After integrating the **PostgreSQL schema** with foreign key relationships, a few **updates** might be needed in your **components** and **app** folders to ensure proper **data fetching, API integration, and UI updates**.

**1️⃣ Backend API Integration (If Not Done Yet)**

If your components currently use **static data**, you need to replace it with **API calls** to your backend.

**Example Update for StaffList.tsx**

Replace the static array in StaffList.tsx with a **fetch request**:

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))

.catch((err) => console.error("Error fetching staff:", err));

}, []);

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>

);

}

✅ **Similar changes should be made for all list components (Bookings, Guests, Billing, Services, etc.).**

**2️⃣ Handling Data in Filters**

If your **filter components** currently use hardcoded options, update them to dynamically fetch values from the backend.

**Example Update for StaffFilters.tsx**

Replace hardcoded roles:

javascript

CopyEdit

import { useEffect, useState } from 'react';

export default function StaffFilters() {

const [roles, setRoles] = useState([]);

useEffect(() => {

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

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

.then((data) => setRoles(data))

.catch((err) => console.error("Error fetching roles:", err));

}, []);

return (

<div className="card">

<h2 className="text-lg font-medium mb-4">Filter Staff</h2>

<div className="grid grid-cols-1 md:grid-cols-3 gap-4">

<div>

<label className="form-label">Role</label>

<select className="form-input">

<option value="all">All Roles</option>

{roles.map((role) => (

<option key={role} value={role}>{role}</option>

))}

</select>

</div>

</div>

</div>

);

}

✅ **Similar changes should be applied for filters in Inventory, Services, Reports, etc.**

**3️⃣ Update API Routes in the App Folder**

If your Next.js **API routes (/app/api/...)** were based on **mock data**, update them to connect to **PostgreSQL**.

**Example Update for /app/api/staff/route.js**

javascript

CopyEdit

import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

export async function GET(req, res) {

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

return Response.json(staff);

}

✅ **Ensure all API routes (/api/guests, /api/rooms, /api/bookings) connect to the PostgreSQL database.**

**4️⃣ Error Handling & Loading States**

If API calls are **slow**, consider adding:

* **Loading indicators (useState for loading state)**
* **Error messages (catch() handling)**

**Summary of Needed Changes**

✅ **Replace static data** with API calls in:

* List components (StaffList.tsx, BookingsList.tsx, etc.)
* Filter components (StaffFilters.tsx, BookingFilters.tsx, etc.)

✅ **Update API routes** in app/api/ to fetch from **PostgreSQL**.

✅ **Add error handling & loading states** for better UX.

**Final Check**

If the API endpoints return correct data and UI updates dynamically, **no further changes** are needed!

Would you like me to generate the missing API route files for Next.js? 🚀