**Using React.js Typescript and possibly Django / Laravel, I would want to have:**

* Total Sales & Total Purchases
* Stock levels & Product performance
* Payments & Invoices
* Profit / Loss Analysis
* Income & Expenditure
* Top Salespersons
* Charts & Graphs
* Recent sales, paid and unpaid
* Recent purchases, paid and unpaid etc.
* Top clients
* Suppliers analysis

Note; All these can be accessed on daily basis, weekly basis, monthly basis, quarterly, half and yearly

That’s a solid plan! With **React.js (TypeScript) for the frontend** and **Django/Laravel for the backend**, your dashboard will be powerful and scalable. Here’s how we can structure it:

**Tech Stack & Architecture**

🔹 **Frontend** → React.js (TypeScript)  
🔹 **Backend** → Django (Python) or Laravel (PHP)  
🔹 **Database** → PostgreSQL / MySQL  
🔹 **API** → REST API (Django Rest Framework or Laravel API)  
🔹 **UI Library** → Tailwind CSS / Material UI / Ant Design  
🔹 **Charts** → Recharts / Chart.js  
🔹 **State Management** → Redux Toolkit or React Query

**Dashboard Features & Sections**

**1️⃣ Key Metrics Overview**

✅ **Total Sales & Total Purchases** – Show values with comparisons (Daily, Weekly, Monthly, etc.)  
✅ **Profit/Loss Analysis** – Revenue, Cost of Goods Sold (COGS), Expenses, Net Profit  
✅ **Income & Expenditure** – Categorized breakdown with filters

**2️⃣ Inventory & Performance**

✅ **Stock Levels** – Low stock alerts, available stock, and movement tracking  
✅ **Product Performance** – Top-selling products, slow-moving stock, profit margins

**3️⃣ Financials & Transactions**

✅ **Payments & Invoices** – Paid, unpaid, overdue invoices  
✅ **Recent Sales & Purchases** – Show latest transactions with statuses  
✅ **Top Clients** – Customers ranked by purchases and payment behavior  
✅ **Suppliers Analysis** – Order frequency, total amount spent

**4️⃣ Sales & Employee Performance**

✅ **Top Salespersons** – Based on revenue generated and number of transactions  
✅ **Commission Tracking** (if applicable)

**5️⃣ Visual Analytics**

✅ **Charts & Graphs** – Sales trends, expense breakdowns, stock movements  
✅ **Filters** – Daily, Weekly, Monthly, Quarterly, Half-yearly, Yearly

**How to Proceed**

🔹 **Step 1**: Set up the **React.js + TypeScript** frontend  
🔹 **Step 2**: Build the **Django/Laravel backend** with API endpoints  
🔹 **Step 3**: Design database tables for transactions, invoices, stock, sales, etc.  
🔹 **Step 4**: Integrate **charts & graphs** for visual insights  
🔹 **Step 5**: Implement **authentication & role-based access**

Here’s a **detailed kickstart guide** with the **folder structure, API design, and UI wireframe** for your **React.js (TypeScript) + Django/Laravel** dashboard.

# **📂 Folder Structure (React + Django/Laravel)**

This is a structured way to organize both frontend and backend projects.

### ****Frontend (React + TypeScript)****

/dashboard-frontend

│── /src

│ │── /components # Reusable UI components (cards, tables, charts)

│ │── /pages # Pages (Dashboard, Sales, Inventory, Reports, etc.)

│ │── /hooks # Custom React hooks

│ │── /context # Context API (if using global state)

│ │── /services # API calls

│ │── /utils # Helper functions, constants

│ │── /assets # Images, icons, styles

│ │── /styles # Tailwind CSS or SCSS files

│ │── App.tsx # Main App component

│ │── index.tsx # Entry point

│── package.json # Dependencies

│── tailwind.config.js # Tailwind CSS config (if used)

│── tsconfig.json # TypeScript config

### ****Backend (Django REST or Laravel API)****

For **Django REST Framework**:

/dashboard-backend

│── /dashboard\_api # Main API app

│ │── /models # Database models

│ │── /serializers # Data serializers

│ │── /views # API logic

│ │── /urls.py # API routes

│ │── /tests # Unit tests

│── manage.py # Django CLI tool

│── requirements.txt # Dependencies

│── settings.py # Configuration

For **Laravel API**:

/dashboard-backend

│── /app

│ │── /Models # Database models

│ │── /Http/Controllers # API logic

│ │── /Http/Requests # Form requests (validations)

│── /routes/api.php # API routes

│── composer.json # Dependencies

│── .env # Environment variables

# **📌 API Design**

The **backend API** will expose endpoints for the frontend to consume.

### ****1️⃣ Authentication****

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/auth/register | POST | User registration |
| /api/auth/login | POST | User login |
| /api/auth/logout | POST | Logout user |
| /api/auth/profile | GET | Get user profile |

### ****2️⃣ Sales & Purchases****

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/sales | GET | Fetch all sales |
| /api/sales/:id | GET | Fetch a single sale |
| /api/sales | POST | Create a new sale |
| /api/sales/:id | PUT | Update a sale |
| /api/sales/:id | DELETE | Delete a sale |

| /api/purchases | GET | Fetch all purchases | | /api/purchases/:id | GET | Fetch a single purchase | | /api/purchases | POST | Create a new purchase | | /api/purchases/:id | PUT | Update a purchase | | /api/purchases/:id | DELETE | Delete a purchase |

### ****3️⃣ Stock & Product Performance****

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/inventory | GET | Fetch current stock levels |
| /api/inventory/:id | GET | Get details of a product |
| /api/inventory/performance | GET | Get top-performing products |

### ****4️⃣ Payments & Invoices****

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/invoices | GET | Fetch all invoices |
| /api/invoices/:id | GET | Fetch a single invoice |
| /api/invoices | POST | Create an invoice |
| /api/invoices/:id | PUT | Update an invoice |
| /api/invoices/:id | DELETE | Delete an invoice |

| /api/payments | GET | Fetch all payments | | /api/payments/:id | GET | Fetch a single payment | | /api/payments | POST | Create a payment |

### ****5️⃣ Reports & Analytics****

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/reports/sales | GET | Get sales report |
| /api/reports/expenses | GET | Get expense report |
| /api/reports/profit-loss | GET | Get profit & loss analysis |
| /api/reports/top-clients | GET | Get top clients |
| /api/reports/top-salespersons | GET | Get top salespersons |

# **🖥 UI Wireframe**

### ****📊 Dashboard Layout****

* **Top Section:** KPIs (Total Sales, Purchases, Profit, Inventory)
* **Middle Section:** Graphs & Charts (Sales trends, product performance)
* **Bottom Section:** Recent Sales, Purchases, Unpaid Invoices

**💡 Example Layout:**

---------------------------------------------------

| 🔵 Total Sales | 🔴 Total Purchases | 🟢 Profit/Loss |

---------------------------------------------------

| 📈 Sales Trend Chart | 📊 Inventory Chart |

---------------------------------------------------

| 🛒 Recent Sales | 📦 Recent Purchases |

---------------------------------------------------

| ✅ Paid Invoices | ⏳ Unpaid Invoices |

---------------------------------------------------

## ****🚀 Next Steps****

1️⃣ **Set up React & TypeScript** → npx create-react-app dashboard-frontend --template typescript  
2️⃣ **Install Django/Laravel** & Set up API endpoints  
3️⃣ **Connect Frontend & Backend via API Calls**  
4️⃣ **Integrate Charts (Recharts / Chart.js)**  
5️⃣ **Deploy on AWS / DigitalOcean (if required)**

**💬 Let me know if you need code snippets or a GitHub starter template!** 🚀🔥