**SDE Internship Assignment**

**1.Why MongoDB is taken as Database?**

**\* NoSQL database, is schema-flexible easy to add data**

**\* Easy to use and excellent document is available**

**\* my previous experience during the college days with Mongodb**

**\* simple and avoid complexity in tables like relationships of tables**

**Instruction To Run The Project**

**Backend**

**1.open the project navigate to the backend**

**1.1 cd ContactManagement/Backend**

**1.2 npm install**

**2.** **In the Backend folder, create a .env file to store the environment**

**Variables MONGODB\_URI=mongodb://localhost:27017/contact\_management**

**Note:You have to install Mongodb before doing this**

**3.npm run dev => AS I AM USING NOT NODEMON FOR THE PROJECT**

**Frontend**

**1.** Now to install the frontend

**1.1cd ContactManagement/Frontend/contact\_managemnet**

**1.2npm install**

**2.** **npm start**

**Project Description**

**Contact management is a full stack project with react js as frontend,express js as**

**Backend and mongodb as database;**

**It is a simple project where you can adding, viewing, editing, and deleting them in a user-friendly interface.**

**Major Technical Decisions**

1. **React with Material UI: I have no knowledge regarding material ui and first time knowing about it as it is mentioned in the description of the project I had to use that which was challenging**
2. **Express js : These technologies handle the backend as they provide an efficient runtime and a minimal setup for building RESTful APIs.**

**How Each Part of the App Works**

**1. Frontend (React + Material UI)**

**1 App.js: The main component of the frontend that manages state and renders the primary components: the contact form and the contacts table.**

**2 AddContactForm.js: This component provides a form for adding new contacts and editing existing contacts. It takes advantage of React's controlled components to manage form state and validate user input.**

**3.ContactsTable.js: Displays a list of contacts in a table format. It includes buttons to edit and delete contacts, leveraging Material UI’s Table and Button components for a clean design.**

**4.Conditional Rendering: Based on whether a contact is selected for editing, the App component conditionally renders the AddContactForm with pre-filled data for editing or as an empty form for new entries.**

**2. Backend (Node.js + Express)**

* **app.js: The main file that initializes the Express server and sets up middleware.**
* **routes/contactRoutes.js: Defines routes for contact-related API endpoints (GET, POST, PUT, DELETE) and associates them with the appropriate controller functions.**
* **controllers/contactController.js: Houses the logic for each route, including functions for adding, retrieving, updating, and deleting contacts. Each function interacts with the MongoDB database to carry out its operations.**
* **models/contactModel.js: Defines the MongoDB schema for contacts, including fields such as firstName, lastName, email, phone, etc. This model interacts with MongoDB through Mongoose for data persistence.**

**3. Database (MongoDB)**

**MongoDB Collection: A collection named contacts stores each contact document. Each document follows the schema defined in the contactModel.js file, capturing essential fields like name, email, phone, company, and job title.**

**Data Validation: MongoDB handles data storage, and Mongoose helps enforce schema validation for fields like email and phone to ensure data consistency.**