1. **Course Project Intro**

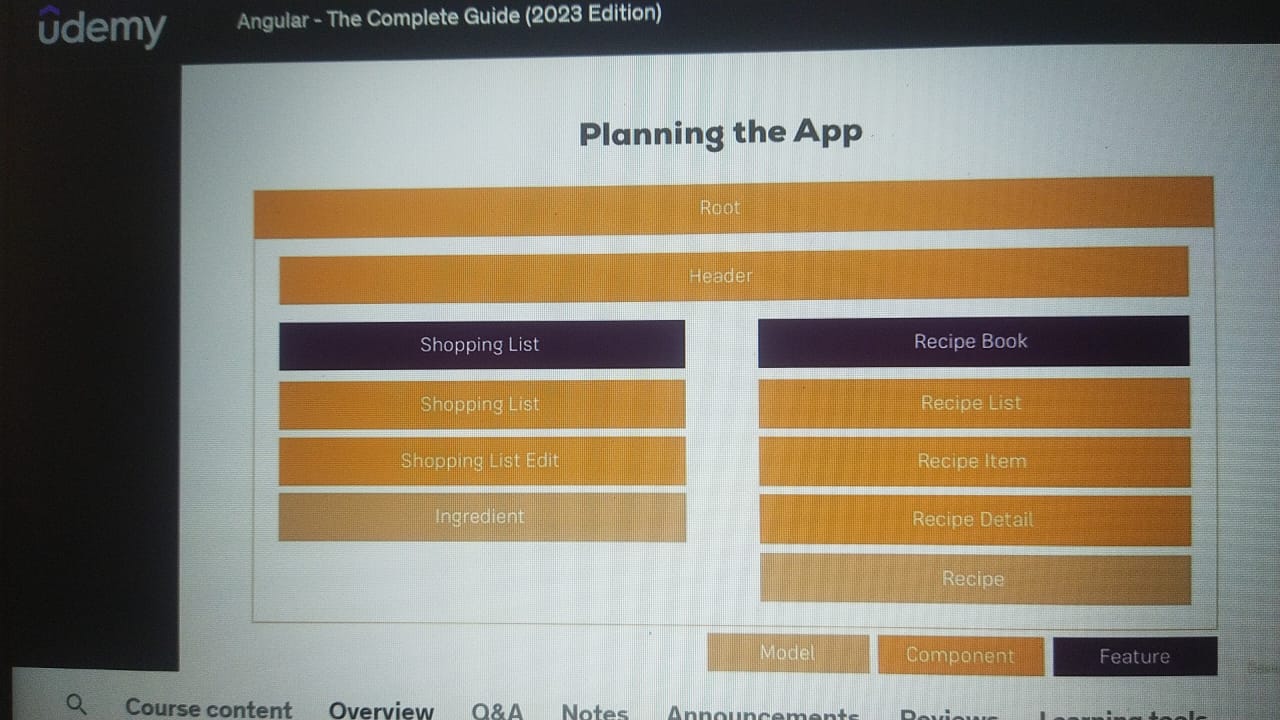
* This project is to build a recipe book and shopping list app.
* Basically, there will be two sections, shopping list and recipe book
* we will be able to i) manage our recipes ii) view them in detail, and iii) also to manage our shopping list iv) even push ingredients from a recipe directly to the shopping list.

1. **Planning the application**

* As part of the plan, we're going to think about which components we will need for the different features of this application.

**Steps in Planning:**

* The first step in creating an Angular application, is to lay out the structure of this app, especially plan which components we will need



* So let's start with the features of this app.
* We will have a root component, which holds the overall application
* We will have two sections: shopping list and recipe book.
* Since we have two sections, it probably makes sense to have some kind of header component where we can navigate between these two sections.
* Now let's dive into the individual features.

**Shopping List:**

* Let's start with the shopping list here. I guess it makes sense to have the overall shopping list component. Remember the purple boxes only the feature.
* So we need a component holding our shopping list. And maybe in this shopping list component, we also want to have a shopping list edit part, which allows us to add new items.
* So a input field and a button, because that again has its own logic.
* We will have to decide if we're editing an existing item or adding a new one.
* We need to submit this so it makes sense to create its own component for this.
* So as we have our overall list component, mainly holding the data off the list and the added component being responsible for managing this data, kind of though the added component will be nested inside the list component.

**Recipe book.**

* Here, it certainly makes sense to have again, just like with the shopping list a recipe list component, which shows us a list of all our recipes.
* Here we might later also put each individual recipe into its own item, though since it holds a little bit more information than just one line of HTML code, and a recipe detail area. So a area where once we select a recipe we can see informations about that.
* Later in the course, we will also add another component here because we will then somehow need a component which allows us to add existing recipes or add new ones.
* But since this is really advanced and we would not be able to fill it with much life right now, I will omit it for now and focus on the displaying part. So a list with items and then the details section which displays the details for the currently selected item.

**Models:**

* A model for holding Ingredient
* A model for recipe

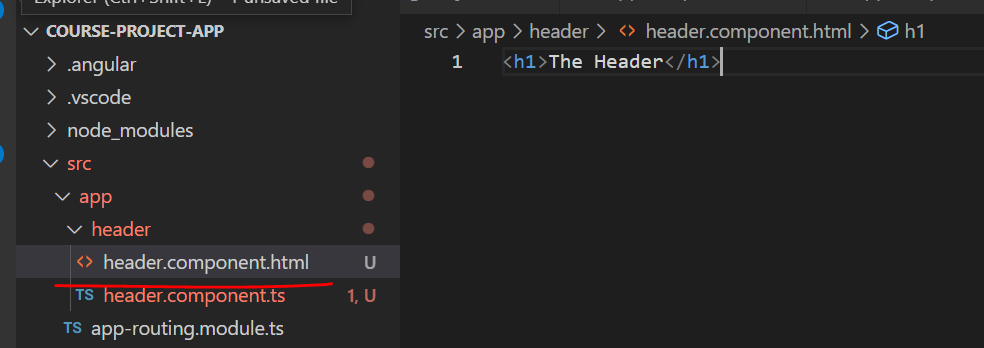
1. Creating new project, installing boot strap and then bringing up the application

* Followed steps mentioned in : Video 47: Setting up the application

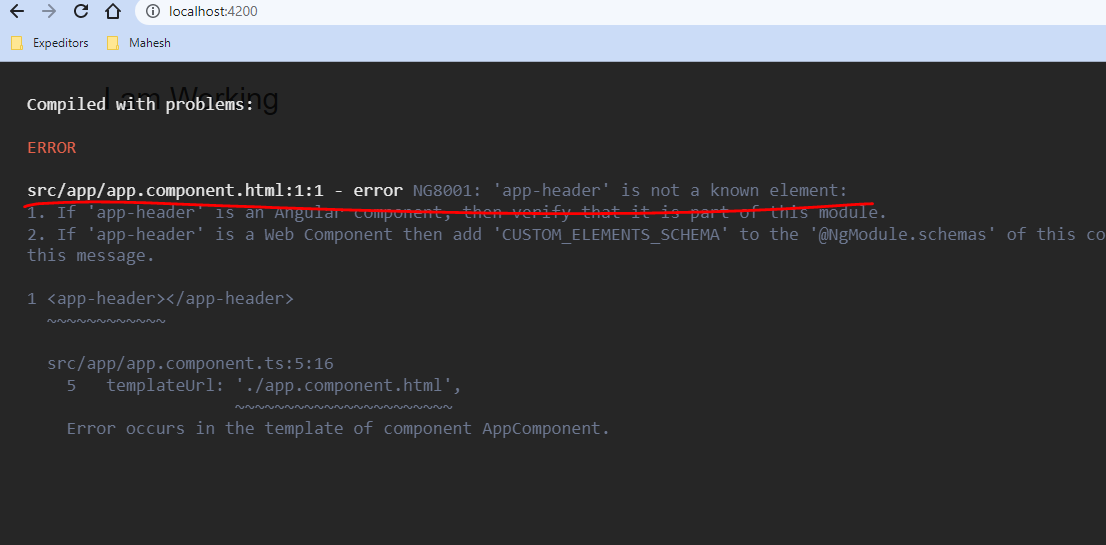
1. **Creating the Components**

**Header Component:**

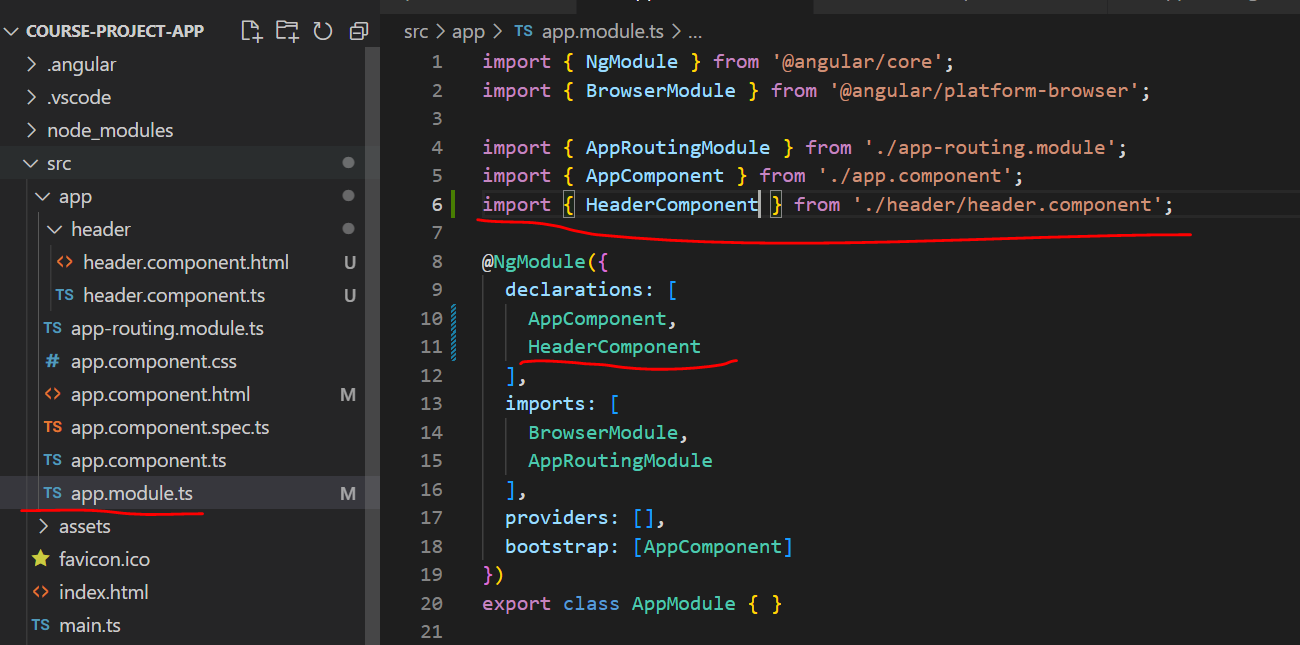
* I will create a header folder under the app folder and create file: header.component.ts file
* Component is simply the TS class – lets name it HeaderComponent
* import { Component } from "@angular/core";
* @Component({
* selector:'app-header',
* templateUrl:'./header.component.html'
* })
* export class HeaderComponent{
* }
* This component will be recognized by Angular only when we add the @component decorator. We have to import component between color braces from @angular/core.
* Then we need to pass a JavaScript object to this decorator, To configure this component.
* Specifically, we need to add a template and I'll put this in external file. So template URL in this case and a selector so that we can use this component. So the selector here is app header because I want to ensure that I have a unique selector and that I don't override an existing HTML element, for example.



* The template URL here, I want to point to the header.component.HTML file.
* I will include header component in app component(app.component.html), maybe above the container because in the container, I later only want to contain my content component.
* Now if we save and check in browser, we can see an error:



* And the error is pretty clear. App header is not a known element.
* Important point we need to remember is: you have to register all the features you're going to use in the app module. And since we didn't use the CLI to create this component it wasn't added there automatically. So we have to add it manually to our declarations array. Also we have add import for HeaderComponent in app.module. ts file



* Now we can see the output in the browser:

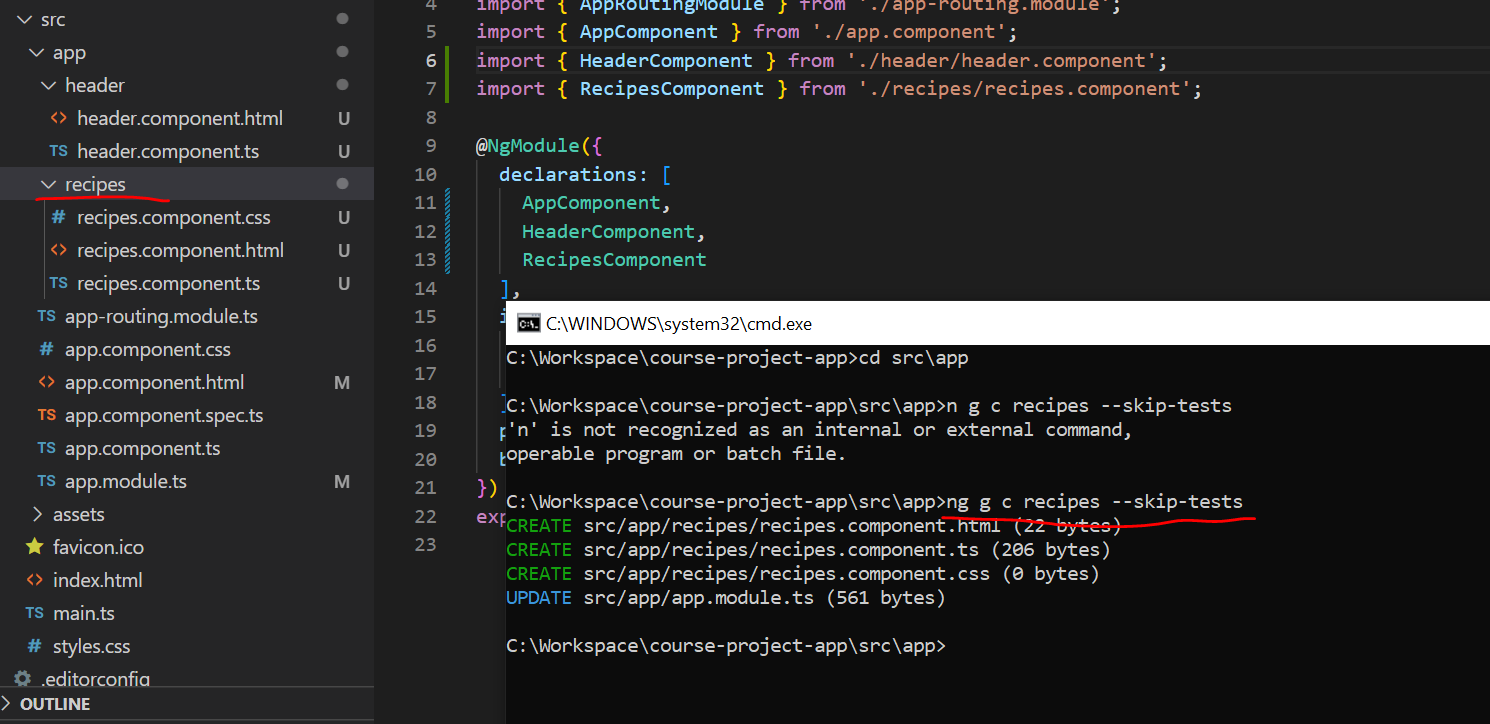
Graphical user interface, text, application

Description automatically generated

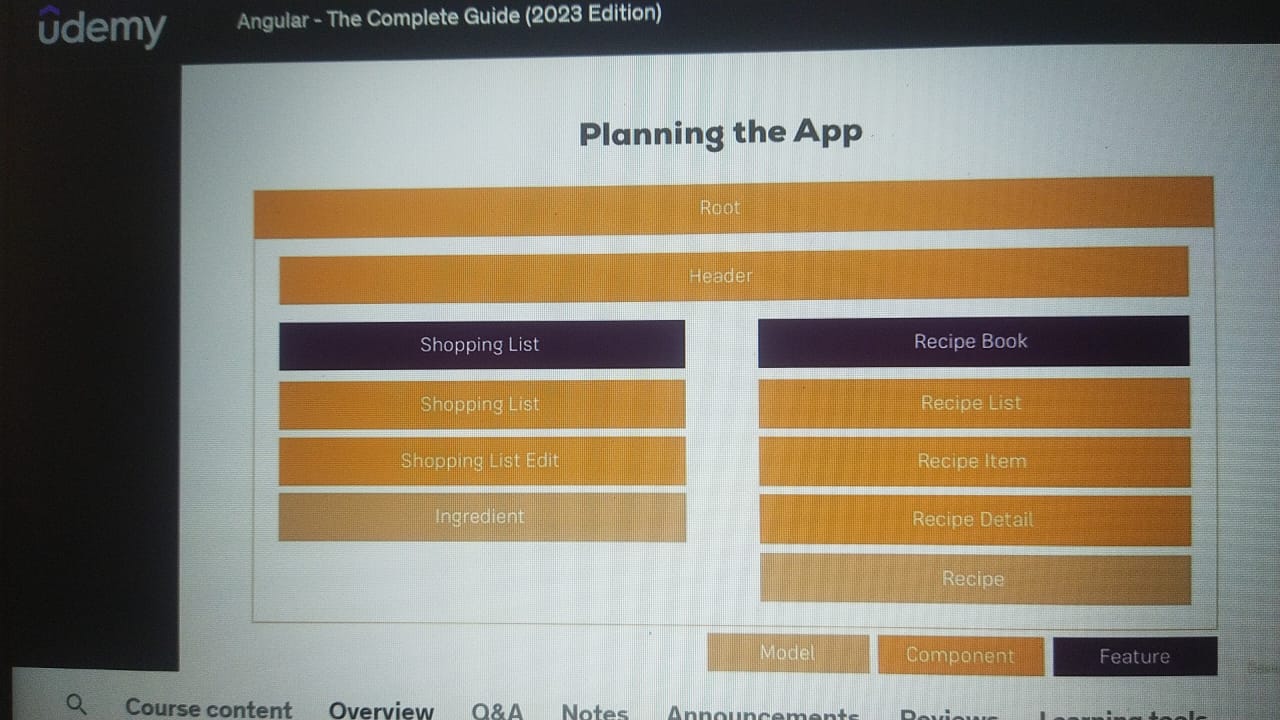
* Now lets add other components via CLI

**Recipes Component:**

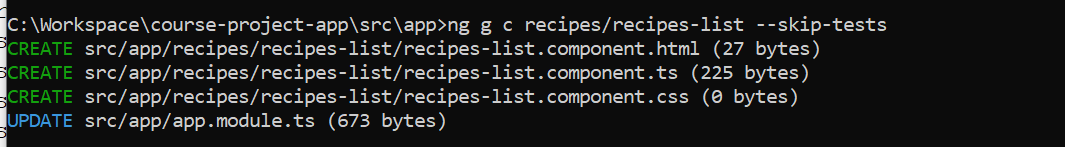
* Lets create recipes component using CLI like below:



* Now this will be our overall recipes feature component. Note that we did not have this recipes feature in the diagram we discussed.



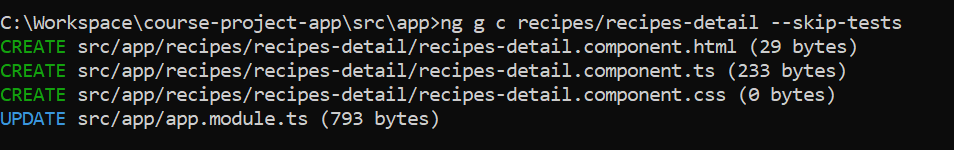
**Recipe-list component:**

* Next I will create the recipe list component as laid out on the slides.
* Now here, I don't want to create the recipe list on the or inside the app folder though which is a default behavior by the CLI.
* If I now would hit enter. Instead, that should go inside the recipe's folder because that is where all new recipe related components should go.
* We can easily tell the CLI to create this component in a sub folder by basically passing a path here.
* So instead of just saying recipe list we can say recipes/ recipe list. And since you already have the recipes folder this will create the recipes list folder inside of the recipes folder.
* 

Graphical user interface, text, application

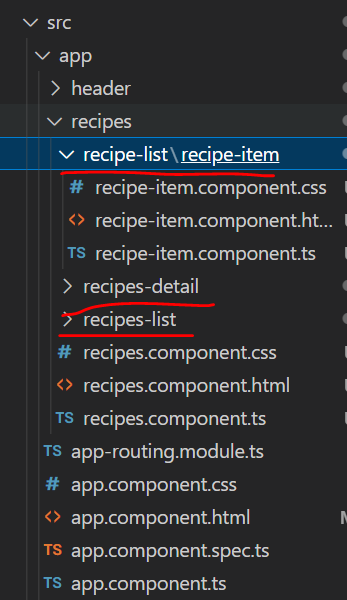
Description automatically generated

* Now with the recipe list added I also want to add the recipe detail here



* Next I want to add the recipe item under the recipe-list folder
* Text

  Description automatically generated
* Now we can observe the same in folder structure



* Now let's create shopping list component
* Text

  Description automatically generated
* Inside the shopping list component, I will create shopping list edit component which will allow us to add new ingredients or edit existing ones.

Text

Description automatically generated

* That should now go into the shopping list component.
* Now we can see all the components added in app.module.ts file

