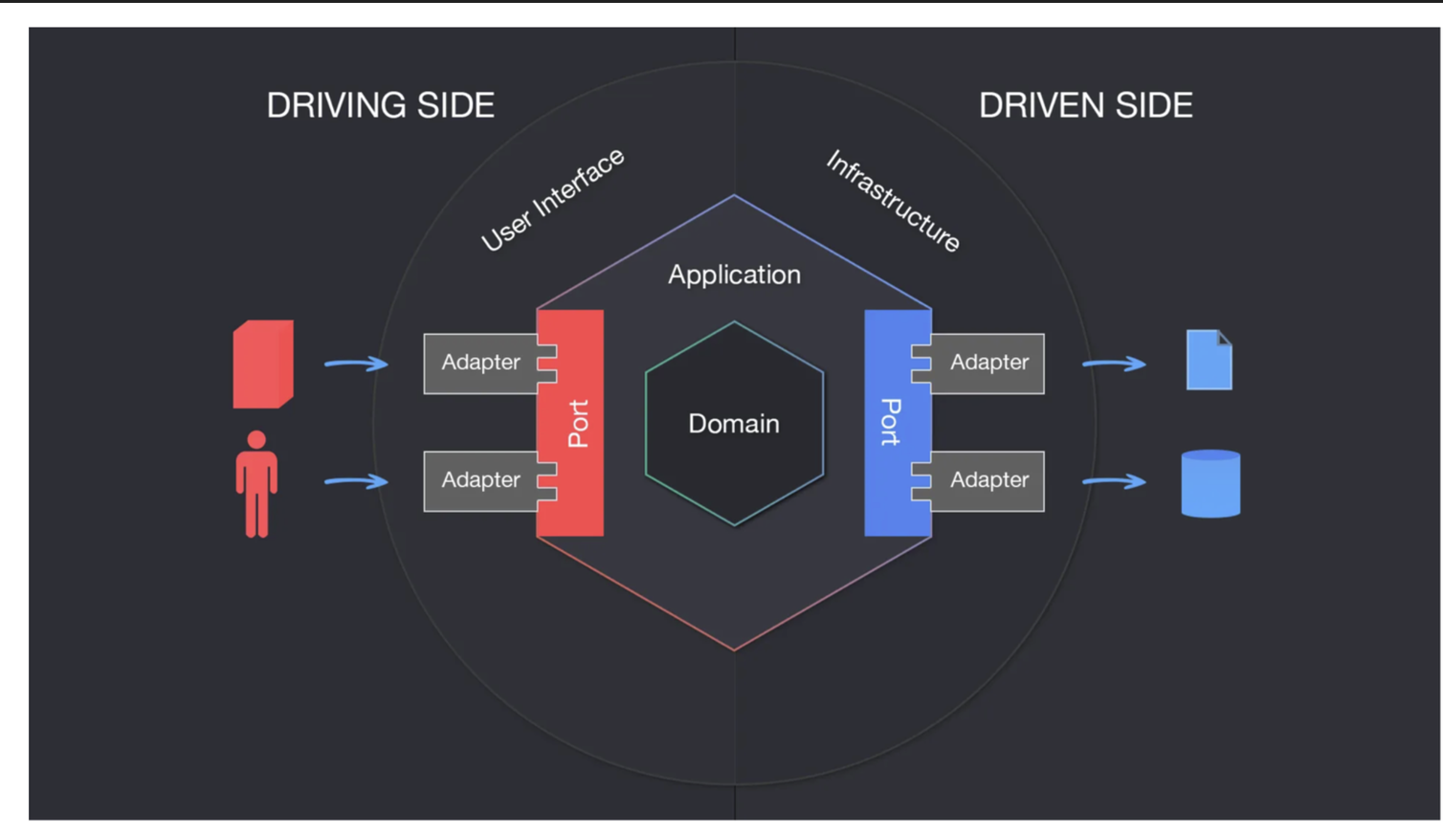
**Build a Full Stack E-commerce app with Spring boot 3, PostgreSQL, Angular 18 and Tailwind CSS (2024)**

<https://www.youtube.com/watch?v=4npG3sAMT5I>

<https://github.com/C0de-cake/ecommerce-app/tree/main/apps>

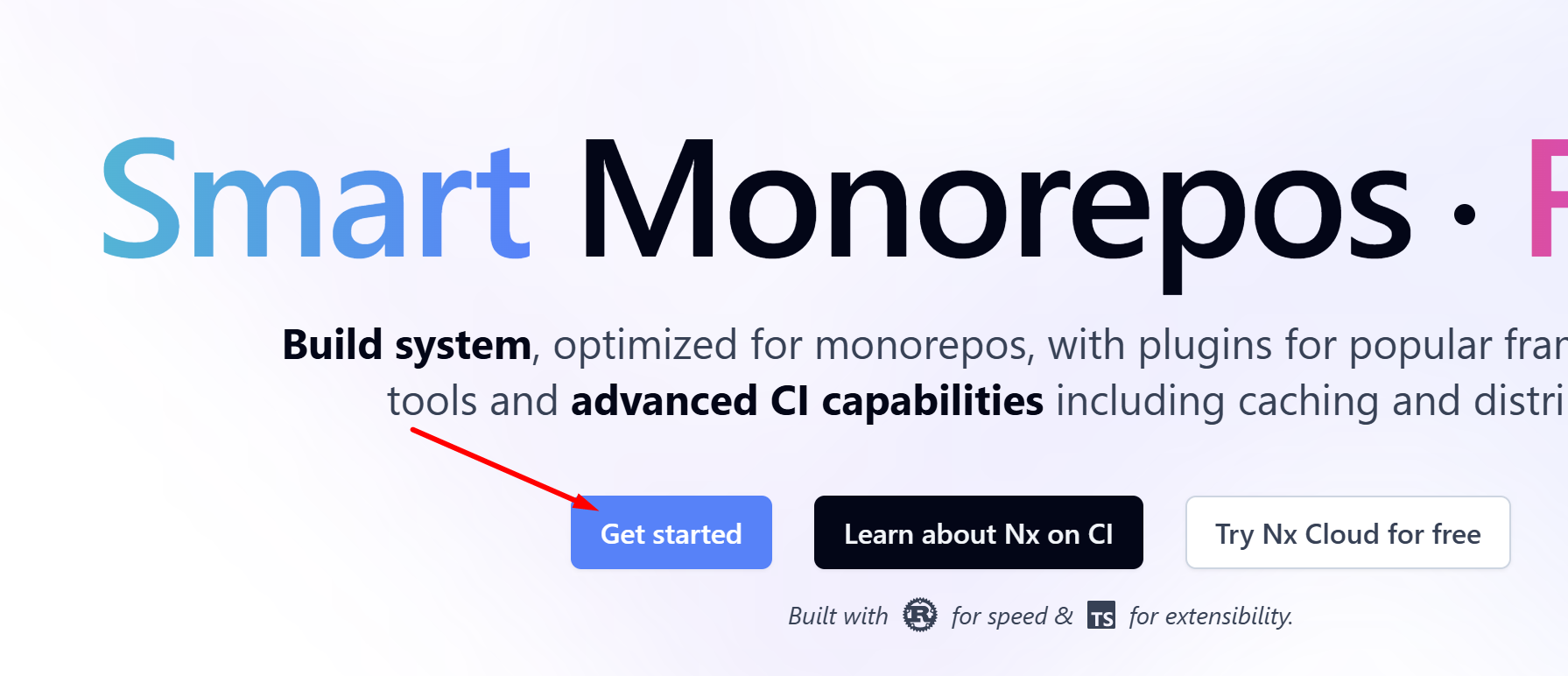


Build this project as mono repo us Nx!

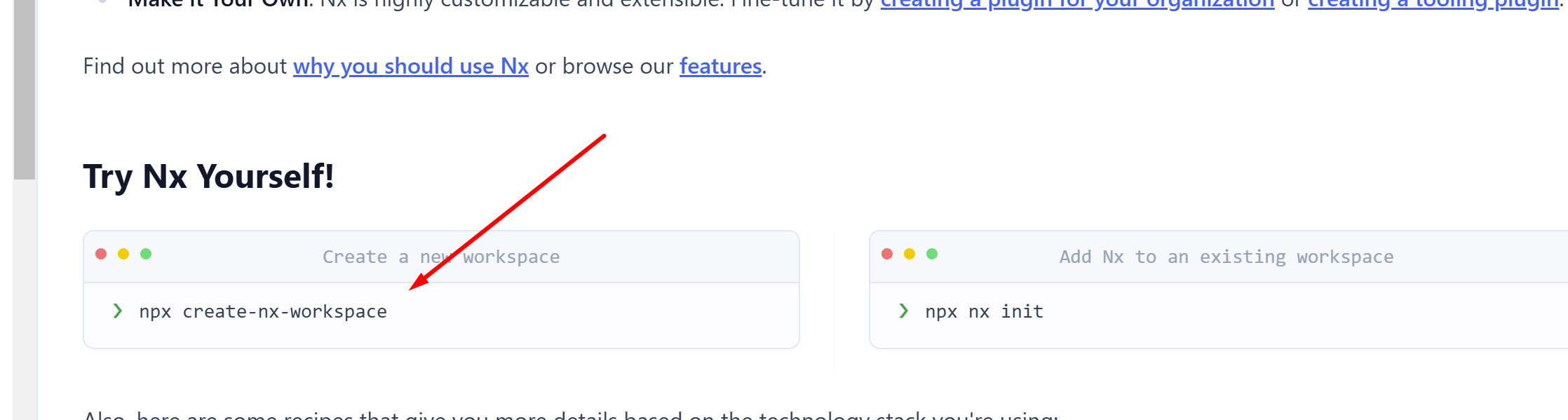
**Schema documentation** [https://github.com/C0de-cake/ecommerc...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbDF3QmJ3aUlvRGRsMndMeng1T3lKUVVDeHpJd3xBQ3Jtc0tudXQ4NDRidzNQOFB6SldfemxwZDBKRDl1UU9DSHI4bGhTanJ3VXhqU2djUnoya0EyMW9MV3V4X29uQW94cHdJQWdJSXdyUkdRWmNFWVJQX2hkX0JZVG5rbVB5V3JPMXhyMTlQNHNYb1BiQm9weVZ4VQ&q=https%3A%2F%2Fgithub.com%2FC0de-cake%2Fecommerce-app%2Fblob%2Fmain%2Fschema-ecommerce-app.png&v=4npG3sAMT5I" \t "_blank)

|  |  |
| --- | --- |
| **Generate monorepo project** https://nx.dev/ | **npx create-nx-workspace** |

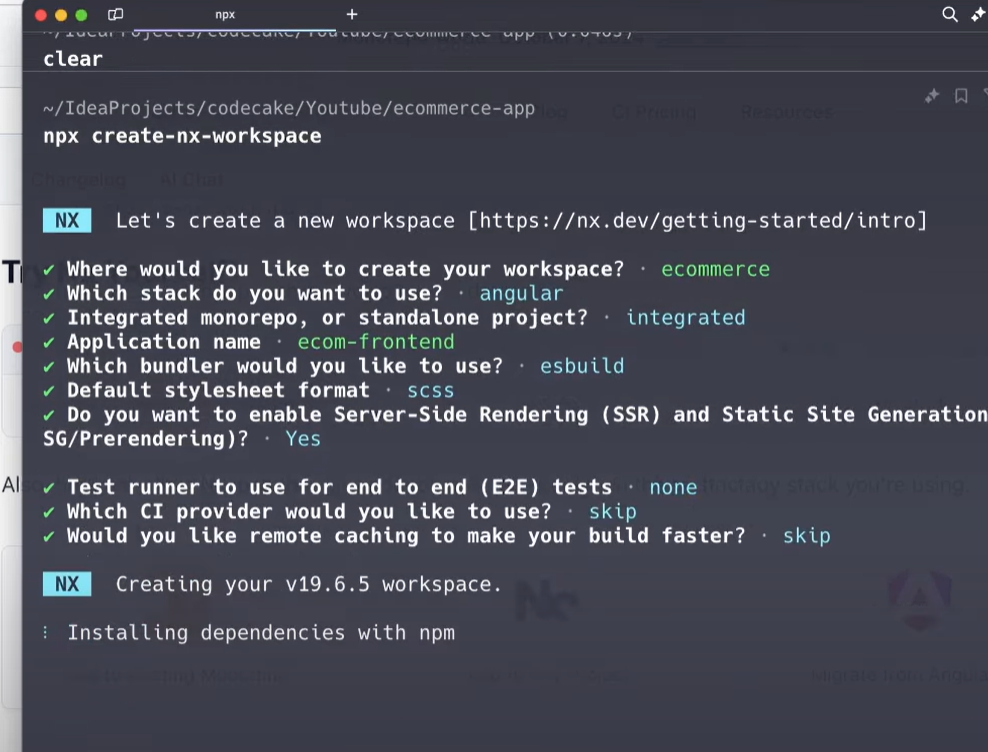
First generate front end and then backend project



2



3



|  |  |
| --- | --- |
| To run the dev server for your app, use: | **npx nx serve frontend** |
| Verify **nx** is installed: | **npx nx --version** |
| Install **Tailwind CSS** | **npx nx g @nx/angular:setup-tailwind frontend** |
| **DaisyUI**  (Tailwind plugin which extends Tailwind with pre-styled components) adds pre-built components and themes to Tailwind CSS, making it easier to design user interfaces. https://daisyui.com/ | **npm i -D daisyui@latest** |
| **Tailwind CSS Typography** The official Tailwind CSS Typography plugin provides a set of prose classes you can use to add beautiful typographic defaults to any vanilla HTML you don’t control, like HTML rendered from Markdown, or pulled from a CMS. **https://github.com/tailwindlabs/tailwindcss-typography** | **npm install -D @tailwindcss/typography** |

**Update tailwind.config.js**

Here's a concise explanation of the tailwind.config.js file, breaking down each row or section:

**Basic Configuration**

javascript

Copy code

const { createGlobPatternsForDependencies } = require('@nx/angular/tailwind');

const { join } = require('path');

* **Purpose:** These are required dependencies:
  + createGlobPatternsForDependencies: Generates patterns to include Tailwind styles from all dependencies.
  + join: Used to construct file paths.

**content Section**

javascript

Copy code

content: [

join(\_\_dirname, 'src/\*\*/!(\*.stories|\*.spec).{ts,html}'),

...createGlobPatternsForDependencies(\_\_dirname),

],

* **Purpose:** Specifies the files where Tailwind should look for class usage.
  + join(\_\_dirname, 'src/\*\*/!(\*.stories|\*.spec).{ts,html}'): Includes all .ts and .html files in the src directory but excludes \*.stories and \*.spec files.
  + ...createGlobPatternsForDependencies(\_\_dirname): Includes styles from dependent libraries (helpful in Nx workspaces).

**theme Section**

javascript

Copy code

theme: {

fontFamily: {

'sans': 'Inter var, ui-sans-serif, system-ui',

'serif': 'Inter var, ui-sans-serif, system-ui',

},

fontSize: {

sm: '0.875rem',

base: '1.35rem',

xl: '1.55rem',

'2xl': '1.563rem',

'3xl': '1.953rem',

'4xl': '2.441rem',

'5xl': '3.052rem',

},

extend: {},

},

* **Purpose:** Customizes the design tokens for your project.
  + fontFamily: Defines default fonts for sans and serif.
  + fontSize: Customizes font sizes for specific classes like sm, base, xl, etc.
  + extend: A placeholder to add custom styles later.

**daisyui Section**

daisyui: {

themes: [

{

'fantasy': {

primary: '#0000ff',

'primar-content': 'white',

secondary: '#F6F6F6',

neutral: '#E8E8E8',

},

},

],

base: true,

styled: true,

utils: true,

prefix: '',

logs: true,

themeRoot: ':root',

},

* **Purpose:** Configures **DaisyUI** (a Tailwind CSS plugin for themes and components).
  + themes: Defines a custom theme named fantasy with colors for primary, secondary, neutral, etc.
  + base, styled, utils, logs: Enable DaisyUI features like base styles, utility classes, and logging.
  + themeRoot: ':root': Adds CSS variables to the :root element for theme customizations.

**plugins Section**

javascript

Copy code

plugins: [

require('@tailwindcss/typography'),

require('daisyui'),

],

* **Purpose:** Adds plugins to extend Tailwind functionality.
  + @tailwindcss/typography: Adds pre-styled typography utilities.
  + daisyui: Enables DaisyUI components and theming.

**Simplified Overview**

* **Dependencies:** Import required modules for path handling and Nx compatibility.
* **Content:** Defines files where Tailwind should scan for class usage.
* **Theme:** Customizes fonts, font sizes, and other styles.
* **DaisyUI:** Configures a custom theme for easy styling using DaisyUI.
* **Plugins:** Adds plugins for extra functionality.

Let me know if you'd like further clarification!

<https://www.youtube.com/watch?v=4npG3sAMT5I>

<https://github.com/C0de-cake/ecommerce-app/tree/main/apps>

|  |  |
| --- | --- |
| Cleanup unnecessary data. Remove: | **nx-welcome.component.ts** |
| Install icons library FortAwesome form <https://github.com/FortAwesome/angular-fontawesome>. Add difference pack: solid, reguker, brands | **npm install @fortawesome/free-solid-svg-icons @fortawesome/free-brands-svg-icons @fortawesome/free-regular-svg-icons** |
| Integration between our icons an Anguler | **npm install @fortawesome/angular-fontawesome** |

|  |  |
| --- | --- |
| Where we list all difference icons that we use in project without neessertu to load all three libraries. If we need icon we just for here and add the sertan icin.  This is helpful for managing icons in a centralized way, especially if you have multiple icons to use in your application. Create: | **font-awesome-icons.ts** |
|  |  |

**1. The Angular Component**

This code is an **Angular component** (AppComponent) that sets up **Font Awesome** for use in your application. Here’s what each part does:

**Code Breakdown**

**import { Component, inject, OnInit } from '@angular/core';**

**import { RouterModule } from '@angular/router';**

**import { FaConfig, FaIconLibrary, FontAwesomeModule } from '@fortawesome/angular-fontawesome';**

**import { fontAwsomeIcons } from './shared/font-awesome-icons';**

* **What’s Being Imported?**
  + **Angular Core**: Component, inject, and OnInit are used to define and manage Angular components.
  + **RouterModule**: Allows navigation between different parts of your app (not directly related to Font Awesome but included here).
  + **FontAwesomeModule**: Enables the use of fa-icon components.
  + **FaConfig**: Configures Font Awesome settings (e.g., default prefix for icons).
  + **FaIconLibrary**: Registers icons so they can be used in your app.
  + **fontAwsomeIcons**: A custom list of icons imported from another file.

**Component Metadata**

**@Component({**

**imports: [ RouterModule, FontAwesomeModule ],** // Modules this component depends on

**selector: 'app-root',** // HTML tag for this component

**templateUrl: './app.component.html',** // HTML file for the component

**styleUrl: './app.component.scss',** // SCSS (styles) file for the component

**})**

* This defines how Angular recognizes and uses the component:
  + imports: Includes additional modules needed by this component.
  + selector: Defines the HTML tag (<app-root>) to embed this component in other files.
  + templateUrl: Specifies the HTML template file.
  + styleUrl: Specifies the SCSS (style) file.

**Setting Up Font Awesome**

typescript

Copy code

**export class AppComponent implements OnInit {**

**private faIconLibrary = inject(FaIconLibrary);** // Access Font Awesome icon library

**private faConfig = inject(FaConfig);** // Access Font Awesome configuration

**ngOnInit(): void {**

**this.initFontAwesome();** // Runs when the component is initialized

**}**

**private initFontAwesome(): void {**

**this.faConfig.defaultPrefix = 'far';** // Sets the default prefix for icons (e.g., 'far' for regular)

**this.faIconLibrary.addIcons(...fontAwsomeIcons);** // Adds your list of icons to the library

**}**

**}**

1. **Inject Font Awesome Services**:
   * FaIconLibrary: Manages the list of available icons.
   * FaConfig: Configures how Font Awesome behaves (e.g., sets a default icon style).
2. **ngOnInit Lifecycle Hook**:
   * Runs the initFontAwesome() function when the component is initialized.
3. **initFontAwesome()**:
   * Sets the default prefix for icons to 'far' (for regular icons).
   * Registers the list of icons (fontAwsomeIcons) in the library so they can be used in templates.

**2. The Font Awesome Icon List**

This is a **separate file** (font-awesome-icons.ts) that defines which icons will be used in the app.

**Code Breakdown**

**import { IconDefinition } from "@fortawesome/angular-fontawesome";**

**import { faUser } from "@fortawesome/free-solid-svg-icons";**

* **IconDefinition**:
  + A type that ensures the icons added are valid Font Awesome icons.
* **faUser**:
  + An icon (e.g., a user silhouette) imported from the Font Awesome **solid icons** library.

**export const fontAwsomeIcons: IconDefinition[] = [**

**faUser,**

**];**

* This creates and exports a list of icons (fontAwsomeIcons), which includes the faUser icon.
* You can add more icons to this list as needed.

**How It All Fits Together**

1. **fontAwsomeIcons**:
   * Defines the icons you want to use in your app (e.g., faUser).
2. **AppComponent**:
   * Imports this list of icons (fontAwsomeIcons).
   * Registers the icons with FaIconLibrary so they can be used in your app.
   * Configures Font Awesome to use a default prefix (far, fas, or fab).
3. **Using Icons in HTML**:
   * After everything is set up, you can use icons in your templates like this:

**<fa-icon [icon]="['far', 'user']"></fa-icon> <!-- Uses the "user" icon -->**

**In Simple Terms**

* The **first file** (AppComponent) sets up Font Awesome and registers the icons in your Angular app.
* The **second file** (font-awesome-icons.ts) defines which icons you want to use.
* Once everything is configured, you can easily display icons in your app's HTML using the <fa-icon> component.

In ***project.json***

|  |  |
| --- | --- |
| Change prefix on  It will initiate components with such prefix… | **"prefix": "ecom"** |

In ***package.json***

|  |  |
| --- | --- |
| Change on another version  + npm i | **"@types/node": "20.2.0"** |

Create UI architecture for app

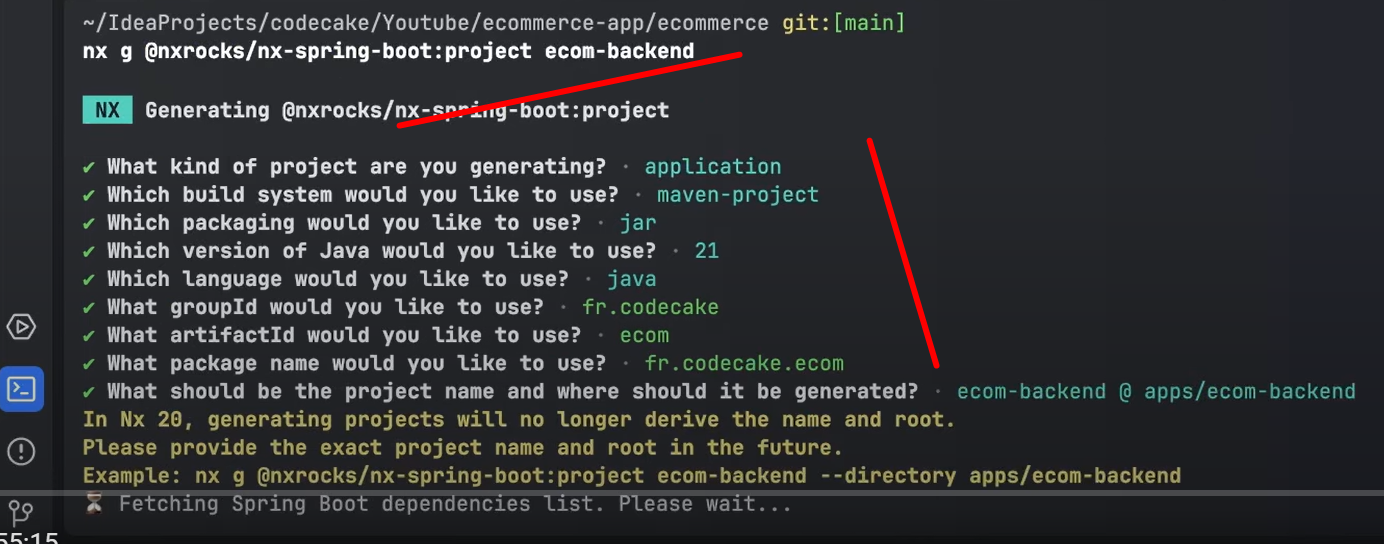
|  |  |
| --- | --- |
| is generating an Angular component using the Nx CLI. | **npx nx g @nx/angular:component apps/frontend/src/app/layout/navbar/navbar.component.ts** |
| Generate footer component | **npx nx g @nx/angular:component apps/frontend/src/app/layout/footer/footer.component.ts** |
| Avoid component in selector | **npx nx g @nx/angular:component apps/frontend/src/app/layout/navbar/navbar --selector=ecom-footer** |

In ***nx.json***

|  |  |
| --- | --- |
| We have scss in  That will create all component styles file with .scss | **"style": "scss"** |

**Generate back end**

|  |  |
| --- | --- |
| Integrate nx project with Spring Boot | [**https://www.npmjs.com/package/@nxrocks/nx-spring-boot**](https://www.npmjs.com/package/@nxrocks/nx-spring-boot) |
| install the plugin in order to generate Spring Boot applications later on. | **npm install @nxrocks/nx-spring-boot --save-dev** |
| Generating Projects (project generator) | **npx nx g @nxrocks/nx-spring-boot:project backend** |



What About **Installing Nx Globally?**

|  |  |
| --- | --- |
| install Nx globally with | npm install -g nx |

<https://www.youtube.com/watch?v=4npG3sAMT5I>

<https://github.com/C0de-cake/ecommerce-app/tree/main/apps>

**31:50**