**Project Design Phase**

**Solution Architecture**

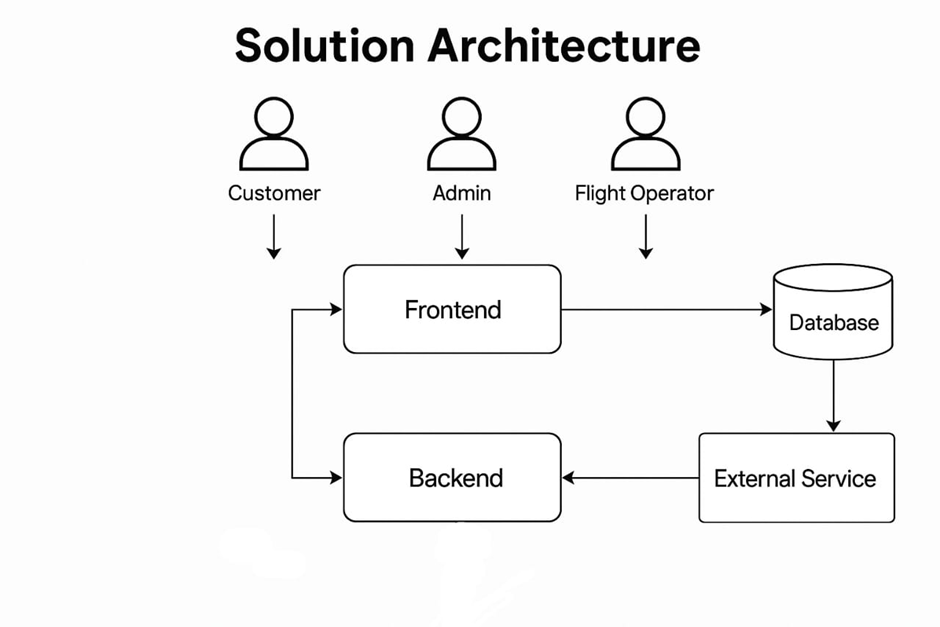
|  |  |
| --- | --- |
| Date | 30-06-2025 |
| Team ID | LTVIP2025TMID43096 |
| Project Name | Flight Finder: Navigating Your Air Travel Options |
| Maximum Marks | 4 Marks |

**Solution Architecture:**

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

* Find the best tech solution to solve existing business problems.
* Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
* Define features, development phases, and solution requirements.
* Provide specifications according to which the solution is defined, managed, and delivered.

**Solution Architecture of our project:-**

****

**A diagram of a computer

AI-generated content may be incorrect.**

The Flight Booking System follows a modular and scalable three-tier architecture designed for flexibility, maintainability, and ease of deployment. The system consists of:

1.Frontend (React.js)

The client-side is developed using React.js.

It provides an interactive UI for three user roles: Customer, Admin, and Flight Operator.

React Router is used for route management and navigation between pages.

Axios handles API calls to the backend.

2.Backend (Node.js + Express.js)

The server-side is built with Express.js, exposing RESTful APIs.

It includes logic for authentication, role-based access control, flight and booking management, and operator approval.

Middleware like body-parser, cors, and dotenv are used for secure and efficient request handling.

3.Database (MongoDB with Mongoose)

MongoDB stores user data, flight information, and booking records.

Mongoose ODM provides schema validation, easy querying, and relationship mapping between documents (e.g., linking a booking to a user and flight).

4.User Roles & Flow

Customer: Can search, book, and cancel flights.

Flight Operator: Can add and manage their own flights.

Admin: Has privileges to approve or reject operator requests and view system-wide bookings/users.

5.Data Flow

Frontend sends requests via Axios to the backend.

Backend validates and processes requests, then interacts with MongoDB.

Responses are sent back and dynamically displayed on the UI.

This architecture ensures a clear separation of concerns, enabling collaborative development, efficient debugging, and easy scaling of individual components.