**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 15 JUNE 2025 |
| Team ID | LTVIP2025TMID46308 |
| Project Name | DOC SPOT |
| Maximum Marks | 4 Marks |

Technical Architecture:

DocSpot leverages the **MERN Stack**—**MongoDB**, **Express.js**, **React.js**, and **Node.js**—to create a full-stack, modern, and scalable healthcare appointment booking system. This architecture enables seamless interaction between users (patients, doctors, and administrators) while ensuring fast performance, security, and extensibility.

DocSpot is designed with modularity and real-world reliability in mind. For instance, during high-demand periods such as health crises or pandemics, it can be adapted for offline data handling or extended to support telemedicine features.

**🌐 Frontend (Client-Side)**

* **Technology Used**: React.js
* **Key Features**:
  + Built as a **Single Page Application (SPA)** for a seamless user experience.
  + Uses **Axios** for HTTP requests to interact with backend APIs.
  + Incorporates **Material UI** and **Bootstrap** for responsive, clean, and accessible UI components.

**🖥️ Backend (Server-Side)**

* **Technology Used**: Node.js with Express.js
* **Core Responsibilities**:
  + Handles **API routing**, **user authentication**, and **data validation**.
  + Manages **appointment scheduling** and role-based logic for patients, doctors, and admins.
  + Implements **secure middleware** and **JWT-based authentication** for protected route access.

**📦 Database Layer**

* **Technology Used**: MongoDB with Mongoose ODM
* **Role**:
  + Stores user profiles, doctor availability, appointment data, and admin records.
  + Schema-based structure ensures data consistency and validation.
  + Cloud-hosted using **MongoDB Atlas** for high availability and easy scaling.

**Architecture Breakdown – Tables**

**Table 1: Components & Technologies**

| **S.No** | **Component** | **Description** |
| --- | --- | --- |
| 1 | User Interface | Web & mobile-responsive UI for patients, doctors, and admins |
| 2 | Application Logic (Part 1) | Booking appointments, setting doctor availability, sending reminders |
| 3 | Application Logic (Part 2) | Admin panel with user management, doctor verification, and system analytics |
| 4 | Database | Stores persistent data: user profiles, appointments, availability, approvals |

**Table 2: Application Characteristics**

| S.No | Characteristic | Description | **Technology** |
| --- | --- | --- | --- |
| 1 | Open-Source Frameworks | Leverages robust JS frameworks and libraries | React.js, Node.js, Bootstrap, Tailwind CSS |
| 2 | Scalable Architecture | Layered 3-tier model with RESTful API communication | Microservices-compatible backend for future enhancements |
| 3 | Role-Based Access Control | Distinct access for patient, doctor, and admin functionalities | JWT + Express middleware |

**Security and Best Practices**

* Secure data transmission using **HTTPS**
* Authentication handled via **JWT (JSON Web Token)**
* Input sanitization and error handling to prevent injection attacks
* Role-based middleware protects sensitive routes

**Scalability Potential**

* Built to support future integrations:
  + Video consultations
  + E-prescriptions
  + Payment gateways
  + Progressive Web App (PWA) or mobile app extensions