**How to Run the Contact Management Application**

**Prerequisites**

Ensure the following are installed on your system:

* **Node.js** (v16 or higher)
* **MongoDB** (installed locally or using a cloud service like MongoDB Atlas)
* **Git** (to clone the repository)
* **Setup Instructions**
* **1. Clone the Repository**
* bash
* Copy code
* git clone <repository-url>
* cd contact-management

**2. Backend Setup**

1. **Navigate to the Backend Directory**:

bash

Copy code

cd backend

1. **Install Dependencies**:

bash

Copy code

npm install

1. **Configure Environment Variables**:
   * Create a .env file in the backend directory.
   * Add the following:

env

Copy code

PORT=5000

MONGO\_URI=mongodb://localhost:27017/contact\_management

* + Replace mongodb://localhost:27017/contact\_management with your MongoDB connection string if you are using a remote database.

1. **Start the Backend Server**:

bash

Copy code

npm run dev

* + The backend server will run at http://localhost:5000.

**3. Frontend Setup**

1. **Open a New Terminal Window** and navigate to the frontend directory:

bash

Copy code

cd ../frontend

1. **Install Dependencies**:

bash

Copy code

npm install

1. **Start the React Application**:

bash

Copy code

npm start

* + The React app will run at http://localhost:3000.

**4. Testing the Application**

1. **Access the Application**:
   * Open a browser and navigate to http://localhost:3000.
   * Use the form to add, edit, or delete contacts, and view the updated contact list.
2. **Database Verification**:
   * If using **MongoDB Compass**, connect it to your MongoDB instance to view the contact data being stored.
3. **Backend Logs**:
   * Check the terminal running the backend to see if the API requests are being handled correctly.

**Project Overview**

**Purpose**

The Contact Management Application is a simple CRUD (Create, Read, Update, Delete) application designed to manage contact details. It uses a **React frontend**, an **Express.js backend**, and **MongoDB** for the database.

**Major Technical Decisions**

* MongoDB was chosen for its flexibility in handling dynamic schemas for contact data.
* The application follows a RESTful API structure to enable modularity and scalability.
* React's Material-UI provides a modern and responsive user interface.

**Application Workflow**

1. **Frontend**:
   * React components handle form submissions and display contact data dynamically.
   * Axios is used to send HTTP requests to the backend.
2. **Backend**:
   * Express routes handle CRUD operations with MongoDB.
   * Mongoose provides schema-based validation for the contact data.
3. **Database**:
   * MongoDB stores contact information with the following fields:
     + firstName, lastName, email, phone, company, jobTitle.

Directory Structure

contact-management/

├── backend/

│ ├── models/ # Mongoose schemas

│ ├── routes/ # Express API routes

│ ├── .env # Backend configuration

│ ├── server.js # Main backend server file

│ └── package.json # Backend dependencies

├── frontend/

│ ├── src/

│ │ ├── components/ # React UI components

│ │ ├── App.js # Main React application file

│ │ └── index.js # React entry point

│ └── package.json # Frontend dependencies

└── README.md