how would you design a large scale angular app using with multiple teams members?

1. **Modular Architecture (Feature Modules)**

**Ng new <appName>**

**Structure:-**

/src

/app

/core

/shared

/layouts

/services

/features

/users

/orders

/dashboard

1. **Nx Workspace for Large Angular App**

**Step 1: Create an Nx Workspace**

**npx create-nx-workspace@latest my-enterprise-app**

**Generate Main App**

**nx generate @nrwl/angular:application main-app**

**Generate Feature Library**

**nx generate @nrwl/angular:library feature-dashboard --directory=features --style=scss --lazy**

**Generate Shared Component Library**

**nx generate @nrwl/angular:library ui-button --directory=shared/ui**

**Step 3: Use Libraries in Your App**

**import { DashboardModule } from '@my-enterprise-app/features/dashboard';**

**@NgModule({**

**imports: [**

**DashboardModule,**

**// other modules**

**],**

**})**

**export class AppModule {}**

**Step 4: Run, Test, and Lint**

**nx serve main-app # Run app**

**nx test shared-ui-button # Unit test for a specific library**

**nx lint core-data-access # Lint a specific library**

**nx affected:build # CI - only build what changed**

**Structures:-**

my-enterprise-app/

├── apps/

│ ├── main-app/ --> Angular main application

│ ├── admin-app/ --> Separate admin Angular app (optional)

│ └── shell/ --> Micro-frontend shell (optional)

├── libs/

│ ├── core/ --> Core services (e.g., Auth, API)

│ │ ├── data-access/

│ │ └── ui/

│ ├── shared/ --> Shared components, directives, pipes

│ │ ├── ui/

│ │ ├── utils/

│ │ └── models/

│ ├── features/ --> Feature modules

│ │ ├── dashboard/

│ │ ├── profile/

│ │ ├── settings/

│ │ └── ...

└── nx.json, angular.json, etc.

**✅ Benefits of Nx for Large Angular Apps**

* **Separation of concerns** via libraries
* **Code sharing** across multiple apps
* **Scalable CI/CD** using nx affected
* **Improved developer productivity** via generators
* **Automatic dependency graph** visualization

🖼️ Dependency Graph

**nx graph**

**🆚 ng new vs nx workspace**

| **Feature / Concept** | **ng new (Angular CLI)** | **nx workspace (Nrwl Nx)** |
| --- | --- | --- |
| **Purpose** | Create a single Angular app | Create a monorepo for one or many apps and libraries |
| **Default structure** | Flat structure: all code inside one app | Modular structure with apps/ and libs/ folders |
| **Scalability** | Suited for small to medium projects | Designed for large, enterprise-scale projects |
| **Monorepo support** | Not natively supported | Built-in and first-class monorepo support |
| **Library generation** | Limited support via ng generate library | Powerful and structured libraries with tags and boundaries |
| **Code sharing** | Manual via imports | Easy via libraries (shared/core/features, etc.) |
| **Dependency graph** | ❌ No built-in tooling | ✅ nx graph shows full dependency tree |
| **Affected commands** | ❌ Not available | ✅ nx affected:\* optimizes CI and rebuilds |
| **Built-in tools (lint/test/build)** | Limited to one app | Tooling per app/library with caching and optimization |
| **Generators and Plugins** | Basic Angular CLI schematics | Rich plugin ecosystem: Angular, React, Node, NestJS, etc. |
| **Module Federation (Micro-Frontend)** | Manual setup required | First-class support via plugins |
| **Caching and CI Optimization** | ❌ No smart caching | ✅ Remote caching, task scheduling, and affected builds |
| **Learning curve** | 🟢 Easier to start | 🔵 Steeper at first, more powerful long term |

🏗 Example

✅ ng new my-app

ng new my-app

my-app/

├── src/

│ └── app/

├── angular.json

├── package.json

✅ npx create-nx-workspace my-workspace

npx create-nx-workspace@latest my-workspace

my-workspace/

├── apps/

│ └── main-app/

├── libs/

│ ├── shared/

│ └── features/

├── nx.json

├── angular.json