**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

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

**Technical Architecture:**

**The technical architecture of the Flight Booking System is based on a modern MERN stack (MongoDB, Express.js, React.js, Node.js), designed to ensure scalability, maintainability, and seamless user experience across multiple user roles: Customer, Flight Operator, and Admin.**

**1. Frontend (Client Layer)**

* **Built using React.js, it provides a responsive, interactive UI.**
* **Separate dashboards for each role:**
  + **Customers can search and book flights.**
  + **Flight Operators can manage flights and view bookings.**
  + **Admins approve or reject operator applications and monitor system stats.**
* **Communicates with backend via Axios over HTTP.**

**2. Backend (Server Layer)**

* **Developed using Node.js and Express.js.**
* **Provides RESTful APIs for:**
  + **User authentication (login/register for all roles)**
  + **Flight search, add, update, and delete**
  + **Booking and ticket cancellation**
  + **Admin approvals**
* **Middleware handles validation, error handling, and role-based access control.**

**3. Database Layer**

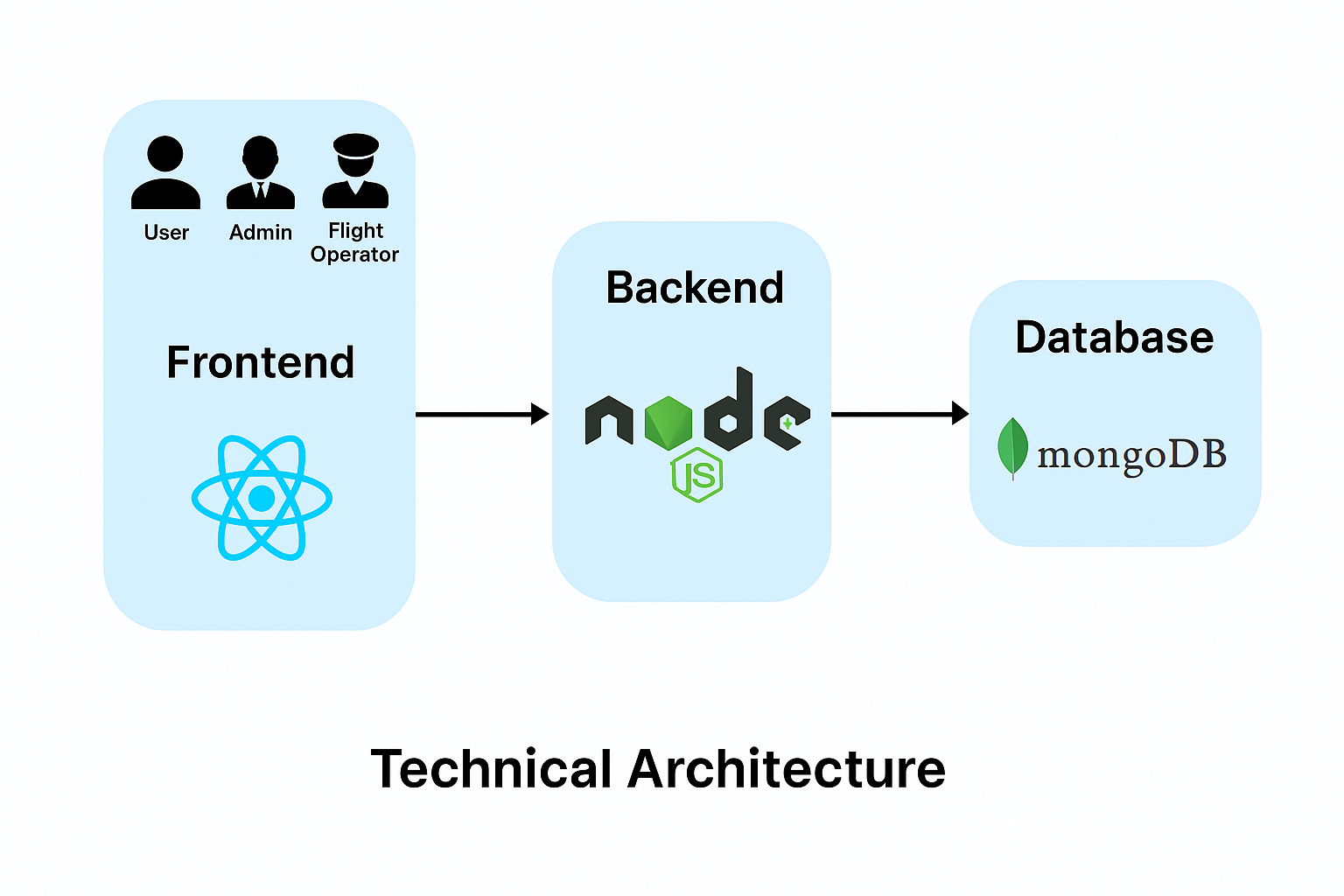
* **MongoDB is used as the primary database, managed via Mongoose ORM.**
* **Collections:**
  + **Users: Stores customer, operator, and admin details.**
  + **Flights: Stores all available flights.**
  + **Bookings: Stores booking records with references to user and flight.**

**4. Authentication & Authorization**

* **User sessions managed using JWT (JSON Web Tokens).**
* **Role-based access ensures that:**
  + **Customers can only book and view their bookings.**
  + **Operators manage only their flights/bookings.**
  + **Admins have full access.**

**5. Deployment & Hosting**

* **Frontend can be deployed on Netlify or Vercel.**
* **Backend on Render or Heroku.**
* **MongoDB hosted on MongoDB Atlas.**
* **Environment variables managed using .env files to ensure security.**

****

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Interface for users (customer/admin/operator) to interact with system | HTML, CSS, JavaScript, React.js |
|  | Application Logic-1 | Logic for user authentication and role-based access | Node.js, Express.js |
|  | Application Logic-2 | Logic for booking flights, handling cancellations | Node.js, Express.js |
|  | Application Logic-3 | Logic for flight operator approval, flight creation | Node.js, Express.js |
|  | Database | Storage of users, flights, bookings data | MongoDB |
|  | Cloud Database | Hosted version of MongoDB | MongoDB Atlas |
|  | File Storage | Not applicable (no file uploads) | - |
|  | External API-1 | Calendar/date inputs, location lists (if extended) | (Optional) e.g., Google Maps API |
|  | External API-2 | N/A | - |
|  | Machine Learning Model | Not used | - |
|  | Infrastructure (Server / Cloud) | Hosting frontend/backend | Localhost (development) or Render/Netlify/Vercel |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | All components built using open-source frameworks | React.js, Node.js, Express.js, MongoDB |
|  | Security Implementations | Password hashing, role-based access, CORS | bcrypt, JWT (optional), Helmet, CORS |
|  | Scalable Architecture | Logic is modular and RESTful; can be scaled using services/microservices | Node.js with Express routing |
|  | Availability | Can be deployed on cloud platforms for high availability | MongoDB Atlas, Render/Netlify (optional) |
|  | Performance | Efficient use of API calls, optimized data retrieval | Axios, Mongoose query optimization |

**References:**

[**React.js Documentation**](https://react.dev/)

[**JSON Web Server Referance**](https://www.npmjs.com/package/json-server)

[**Node js Best Practice**](https://nodejs.org/en/learn/getting-started/introduction-to-nodejs)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)