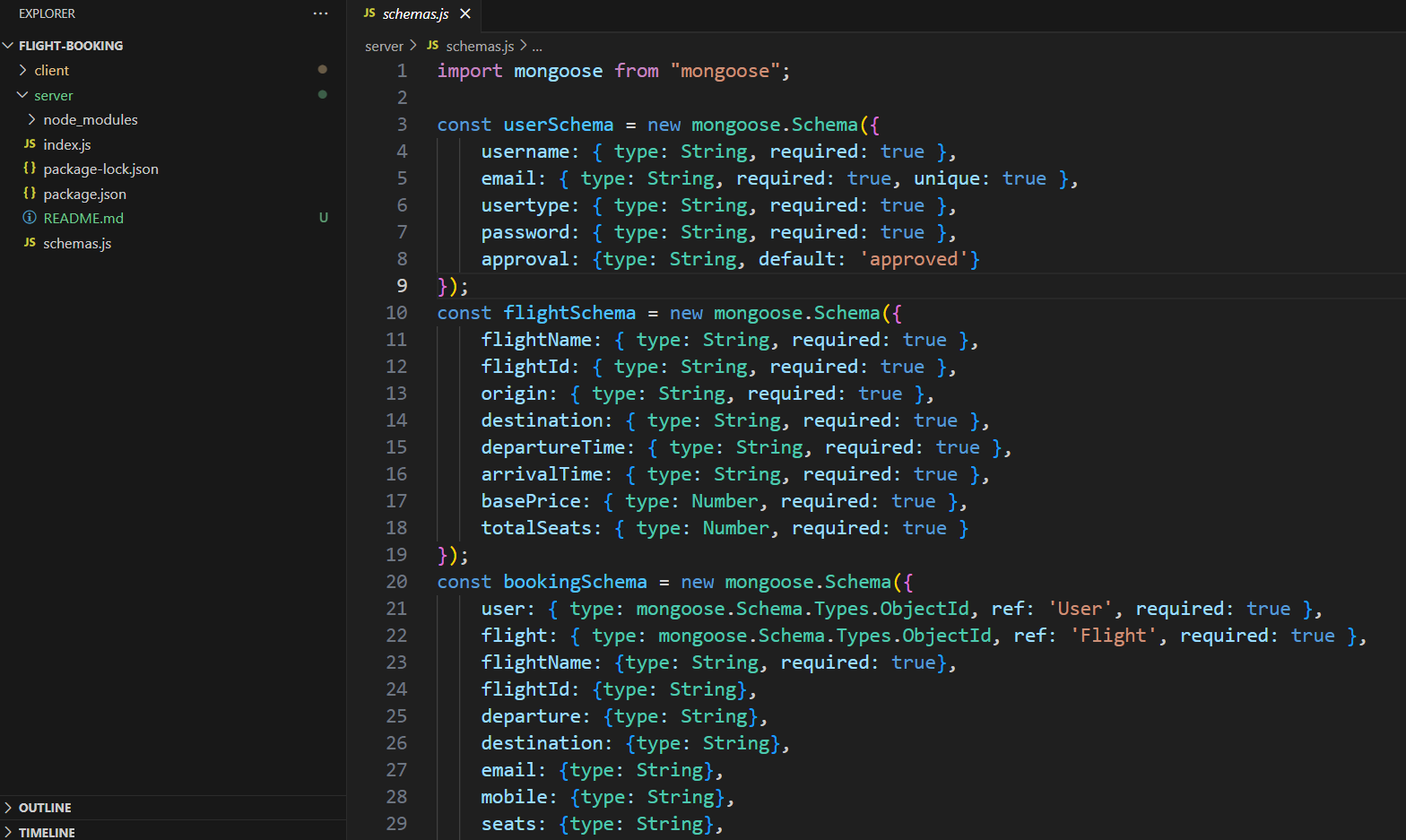
· 2.Back-end:

* Created with **Node.js** and **Express.js** to manage RESTful API requests and connect to the database.
* **API Endpoints**: Provides endpoints for user authentication, searching flights, processing bookings, handling payments, and more.
* **Data Validation**: Uses libraries like **Joi** or **Express-validator** to validate data sent from the frontend.
* **Error Handling and Logging**: Structured error handling to capture user and system errors, with logging mechanisms for debugging.



·3. **Database**:

* Utilizes **MongoDB** as a NoSQL database to manage collections for users, flights, bookings, and transactions.
* **Data Schema**: Defines schemas for each major data entity, including indexes for fast search on common filters like destination and price.
* **Security**: Ensures sensitive data protection through encryption and controlled access.



**SETUP INSTRUCTION**

1. **Prerequisites**:

* **Node.js:and** **npm**: Required to run the backend server and manage dependencies.
* **MongoDB**: Install MongoDB locally or configure a cloud MongoDB Atlas instance.
* **Environment Variables**: Create a .env file for storing sensitive information like database URIs, API keys, and JWT secrets.

2. **Installation**:

* Clone the repository: git clone [repository URL].
* Navigate to project directories (client and server) and install dependencies: npm install.
* Set up environment variables in .env files (e.g., MongoDB URL, JWT secret, payment gateway keys).

**3.Database Configuration**:

* Create MongoDB collections and load initial data as required (e.g., available flights, test user accounts).

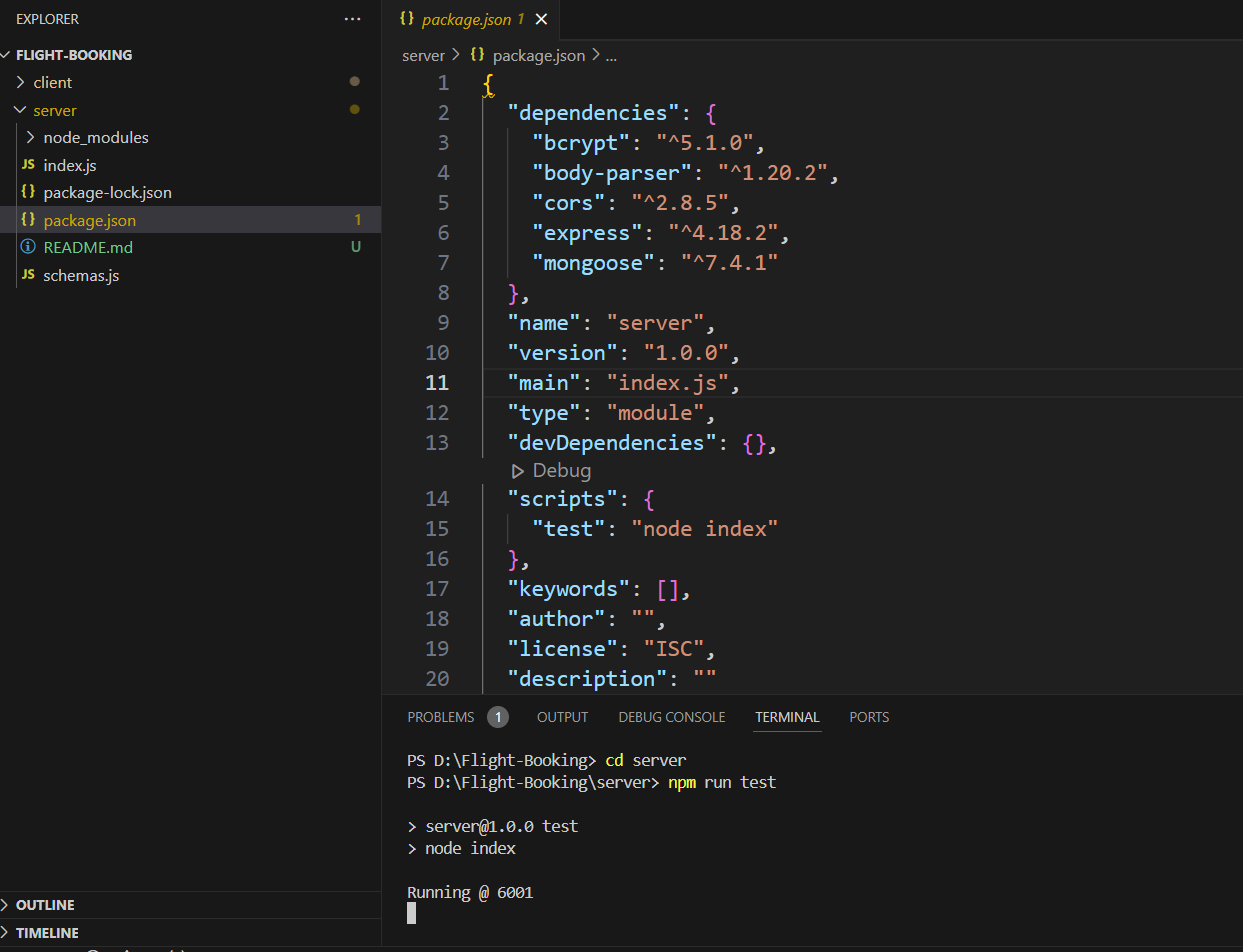
FOLDER STURCURE

1. **Client**:

* **/src/components**: Contains React components for UI elements (e.g., FlightSearch, BookingForm, UserProfile).
* **/src/redux**: Manages application state with actions and reducers for booking and user data.
* **/src/assets**: Holds static assets like images and style files.

2. **Server**:

* **/controllers**: Handles business logic for each route (e.g., managing bookings, user authentication).
* **/routes**: Defines Express routes for endpoints (e.g., /flights, /bookings).
* **/models**: Contains MongoDB schemas for entities like User, Flight, Booking.
* **/middleware**: Houses middleware functions like authentication checks and error handling.



RUNNING THE APPLICATION

· **Frontend**: Run npm start in the client directory to start the React development server.

· **Backend**: Run npm start in the server directory to start the Express server.

· **Environment**:Access the frontend on http://localhost:3000 and the backend on <http://localhost:5000.>

API DOCUMENTATION

1. Provide detailed documentation for each API endpoint:

* **Method**: GET, POST, PUT, DELETE.
* **Endpoint**: The URL path (e.g., /api/flights/search).
* **Parameters**: Query or body parameters (e.g., destination, date, priceRange).
* **Responses**: Example responses for success and error cases.

AUTHENTICATION

· **JWT Authentication**: Securely manages user sessions using JSON Web Tokens (JWT).

· **User Roles**: Implements role-based access (e.g., users and admins) to restrict access to certain features.

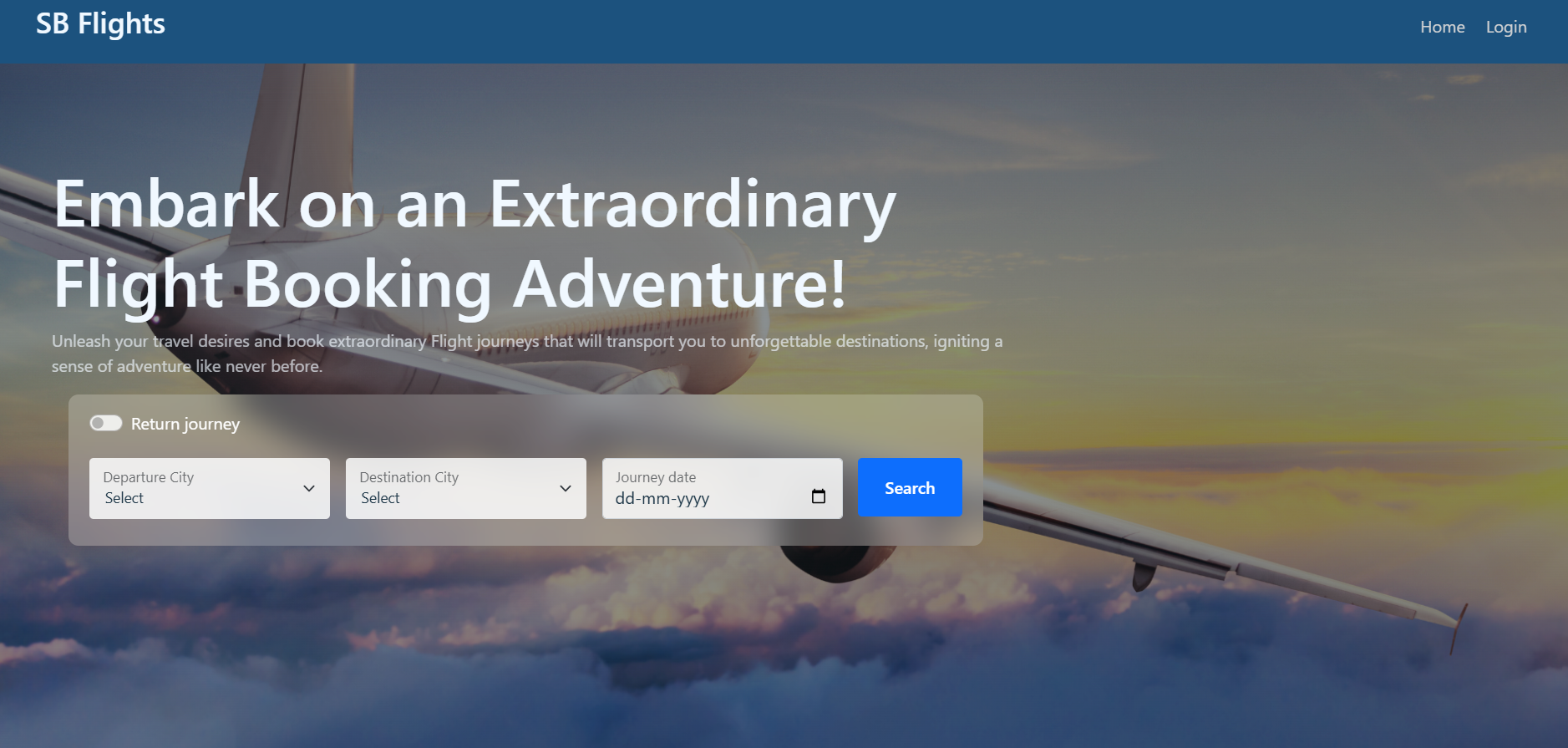
· **Token Storage**: Stores tokens securely in cookies or local storage on the client side.

USER INTERFACE

· **Search and Booking Workflow**: Screenshots showing the flow from searching flights to booking.

· **User Dashboard**: Highlights profile and booking management features.

· **Responsive Design**: Demonstrates views on mobile, tablet, and desktop screens.



TESTING

· Front-end **Testing**: Uses **Jest** and **React Testing Library** for component testing.

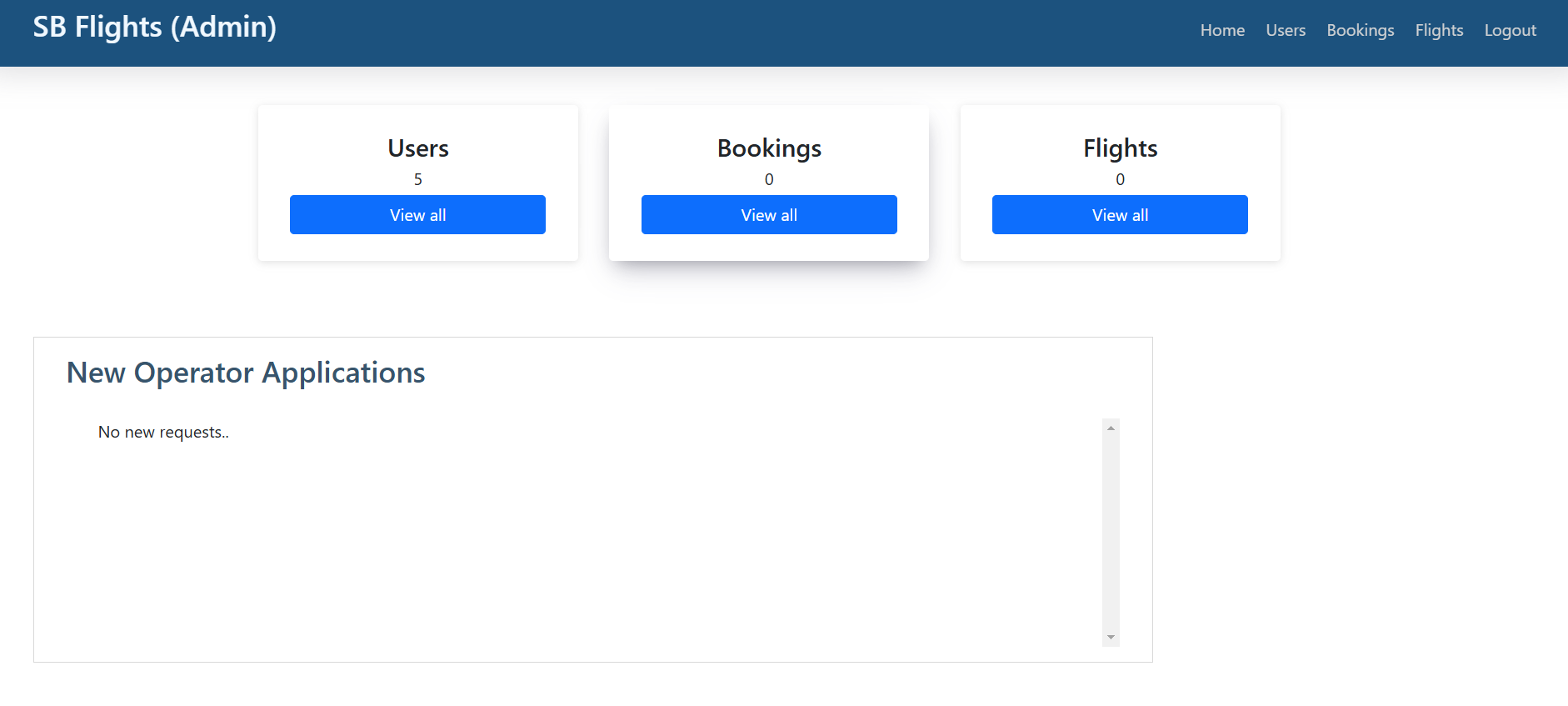
· **Backend Testing**: Utilizes **Mocha**, **Chai**, and **Supertest** for API testing.

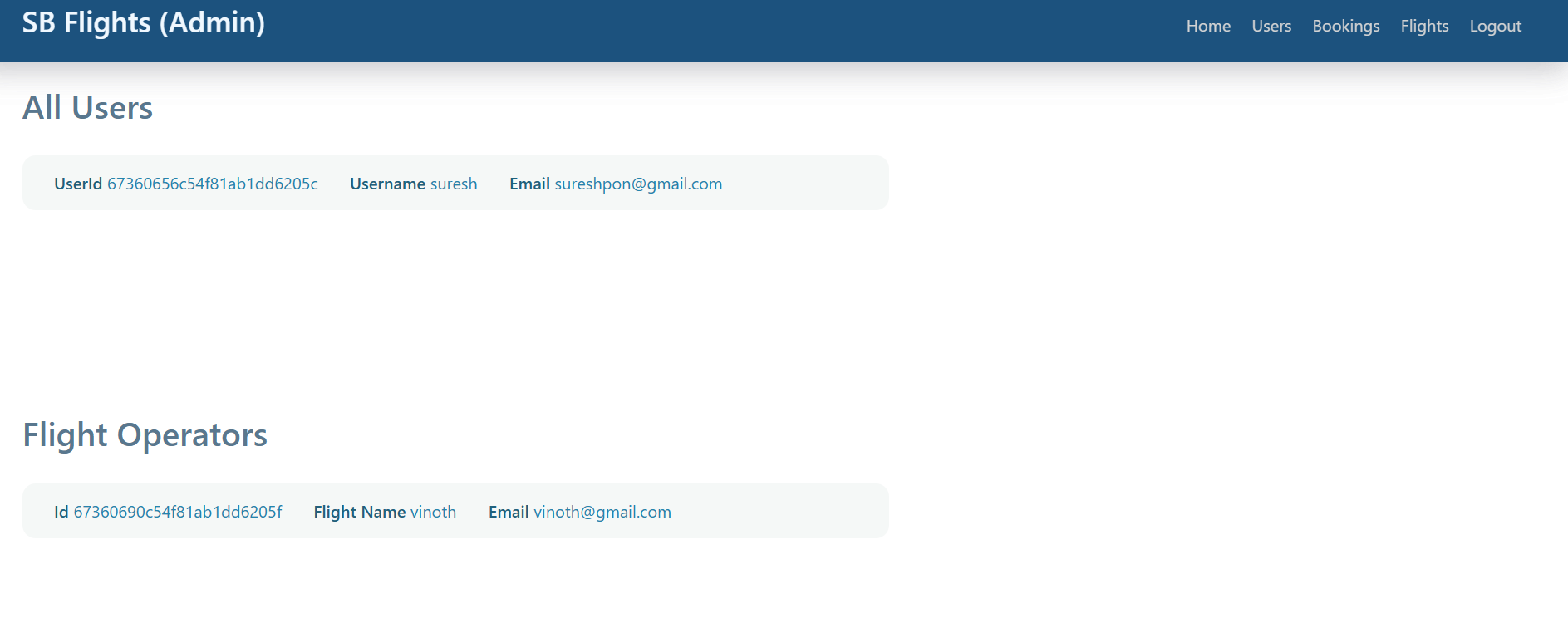
· **Integration Tests**: Tests the full user flow, ensuring both frontend and backend interactions work as expected.

SCRENNSHOTS OR DEMO

· **Screenshots**: Provide images for each major feature (e.g., flight search results, booking confirmation).

· **Demo Link**: Include a link to a live demo or a video walkthrough.





KNOWN ISSUES

· **List of Bugs**: Include unresolved bugs or limitations (e.g., specific edge cases that may not yet be handled).

· **Workarounds**: Provide potential fixes or steps to address known issues.

FUTURE ENHANCEMENT

· **Improved Filtering Options**: Enable users to filter flights based on layovers, airlines, and amenities.

· **Internationalization**: Add support for multiple languages to make the application more globally accessible.

·  **Loyalty Program**: Implement a loyalty points system for frequent users.

· **AI-Based Recommendations**: Use machine learning to suggest flights based on user preferences and past bookings.