✅ Problem Statement: AltiusHub Frontend + Backend Invoice App

You are required to develop a **Full Stack Invoice Management Application** using Material Design (for frontend) and RESTful APIs (for backend). The app should support creating, viewing, updating, and deleting invoices with line items and bill sundries.

🔹 FRONTEND REQUIREMENTS

1. UI Architecture

- Use **Material Design** components.
- The app must include a **fixed toolbar** with a **side navigation drawer**.
- Side nav should contain one list item: "**Invoices**".
- On clicking "Invoices", navigate to a **List View** of invoices.

2. InvoiceListComponent

- Display list of invoices in a **Material Table**.
- Include **paginator** for navigation.
- Above the table, include an **"Add" button** → navigates to `InvoiceDetailComponent()` (Create Mode).
- Clicking any row → navigates to `InvoiceDetailComponent/:id` (Update Mode).

3. InvoiceDetailComponent

- Create a dynamic form using the `Invoice` interface structure.
- **Form Fields** (Based on Interface from Page 2):

ts Copy Edit

```ts
interface Invoice { Id: string; Date: string; InvoiceNumber: number; CustomerName: string; BillingAddress: string; ShippingAddress: string; GSTIN: string; Items: InvoiceItem[]; BillSundrys: InvoiceBillSundry[]; TotalAmount: number; } interface InvoiceItem { itemName: string; quantity: number; price: number; amount: number; } interface InvoiceBillSundry { billSundryName: string; amount: string; }
```

4. Form Validations

General:

- All fields are **mandatory**.

Items:

- Amount = quantity × price.
- Price must be > 0.

Bill Sundry:

- Can have **negative or positive** values.
- TotalAmount = Sum(items.amount) + Sum(billSundry.amount).

Form Rules:

- At least one item must be present.
- Date must not allow backdated entries.
- Invoice number must be **auto-incremented** based on previous invoice.

5. **Form Modes**

- **Create Mode**:
  - Show **Cancel** and **Save** buttons.
- **Update Mode**:
  - Show **Cancel**, **Delete**, and **Save** buttons.

🔹 BACKEND REQUIREMENTS

1. **Database Schema**

Invoice Header Table

- id: UUID
- date: string (UTC)
- invoiceNumber: number (auto-incremented)
- customerName: string
- billingAddress: string
- shippingAddress: string
- GSTIN: string
- totalAmount: decimal

Invoice Items Table

- id: UUID
- invoiceId: FK (UUID)
- itemName: string
- quantity: decimal

- price: decimal
- amount: decimal

Invoice BillSundry Table

- id: UUID
- invoiceId: FK (UUID)
- billSundryName: string
- amount: decimal

---

2. **REST API Endpoints**

Implement full CRUD (Create, Read, Update, Delete) for Invoices:

- **POST /invoices** → Create an invoice
- **GET /invoices** → Get all invoices
- **GET /invoices/:id** → Get invoice by ID
- **PUT /invoices/:id** → Update invoice
- **DELETE /invoices/:id** → Delete invoice

3. **Backend Validations**

- Validate amounts as:
  - `item.amount = quantity × price` (must be > 0)
  - `totalAmount = sum(items.amounts) + sum(billSundry.amounts)`
- Allow bill sundry to be negative or positive.
- Auto-generate **unique invoiceNumber**.
- Raise **clear validation errors** for bad requests.