ERP Project Setup and Instructions

This document provides instructions on how to set up and run the ERP project, and explains some key concepts of the application.

1. Project Overview

The project is a full-stack ERP (Enterprise Resource Planning) application with a React frontend and a NestJS backend. It uses PostgreSQL as the database and Prisma as the ORM.

2. Database Schema Explanation

Why `invoices` and `invoice_items` are separate tables

The database uses separate tables for `invoices` and `invoice_items` to follow the principles of database normalization. This is a common practice in database design for several reasons:

- **One-to-Many Relationship:** An invoice can have multiple line items. For example, a single invoice can include a laptop, a mouse, and a keyboard. Instead of trying to store all of this information in a single row in the `invoices` table, we create a separate `invoice_items` table. Each row in `invoice_items` represents a single line item and is linked back to the main invoice through a foreign key (`invoiceId`).
- **Data Integrity:** This separation ensures that the data is consistent and avoids redundancy. If we were to store invoice items in the `invoices` table, we would have to either use a less structured format (like a JSON blob) or have many columns for items (item1, item2, etc.), which is inflexible and inefficient.
- **Scalability and Performance:** A normalized schema is generally more scalable and can be queried more efficiently. For example, if you want to get the total

amount of an invoice, you can easily sum the `total` column of the `invoice_items` table for a given `invoiceId`.

This same principle applies to other similar relationships in the database, such as `orders` and `order_items`, and `quotes` and `quote_items`.

Management (HR) and Finance Tables

- **Management (HR):** The `User` table in the database serves as the central table for all employees, including those in management/HR roles. You can assign different roles to users (e.g., `ADMIN`, `MANAGER`, `EMPLOYEE`) to control their access and permissions. There is no separate `Employee` table because all employees are considered users of the system.
- **Finance:** The `Payment` table is used to track all financial transactions. Each payment is linked to an invoice, so you can see which payments have been made for which invoices. There is no separate `Transaction` table because the `Payment` table serves this purpose.

3. User Roles and Permissions

The application has a role-based access control (RBAC) system to manage what different users can see and do. The main roles are:

- **ADMIN:** The administrator has full access to all features and data in the application.
- **FINANCE:** The finance user has access to financial data, such as invoices, payments, and quotes. They can create and manage invoices and payments.
- **MANAGER:** The manager has access to most features, including managing users, customers, products, orders, and sales.
- **SALES:** The sales user has access to sales-related features, such as creating and managing orders and quotes.
- **EMPLOYEE:** The employee has limited access to the system.

Here is a summary of the permissions for the Admin and Finance roles:

Feature	Admin Access	Finance Access
User Management	✅	❌
Customer Management	✅	✅
Product Management	✅	✅
Company Management	✅	❌
Order Management	✅	✅
Quote Management	✅	✅
Invoice Management	✅	✅
Payment Management	✅	✅
Sales Management	✅	✅

4. How to Run the Project

To run the project, you need to have Node.js and PostgreSQL installed on your system.

Backend Setup

1. **Navigate to the `backend` directory:**

   `bash cd backend`

2. **Install dependencies:**

   `bash npm install`

3. **Set up the database:**

   - Make sure your PostgreSQL server is running.
   - Create a new PostgreSQL database named `ASH_ERP_Project`.
   - Create a PostgreSQL user with the username `postgres` and password `usman123`.

    - The backend is pre-configured to connect to this database. If you need to change the database credentials, you can do so in the `.env` file in the `backend` directory.

4. **Run database migrations and seed the database:**

   `bash npx prisma migrate dev --name init npx prisma db seed`

   This will create the necessary tables in your database and populate them with the default admin and finance users.

5. **Start the backend server:**

   `bash npm run start:dev`

   The backend server will be running at `http://localhost:3000`.

Frontend Setup

1. **Navigate to the `frontend` directory:**

   `bash cd frontend`

2. **Install dependencies:**

   `bash npm install`

3. **Start the frontend development server:**

   `bash npm run dev`

   The frontend will be running at `http://localhost:8080`.

5. Default Login Credentials

- **Admin:**
    - **Email:** `admin@ashtech.com`
    - **Password:** `admin123`
- **Finance:**
    - **Email:** `finance@ashtech.com`

    - **Password:** `finance123`