Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	31 January 3035
Team ID	LTVIP2025TMID46399
Project Name	Flight Finder
Maximum Marks	4 Marks

✈️ *Flight Finder App – Technical Architecture*

📌 **Overview:**

The Flight Finder App is a full-stack web application designed to allow users to search, filter, book, and manage flight journeys efficiently. It incorporates a modular, secure, and scalable architecture that leverages open-source frameworks and cloud-ready infrastructure.

Table-1: Components & Technologies

S.No	Component	Description	Technology Used
1.	User Interface	Web UI for users to search, filter, and book flights	HTML, CSS, JavaScript, React.js
2.	Application Logic-1	Handles user authentication, session, and validation	Node.js, Express.js
3.	Application Logic-2	Flight search, filter, booking, seat selection, payment	Node.js, RESTful APIs
4.	Application Logic-3	Admin dashboard & flight management	Node.js, Express Admin Routes
5.	Database	Data storage for users, flights, bookings	MongoDB (NoSQL)
6.	Cloud Database	Cloud-hosted database storage	MongoDB Atlas

S.No	Component	Description	Technology Used
7.	File Storage	Stores documents or seat maps (if any)	Cloudinary / AWS S3 / Local File System
8.	External API-1	Payment processing API	Razorpay / Stripe API
9.	External API-2	Flight data or airline integration	Amadeus API / Skyscanner API (optional)
10.	Machine Learning Model	(Optional) Smart recommendations for flights	Recommendation model using Python / TensorFlow (future scope)
11.	Infrastructure	Application deployed on the cloud	Localhost (dev), Render / Vercel / Heroku / AWS (prod)

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology Used
1.	Open-Source Frameworks	Uses open-source tools for frontend and backend development	React.js, Node.js, Express.js, MongoDB
2.	Security Implementations	Password hashing, secure authentication, secure payment gateway	Bcrypt, JWT, HTTPS, CORS, OAuth (Gmail, LinkedIn)
3.	Scalable Architecture	Modular 3-tier architecture (Frontend–Backend–Database)	RESTful APIs, MVC Pattern, Cloud deployment ready
4.	Availability	Hosted on scalable platforms with near 99.9% uptime, fault tolerance	Render / Vercel / AWS EC2, Load balancing (optional)
5.	Performance	Fast API responses, optimized queries, client-side routing, and caching	React, MongoDB Indexes, Node.js Clustering

Architecture Diagram Suggestion (C4 or Block Style)

[User] --> [React Frontend] --> [Node.js API Server] --> [MongoDB Atlas]

|--> [Admin Routes]

|--> [Payment API (Razorpay)]

|--> [External Flight Data API (Optional)]

References

- https://c4model.com/
- https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/
- https://www.ibm.com/cloud/architecture
- https://aws.amazon.com/architecture
- https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d