Here's the **Final Project Report** for your **Flight Finder** application based on the details you provided:

🛫 FINAL PROJECT REPORT – FLIGHT FINDER

1. PROJECT OVERVIEW

Field	Detail
Project Name	Flight Finder
Team Members	Meghana, Gurijala Venkata Sai Sanjana, Donthi Nithish Sai Guptha, Mudipalli Kalpana
Start Date	20th May 2024
End Date	30th June 2024
Mentor/Guide	SmartInternz
Team ID	LTVIP2025TMID58721
GitHub Repo	Flight Finder GitHub
Team Leader	Devarapalli Lakshmi Meghana
Email	meghana311204@gmail.com

2. OBJECTIVE

To develop a responsive and functional **Flight Booking Application** using the **MERN Stack** that allows customers to search and book flights, while operators manage flight data, and admins approve/decline new users.

3. ROLES AND RESPONSIBILITIES

Role	Capabilities
Customer	Register, log in, search flights, book tickets

Role	Capabilities
Operator	Add, edit, delete flights; view bookings
Admin	Approve/decline new customer/operator registrations; oversee system

4. FUNCTIONAL REQUIREMENTS

Flight Search: Source, destination, travel date

Flight Filters: By price, airline

Booking Page: Minimal implementation (optional)

Authentication: Secure login for all user roles

Admin Panel: Approve/reject users, manage flights

Responsive UI: Clean and mobile-friendly interface

5. TECHNOLOGY STACK

Layer	Technology
Frontend	React JS, HTML, CSS, JavaScript
Backend	Node.js, Express.js
Database	MongoDB (via Mongoose)
Authentication	JWT (JSON Web Tokens)
Deployment	GitHub, optional: Render/Netlify/Vercel/MongoDB Atlas

6. IMPLEMENTATION PLAN (SPRINT-WISE)

Sprint	Duration	Key Tasks
1	Week 1	Requirement gathering, planning, mockup UI
2	Week 2	React setup, login/register, flight search page
3	Week 3	Express API for authentication and bookings
4	Week 4	MongoDB data connection and model creation

Sprint	Duration	Key Tasks
5	Week 5	Admin dashboard and operator CRUD for flights
6	Week 6	Testing, debugging, documentation, and deployment

7. ER DIAGRAM (HIGH-LEVEL OVERVIEW)

```
Users
│
├── userId (PK)
├── name
├── email
├── password
├── role (customer/operator/admin)

Flights
│
├── flightId (PK)
├── airline
├── source
├── destination
├── departureTime
├── arrivalTime
├── price
├── seatsAvailable

Bookings
│
├── bookingId (PK)
├── userId (FK)
├── flightId (FK)
├── bookingDate
├── status
```

8. DATA FLOW DIAGRAM (DFD – LEVEL 1 OVERVIEW)

```
[Customer] → [Login/Register] → [Flight Search] →
[Booking]
```

```
                                    ↓
                              [Backend API]
                                    ↓
                              [MongoDB DB]

[Operator] → [Login] → [Add/Edit/Delete Flights]
                                    ↓
                              [Backend API]

[Admin] → [Approve/Decline Users] → [Manage Operators/
Customers]
```

9. USER STORIES

As a customer, I want to search for flights using origin, destination, and date so I can find the best options.

As an operator, I want to manage flights (add, edit, delete) so that I can keep flight data up to date.

As an admin, I want to approve or decline new users to control access to the system.

10. TEST PLAN

Test Case	Expected Outcome	Result
Invalid login	Show error message	✅
Successful customer registration	Redirect to login	✅
Booking with invalid inputs	Block submission	✅
Admin route access by customer	Denied with 403 error	✅
Operator creates valid flight	Flight visible to customers	✅
Duplicate booking	Prevent duplicate record	✅

11. DEBUGGING & VALIDATION

Postman used for backend API testing.

React Form Validation to block invalid inputs.

Error Logging with try-catch and console tracing.

Manual testing of:

Booking flows

Route protection

Flight filters and CRUD

12. DEPLOYMENT (SUGGESTED STEPS)

Push frontend to GitHub.

Deploy frontend via Vercel/Netlify.

Push backend code to GitHub.

Deploy backend via Render or Railway.

Use MongoDB Atlas for cloud DB access.

Add environment variables:

JWT_SECRET

DB_URL

Final integration testing post-deployment.

13. CONCLUSION

The **Flight Finder** project is a successful full-stack MERN application developed for the SmartInternz internship. It meets all functionality goals:

Efficient **customer experience**

Clear **operator management**

Secure **admin control**

The app combines real-world usability with clean architecture, demonstrating the team’s ability to execute a full lifecycle product from planning to deployment.