Here’s a basic API backend setup for PostgreSQL using **Node.js (Express) with Prisma ORM**. It will handle CRUD operations for your Staff, Settings, and Security data.

**1️⃣ Install Dependencies**

Run the following in your backend directory:

sh

CopyEdit

npm init -y

npm install express prisma @prisma/client pg dotenv cors

**2️⃣ Set Up Prisma for PostgreSQL**

Run:

sh

CopyEdit

npx prisma init

This will create a .env file and a prisma/schema.prisma file.

Edit .env to include your PostgreSQL database connection:

ini

CopyEdit

DATABASE\_URL="postgresql://user:password@localhost:5432/hotel\_db"

Edit prisma/schema.prisma:

prisma

CopyEdit

generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "postgresql"

url = env("DATABASE\_URL")

}

model Staff {

id String @id @default(cuid())

name String

role String

status String

}

model Setting {

id String @id @default(cuid())

name String

value String

}

model Security {

id String @id @default(cuid())

name String

status String

}

Run:

sh

CopyEdit

npx prisma migrate dev --name init

**3️⃣ Create Express Server**

Create server.js:

javascript

CopyEdit

require('dotenv').config();

const express = require('express');

const cors = require('cors');

const { PrismaClient } = require('@prisma/client');

const app = express();

const prisma = new PrismaClient();

app.use(cors());

app.use(express.json());

/\*\* 📌 GET All Staff \*/

app.get('/api/staff', async (req, res) => {

const staff = await prisma.staff.findMany();

res.json(staff);

});

/\*\* 📌 ADD New Staff \*/

app.post('/api/staff', async (req, res) => {

const { name, role, status } = req.body;

const newStaff = await prisma.staff.create({ data: { name, role, status } });

res.json(newStaff);

});

/\*\* 📌 GET All Settings \*/

app.get('/api/settings', async (req, res) => {

const settings = await prisma.setting.findMany();

res.json(settings);

});

/\*\* 📌 UPDATE a Setting \*/

app.put('/api/settings/:id', async (req, res) => {

const { id } = req.params;

const { value } = req.body;

const updatedSetting = await prisma.setting.update({

where: { id },

data: { value },

});

res.json(updatedSetting);

});

/\*\* 📌 GET Security Settings \*/

app.get('/api/security', async (req, res) => {

const security = await prisma.security.findMany();

res.json(security);

});

/\*\* 📌 Start Server \*/

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

**4️⃣ Connect Frontend to Backend**

Update your frontend component:

javascript

CopyEdit

import { useEffect, useState } from 'react';

export default function StaffList() {

const [staff, setStaff] = useState([]);

useEffect(() => {

fetch('http://localhost:5000/api/staff')

.then((res) => res.json())

.then((data) => setStaff(data));

}, []);

return (

<div className="table-container">

<table className="table">

<thead>

<tr>

<th>Staff ID</th>

<th>Name</th>

<th>Role</th>

<th>Status</th>

</tr>

</thead>

<tbody>

{staff.map((s) => (

<tr key={s.id}>

<td>{s.id}</td>

<td>{s.name}</td>

<td>{s.role}</td>

<td>{s.status}</td>

</tr>

))}

</tbody>

</table>

</div>

);

}

**✅ Now You Have**

* A **PostgreSQL** database managed with **Prisma**.
* A **REST API** built with **Express** to fetch/update staff, settings, and security.
* A **React/Next.js** frontend displaying the data.

Let me know if you need enhancements! 🚀