**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 26-05-2025 |
| Team ID | LTVIP2025TMID43459 |
| Project Name | FlightFinder |
| Maximum Marks | 4 Marks |

***Technical Architecture:***

The FlightFinder platform is designed using a scalable 3-tier architecture that separates the presentation, application logic, and data layers. This architecture ensures modularity, performance, and security, while allowing seamless integration with third-party APIs for features like notifications and real-time communication.

The three tiers are:

* Presentation Layer (Frontend): React.js interface for users, agents, and admins
* Business Logic Layer (Backend): Node.js & Express.js APIs handle booking logic, authentication, and assignment workflows
* Data Layer (Database): MongoDB is used for flexible, schema-driven storage of user profiles, bookings, messages, and feedback

|  |  |  |  |
| --- | --- | --- | --- |
| **S. NO.** | **Components** | **Description** | **Technology** |
| **1.** | User Interface | Web-based interface for users, agents, and admins | React.js, HTML, CSS, JavaScript |
| **2.** | Application Logic – 1 | Booking submission, user login, status updates | Node.js, Express.js |
| **3.** | Application Logic – 2 | Admin and agent dashboards, role-based access | React.js, Node.js |
| **4.** | Database | Stores user info, bookings, messages, feedback | MongoDB |

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

Each component of FlightFinder is built using modern, widely-used open-source technologies that enable rapid development, easier maintenance, and strong community support. The frontend is crafted using React.js for fast rendering and state management, while the backend APIs built on Node.js and Express handle core booking workflows securely. MongoDB provides the flexibility needed to store structured and semi-structured booking data, chat logs, and user profiles efficiently.

***Table-2: Application Characteristics:***

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Components** | **Description** | **Technology** |
| 5 | Open-Source Frameworks | Use of modern frontend/backend frameworks | React.js, Node.js, Bootstrap, Tailwind CSS |
| 6 | Scalable Architecture | Tiered design allows flexible scaling of components | Microservices, RESTful APIs |
| 7 | Role-Based Security | Access managed based on roles (User, Agent, Admin) | JWT (JSON Web Token), Role Middleware |
| 8 | API Integration Ready | Easily integrates with external services (e.g., email APIs) | Axios, Nodemailer, Webhooks |

The application architecture follows best practices in modular software design, making it scalable, secure, and easy to maintain. By leveraging open-source frameworks and a microservices approach, the platform can be expanded in future releases (e.g., mobile apps, AI-based booking categorization). It also ensures clean separation of concerns, enabling better collaboration between frontend and backend teams.

**References:**

* [React.js Official Documentation](https://reactjs.org/)
* [Node.js Best Practices Guide](https://github.com/goldbergyoni/nodebestpractices)
* [JWT Authentication Concepts](file:///C:\Users\saiha\AppData\Local\Microsoft\Windows\INetCache\IE\JA4CW89I\•%09JWT%20Authentication%20Concepts)
* [How to Draw Technical Architecture Diagrams](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)