-- Users Table

CREATE TABLE users (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password TEXT NOT NULL,

role VARCHAR(50) CHECK (role IN ('user', 'admin', 'super\_admin')),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Partnership Requests Table

CREATE TABLE partnership\_requests (

id SERIAL PRIMARY KEY,

user\_id INT REFERENCES users(id) ON DELETE CASCADE,

company\_name VARCHAR(255) NOT NULL,

company\_type VARCHAR(100),

status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('pending', 'approved', 'rejected')),

partnership\_type VARCHAR(100),

area\_of\_cooperation TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Admin Actions Table

CREATE TABLE admin\_actions (

id SERIAL PRIMARY KEY,

admin\_id INT REFERENCES users(id) ON DELETE SET NULL,

request\_id INT REFERENCES partnership\_requests(id) ON DELETE CASCADE,

action VARCHAR(50) CHECK (action IN ('approved', 'rejected')),

reason TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

this ain't enough I think this is not correct Listen carefully I am plannin to make a partnership management system where a user applies to be a parner in the page with insa and there are two type of users. external and internal. external one's just sign up with some username password email and so on and asks to be a partner and the data is sent to an admin specifically to the partnership division, and this approves or disapproves. if approves it will sents back to the user who is requesting to be a partner and announce the director it or accept at the final signature is made by the super admin which is the director and it is the final state where the status of the request changes and the user become's a partner.

the second type the internal would have to be in the company since there are a lot of included beareu and devision's which are the one's that asks an external company to be a partner with insa and they provide the request holding the company type and objective of connection and some more to the department division and the department division and the rest follows the previous step where the department devision revise and asks low department and the low department feedback the approval or disapproval and the department division again announces the director for approval and disaaproval If approved the form will be sent to the external comapny with some form's feild and so and make the partnership live connect them with that division or internal User who wanted it in the first place.

the director watchs everything and these requests have a lot of relations considering them make a brand new shcema

# 

# 🚀 PERN Stack Partnership Management System - Full Development Roadmap

## 1️⃣ Project Overview

This document provides a step-by-step guide for developing a Partnership Management System using the PERN stack (PostgreSQL, Express.js, React, Node.js).

### Core Features:

- User Authentication (Sign up, login, JWT-based auth)  
- Partnership Requests (Users submit partnership requests)  
- Admin Approval Process (Admins approve/reject requests)  
- Role-Based Access Control (Admins, users, partners)  
- Partnership Management (Status tracking, company details)  
- Notifications & Email (Users notified about request status)  
- Dashboard & UI (Admin and user dashboards)

## 2️⃣ Tech Stack

- Frontend: React, Redux Toolkit, Tailwind CSS / DaisyUI  
- Backend: Node.js, Express.js  
- Database: PostgreSQL (with Sequelize ORM)  
- Auth: JWT, bcrypt for password hashing  
- Deployment: Docker, Vercel (Frontend), Render / Railway (Backend)

## 3️⃣ PostgreSQL Database Schema

-- Users Table  
CREATE TABLE users (  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(100) NOT NULL,  
 email VARCHAR(255) UNIQUE NOT NULL,  
 password TEXT NOT NULL,  
 role VARCHAR(50) CHECK (role IN ('user', 'admin', 'super\_admin')),  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- Partnership Requests Table  
CREATE TABLE partnership\_requests (  
 id SERIAL PRIMARY KEY,  
 user\_id INT REFERENCES users(id) ON DELETE CASCADE,  
 company\_name VARCHAR(255) NOT NULL,  
 company\_type VARCHAR(100),  
 status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('pending', 'approved', 'rejected')),  
 partnership\_type VARCHAR(100),  
 area\_of\_cooperation TEXT,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- Admin Actions Table  
CREATE TABLE admin\_actions (  
 id SERIAL PRIMARY KEY,  
 admin\_id INT REFERENCES users(id) ON DELETE SET NULL,  
 request\_id INT REFERENCES partnership\_requests(id) ON DELETE CASCADE,  
 action VARCHAR(50) CHECK (action IN ('approved', 'rejected')),  
 reason TEXT,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

## 4️⃣ Backend Setup (Express.js + Node.js)

To set up the backend, follow these steps:

* 1. Initialize the project: `mkdir partnership-management && cd partnership-management && npm init -y`
* 2. Install dependencies: `npm install express pg pg-hstore sequelize cors dotenv bcryptjs jsonwebtoken`
* 3. Create a `server.js` file and set up an Express server.

import express from "express";  
import dotenv from "dotenv";  
import cors from "cors";  
  
dotenv.config();  
  
const app = express();  
app.use(cors());  
app.use(express.json());  
  
app.get("/", (req, res) => {  
 res.send("Partnership Management API Running...");  
});  
  
const PORT = process.env.PORT || 5000;  
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

## 5️⃣ Frontend Setup (React + Redux)

Follow these steps to set up the frontend:

* 1. Create a React app: `npx create-react-app client && cd client`
* 2. Install required packages: `npm install redux react-redux axios react-router-dom tailwindcss daisyui`
* 3. Set up Redux for state management.

import { configureStore } from "@reduxjs/toolkit";  
import userReducer from "./features/userSlice";  
  
export const store = configureStore({  
 reducer: {  
 user: userReducer,  
 },  
});

## 6️⃣ Additional Notes

- Secure your API endpoints using authentication middleware.  
- Use Docker to manage the PostgreSQL database.  
- Consider adding email notifications using Nodemailer.  
- Deploy your backend on Railway or Render for a stable environment.  
- Use Vercel for frontend deployment.  
- Write unit tests to ensure code reliability.