create the api for the below tables that perform the CRUD operation in node.js CREATE TABLE users ( user\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(), username VARCHAR(150) UNIQUE NOT NULL, email VARCHAR(255) UNIQUE NOT NULL, hashed\_password TEXT NOT NULL, salt TEXT NOT NULL, is\_active BOOLEAN DEFAULT TRUE, is\_locked BOOLEAN DEFAULT FALSE, created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ); CREATE TABLE user\_profiles ( user\_id UUID PRIMARY KEY REFERENCES users(user\_id), full\_name VARCHAR(255), contact\_number VARCHAR(20), address TEXT, date\_of\_birth DATE ); CREATE TABLE auth\_tokens ( token\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(), user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE, access\_token TEXT NOT NULL, refresh\_token TEXT, user\_agent TEXT, ip\_address INET, expires\_at TIMESTAMP, revoked BOOLEAN DEFAULT FALSE, created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ); CREATE TABLE secrets ( secret\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(), user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE, secret\_type VARCHAR(50), -- e.g., "API\_KEY", "SSH", etc. secret\_hash TEXT NOT NULL, label VARCHAR(255), created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, expires\_at TIMESTAMP, is\_revoked BOOLEAN DEFAULT FALSE ); CREATE TABLE password\_resets ( reset\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(), user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE, reset\_token TEXT NOT NULL, is\_used BOOLEAN DEFAULT FALSE, expires\_at TIMESTAMP NOT NULL, created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ); CREATE TABLE user\_login\_history ( login\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(), user\_id UUID REFERENCES users(user\_id), success BOOLEAN NOT NULL, login\_method VARCHAR(50), -- e.g., "password", "MFA", "token" ip\_address INET, user\_agent TEXT, login\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, failure\_reason TEXT ); CREATE TABLE token\_audit\_log ( audit\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(), token\_id UUID REFERENCES auth\_tokens(token\_id) ON DELETE CASCADE, action\_type VARCHAR(50), -- e.g., "LOGIN", "LOGOUT", "REFRESH" ip\_address INET, user\_agent TEXT, action\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ); CREATE TABLE mfa\_tokens ( mfa\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(), user\_id UUID REFERENCES users(user\_id), mfa\_type VARCHAR(20), -- "SMS", "TOTP", "EMAIL" secret\_key TEXT, -- Encrypted TOTP shared key is\_enabled BOOLEAN DEFAULT FALSE, verified\_at TIMESTAMP );

**ChatGPT said:**

Updated saved memory

Here is a complete Node.js API using **Express**, **PostgreSQL**, and **pg** for performing CRUD operations on the tables you provided:

**✅ Setup Instructions**

1. **Install Dependencies**

bash

CopyEdit

npm init -y

npm install express pg dotenv uuid

1. **File Structure**

bash

CopyEdit

/project-root

├── db.js

├── server.js

├── routes/

│ ├── users.js

│ └── (other routes...)

└── controllers/

├── usersController.js

└── (other controllers...)

**📦 1. db.js – Database Connection**

js

CopyEdit

// db.js

const { Pool } = require('pg');

require('dotenv').config();

const pool = new Pool({

connectionString: process.env.DATABASE\_URL,

});

module.exports = pool;

**🧠 2. server.js – Entry Point**

js

CopyEdit

// server.js

const express = require('express');

const app = express();

const port = process.env.PORT || 3000;

const usersRoutes = require('./routes/users');

// import other routes similarly...

app.use(express.json());

app.use('/api/users', usersRoutes);

// app.use('/api/user\_profiles', userProfilesRoutes); // and so on

app.listen(port, () => {

console.log(`Server running on port ${port}`);

});

**👥 3. routes/users.js – User Routes**

js

CopyEdit

// routes/users.js

const express = require('express');

const router = express.Router();

const usersController = require('../controllers/usersController');

router.get('/', usersController.getAllUsers);

router.get('/:id', usersController.getUserById);

router.post('/', usersController.createUser);

router.put('/:id', usersController.updateUser);

router.delete('/:id', usersController.deleteUser);

module.exports = router;

**🔧 4. controllers/usersController.js**

js

CopyEdit

// controllers/usersController.js

const pool = require('../db');

const { v4: uuidv4 } = require('uuid');

exports.getAllUsers = async (req, res) => {

try {

const result = await pool.query('SELECT \* FROM users ORDER BY created\_at DESC');

res.json(result.rows);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.getUserById = async (req, res) => {

try {

const result = await pool.query('SELECT \* FROM users WHERE user\_id = $1', [req.params.id]);

res.json(result.rows[0]);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.createUser = async (req, res) => {

const { username, email, hashed\_password, salt } = req.body;

try {

const result = await pool.query(

`INSERT INTO users (user\_id, username, email, hashed\_password, salt)

VALUES ($1, $2, $3, $4, $5) RETURNING \*`,

[uuidv4(), username, email, hashed\_password, salt]

);

res.status(201).json(result.rows[0]);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.updateUser = async (req, res) => {

const { username, email, is\_active, is\_locked } = req.body;

try {

const result = await pool.query(

`UPDATE users SET username=$1, email=$2, is\_active=$3, is\_locked=$4, updated\_at=NOW()

WHERE user\_id=$5 RETURNING \*`,

[username, email, is\_active, is\_locked, req.params.id]

);

res.json(result.rows[0]);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.deleteUser = async (req, res) => {

try {

await pool.query('DELETE FROM users WHERE user\_id = $1', [req.params.id]);

res.json({ message: 'User deleted' });

} catch (err) {

res.status(500).json({ error: err.message });

}

};

**🧩 You can repeat the same structure for these tables:**

* user\_profiles
* auth\_tokens
* secrets
* password\_resets
* user\_login\_history
* token\_audit\_log
* mfa\_tokens

Just copy the pattern and adapt table/field names.

**✅ .env file**

bash

CopyEdit

DATABASE\_URL=postgresql://username:password@localhost:5432/yourdb