3. Contact Manager

1. User models

🧾 Updated Table for `userSchema`

Field Name	Type	Required	Description
`name`	String	✅ Yes	Full name of the user
`email`	String	✅ Yes	Unique email in lowercase
`password`	String	✅ Yes	Hashed user password
`role`	String	❌ No	Either `'user'` or `'admin'` (default: `'user'`)
`createdAt`	Date	Auto	Automatically set on creation
`updatedAt`	Date	Auto	Automatically updated on modification

✅ User API Endpoints (Authentication + Account Management)

Method	Route	Auth Required	Description
`POST`	`/api/auth/register`	❌ No	Register a new user

`POST`	`/api/auth/login`	❌ No	Login and get JWT token
`POST`	`/api/auth/logout`	✅ Yes (client-side only)	Logout by clearing token on frontend
`GET`	`/api/users/me`	✅ Yes	Get logged-in user's profile
`PUT`	`/api/users/me`	✅ Yes	Update user profile (name, email)
`PUT`	`/api/users/password`	✅ Yes	Change password
`DELETE`	`/api/users/me`	✅ Yes	Delete logged-in user account

2. Profile (child) → User (parent) (1 to 1)

✅ Profile Model – Data Fields

Field Name	Type	Required	Description
`user`	ObjectId	✅ Yes	Reference to the associated `User` (_id)
`bio`	String	❌ No	Short user biography or about section
`avatarUrl`	String	❌ No	URL to the user's profile picture
`phone`	String	❌ No	User's phone number
`address`	String	❌ No	Physical or mailing address

🔁 Profile API List

Method	Route	Auth Required	Description
`GET`	`/api/profile/me`	✅ Yes	Get logged-in user's profile
`POST`	`/api/profile`	✅ Yes	Create profile *(if not already exists)*
`PUT`	`/api/profile`	✅ Yes	Update logged-in user's profile

3. contact (child) → user (parent)

✅ Essential Contact API Endpoints

Method	Route	Auth Required	Description
`POST`	`/api/contacts`	✅ Yes	Create a new contact
`GET`	`/api/contacts`	✅ Yes	Get all contacts for the logged-in user
`GET`	`/api/contacts/:id`	✅ Yes	Get a single contact by ID
`PUT`	`/api/contacts/:id`	✅ Yes	Update a contact by ID
`DELETE`	`/api/contacts/:id`	✅ Yes	Delete a contact by ID

🔍 Optional Enhancements (Can be Filtered Client-side or Added as Routes)

Feature	Option	Description
Search/filter	`/api/contacts?search=rohit`	Filter by name, email, tags, etc.
Filter by tags	`/api/contacts?tag=family`	Return contacts with specific tag
Favorites only	`/api/contacts?favorite=true`	Show only favorite contacts
Pagination	`/api/contacts?page=1&limit=10`	Paginate large contact lists

✅ Contact Model – Data Fields (Tabular Format)

Field Name	Type	Required	Description
`user`	ObjectId	✅ Yes	Reference to the owner (`User` _id)
`name`	String	✅ Yes	Full name of the contact
`email`	String	❌ No	Contact's email address
`phone`	String	❌ No	Contact's phone number
`address`	String	❌ No	Contact's physical or mailing address
`birthday`	Date	❌ No	Contact's date of birth
`notes`	String	❌ No	Any additional notes about the contact
`favorite`	Boolean	❌ No	Marks contact as favorite (default: `false`)

`tags`	Array[String]	❌ No	Labels/tags (e.g., "work", "family", "client")
`createdAt`	Date	Auto	Timestamp of contact creation (`timestamps: true`)
`updatedAt`	Date	Auto	Timestamp of last update

✅ Project Tech Stack Overview (Tabular Format)

Layer	Technology	Purpose / Responsibility
Frontend	**Next.js (React)**	User Interface, routing, API consumption, form handling
Backend	**Node.js**	Server runtime environment for executing JS
	Express.js	REST API routing, middleware, business logic
Database	**MongoDB**	NoSQL database to store users, profiles, and contacts
ORM	**Mongoose**	MongoDB object modeling, schema definitions
Auth	**JWT (jsonwebtoken)**	Stateless authentication using access tokens
	bcryptjs	Password hashing for secure storage
Middleware	Custom Express Middleware	Protect routes, extract user from JWT
Dev Tools	Nodemon, TypeScript (optional)	Dev server reload & static typing

✅ MERN Contact Manager App – Deployment Overview

Layer	Platform	Purpose / Description
Frontend	**Vercel**	Deploy your **Next.js** app (auto CI/CD with GitHub)
Backend	**Render**	Deploy your **Node.js + Express + MongoDB** API server
Database	**MongoDB Atlas**	Cloud-hosted MongoDB database (fully managed)

Auth	**JWT**	Stateless authentication across frontend & backend

🚀 Future Features & Enhancements (Categorized)

🔐 Advanced Authentication & Security

Feature	Description
Refresh Token System	Implement access + refresh token flow for better security and session persistence
Forgot/Reset Password	Email-based password reset using tokens (NodeMailer + OTP/token validation)
Google OAuth Login	Allow users to sign in with Google (Passport.js or Firebase Auth)
Rate Limiting	Prevent brute-force attacks using `express-rate-limit`
Account Verification	Send verification email upon registration
Device Detection & Session Management	Track user login sessions per device/browser

👤 Profile & User Experience

Feature	Description
Profile Picture Upload	Upload via **Multer** + **Cloudinary**
Dark/Light Theme Toggle	Store preference in DB or localStorage
Multi-language Support (i18n)	Add support for internationalization/localization
Timezone Handling	Show created/updated dates based on user's timezone

📇 Contact Features

Feature	Description
Contact Import/Export	Export contacts as `.csv` or `.json` and allow CSV upload/import
Bulk Actions	Delete or favorite multiple contacts at once
Activity Log	Track CRUD actions per contact with timestamps
Contact Reminders	Set reminders to follow-up or reach out to specific contacts
Birthday Tracking	Add `birthday` field and notify user when it's near