**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 16 February 2026 |
| Team ID | LTVIP2026TMIDS35442 |
| Project Name | DocSpot: Seamless Appointment Booking for Health |
| Maximum Marks | 4 Marks |

**Solution Architecture:**

DocSpot follows a three-tier client-server architecture with clear separation between presentation, application, and data layers.

**Architecture Overview**

| **Layer** | **Component** | **Description** |
| --- | --- | --- |
| Frontend | React.js | User interface for Patients, Doctors, and Admin. Communicates with backend via RESTful APIs using Axios. |
| Backend | Express.js | Handles business logic, authentication, request processing, and API routing with JWT-based security. |
| Database | MongoDB Atlas | Cloud NoSQL database storing users, doctor profiles, and appointments using Mongoose ODM. |

**Key Components**

**Frontend (React.js)**

* **Components:** Modular UI for each user role (patient, doctor, admin)
* **React Router:** Client-side routing with protected routes based on user roles
* **Axios:** HTTP client for API communication with token attachment
* **Context API:** Global state management for authentication
* **Bootstrap:** Responsive UI components

**Backend (Express.js)**

* **Middleware:** CORS, JSON parsing, JWT authentication, Multer for file uploads
* **Routes:** API endpoints for auth, doctors, appointments, admin
* **Controllers:** Business logic for each entity
* **Models:** Mongoose schemas for Users, Doctors, Appointments
* **JWT:** Token-based authentication with role-based access control

**Database (MongoDB Atlas)**

* **Users Collection:** Stores user credentials, roles, notifications
* **Doctors Collection:** Stores doctor profiles, qualifications, fees, timings
* **Appointments Collection:** Stores booking details, status, document references
* **Indexes:** Optimized queries on email, specialization, status, dates

**Authentication Flow**

1. User registers → Password hashed with bcrypt
2. User logs in → JWT token generated
3. Token stored in localStorage → Sent in Authorization header
4. Middleware verifies token → Grants access based on role

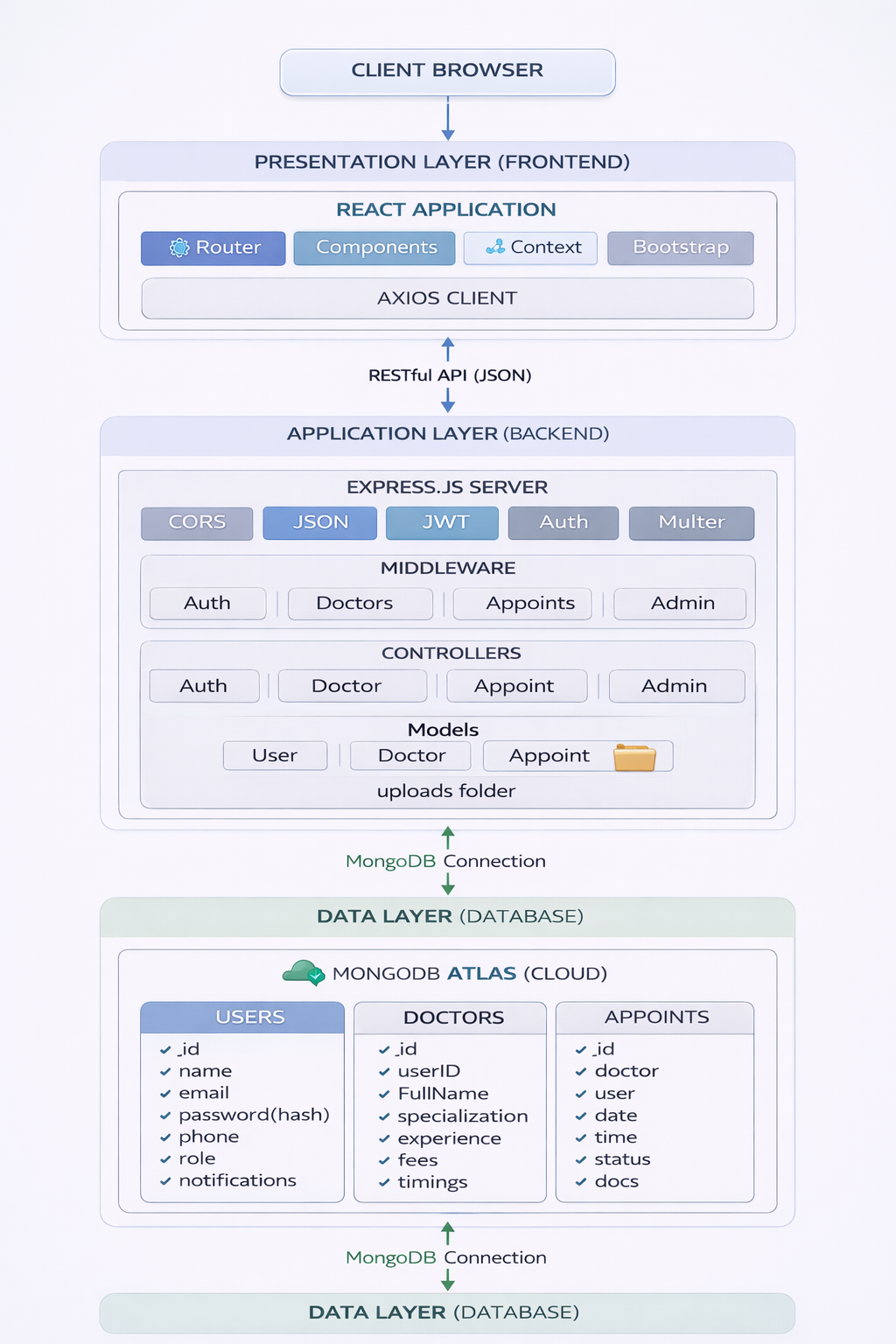
**File Upload Flow**

1. Multer processes multipart form data
2. Files validated (type/size) and stored in uploads folder
3. File paths saved in appointment documents
4. Express serves static files for access

**Data Flow**

1. User action → React component → Axios request
2. Express middleware → Route handler → Controller
3. Controller → Mongoose model → MongoDB
4. Response → React component → UI update

**Example - Solution Architecture Diagram:**

****