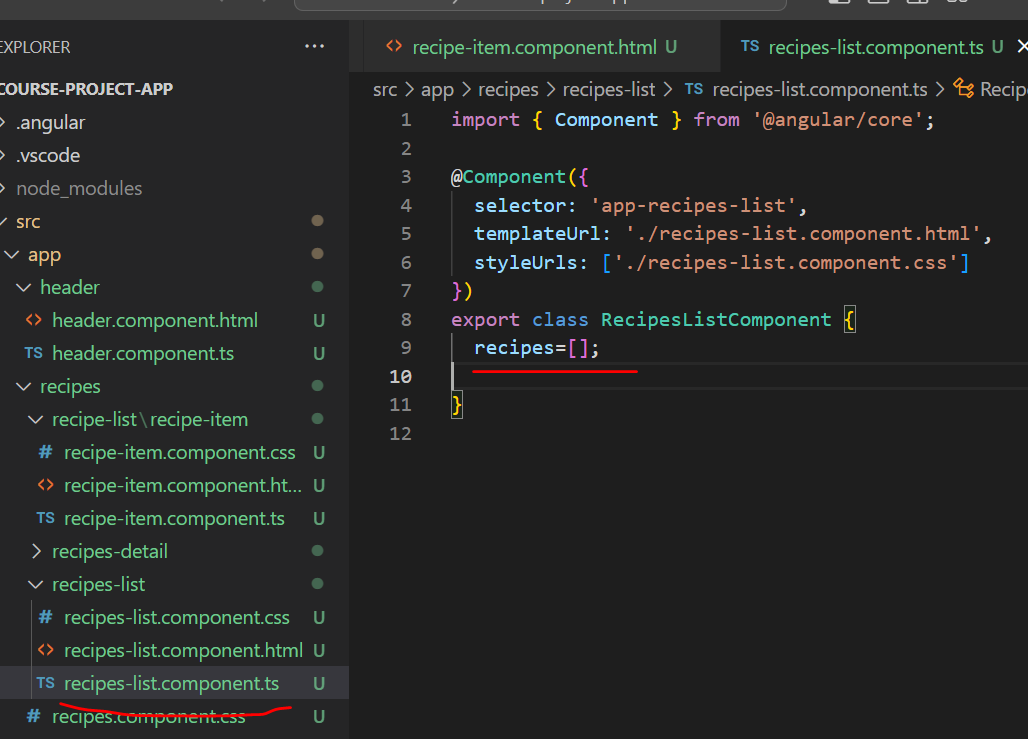
1. **Creating a “Recipe” Model:**

* We added the header in the last lecture. Now I want to work on my recipes.
* So we get the RecipesComponent which holds the recipe list. Now I also want to fill the list with some life. For this I'll go into my RecipeListComponent and define an array of recipes
* 
* Now lets create a model for recipes. Model is simply a TS file. We will create recipe.model.ts file
* Now you could think that we're going to add something like,@Model here, and we're not going to do this, there is no decorator like this.
* And we don't need to do this, because we can use vanilla TypeScript for this.
* A model in the end should just be a blueprint for objects we create.
* A recipe should have a name, and I'll add an accessor to be really clear about that this is publicly available,so that this can be accessed from outside if using this
* as an instantiated object. So I'll add public in front of it and then name as the property name.
* I'll also assign the type of this by adding a colon and the type will be string.
* And that is just how you assign types in TypeScript.
* You add a colon after the property name, and then the type you want to assign.
* So string in this case. I also want to have a description in each recipe. So I'll add my description property, which also is a string because a description is just a text of course.
* Now maybe we also want to store a image for each recipe. So we should have a imagePath
* since we won't store the image itself here of course, that wouldn't work. We can't store files in our code, but we want to store the path pointing to the image.
* And we will simply use images from the web here.
* So this will hold a URL in the end.
* So that is also a string, a text in the end.
* That is the basic model for now. The basic blueprint.
* **Constructor:**
* I'll also add a constructor to it,
* So inside of the body of this constructor we have to assign the arguments we receive here to the properties of our object now, to the properties of our class.

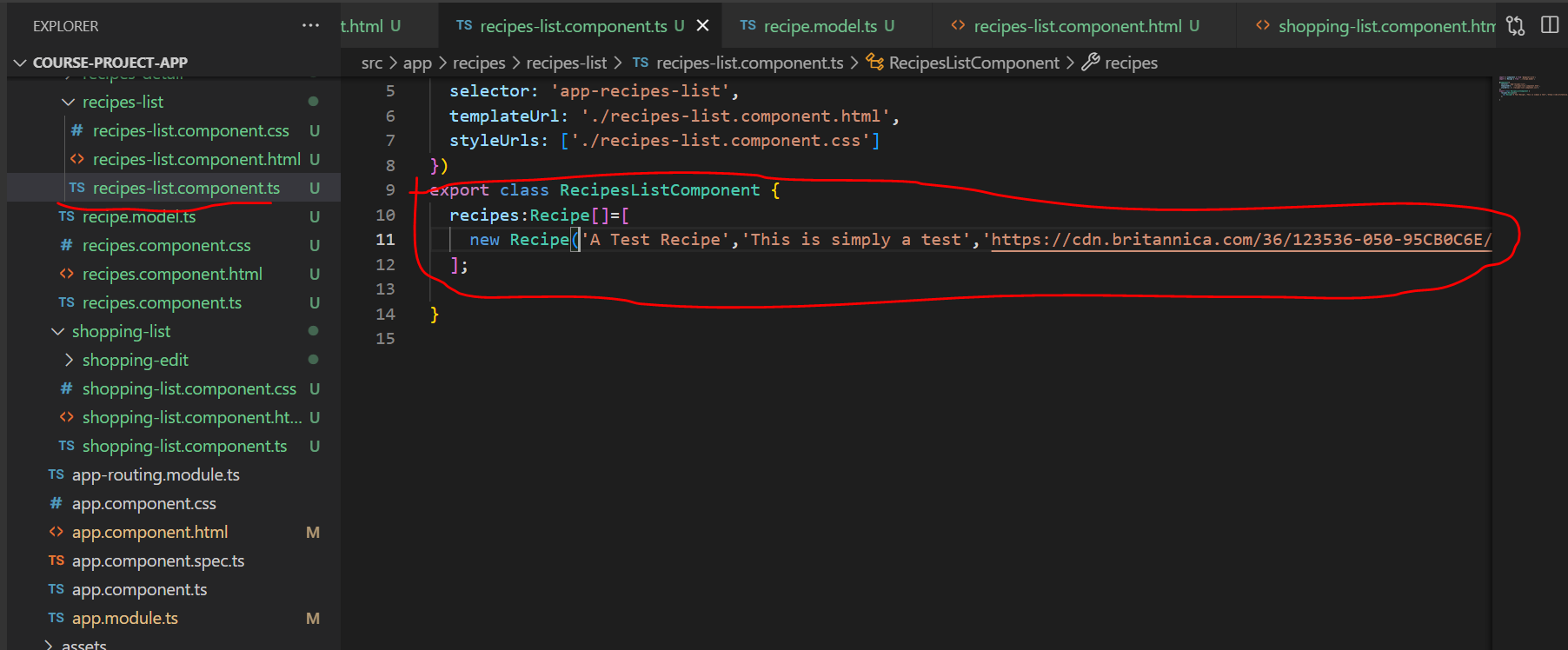
Text

Description automatically generated

1. **Adding Content to Recipe model:**

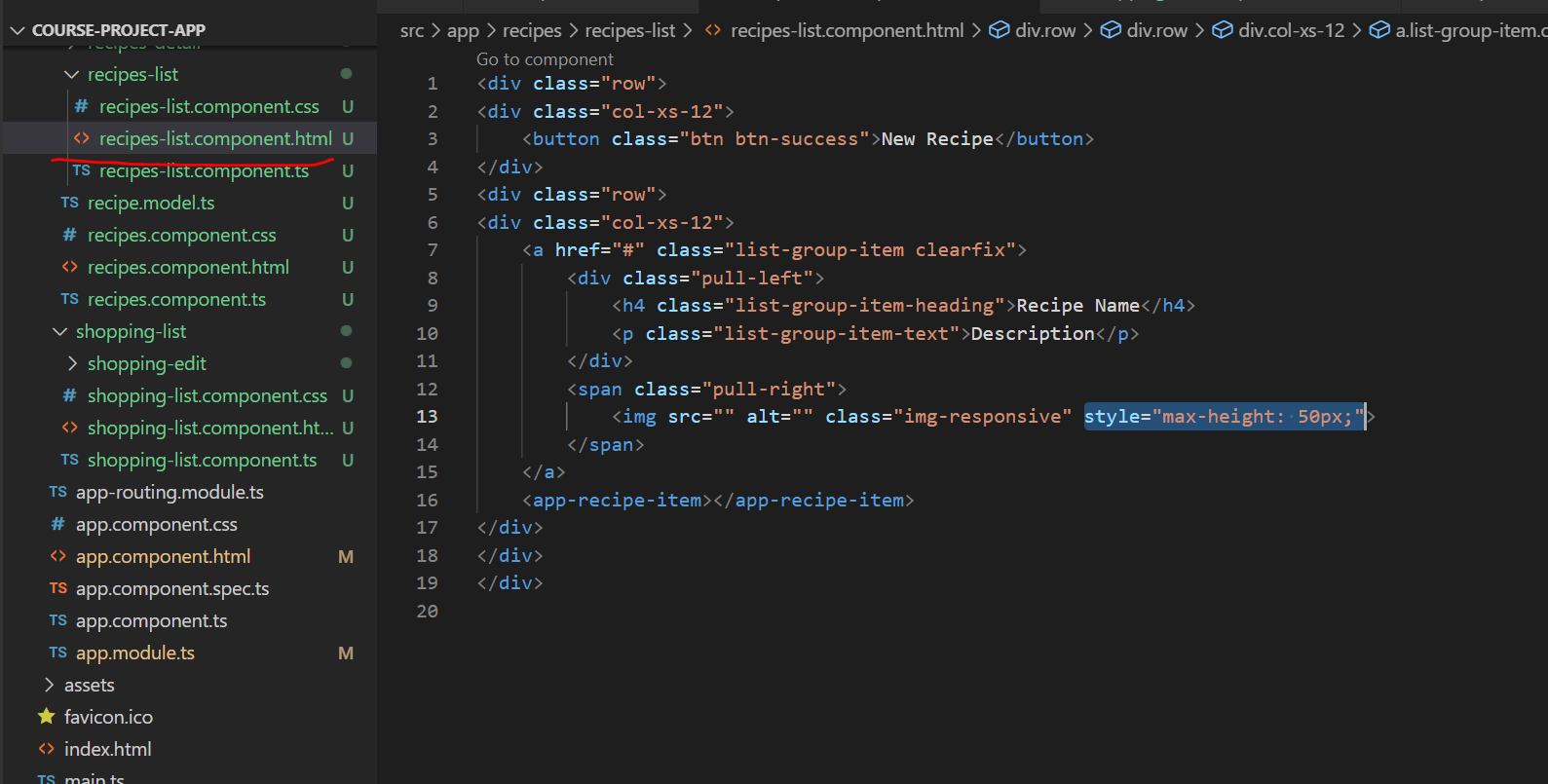
So far, we added our Recipe model, now lets use it.

For the recipes array we defined in recipe-list.component.ts, we can make it of type : Recipe, which is our recently created model and we are populating it in RecipesListComponent’s recipes array



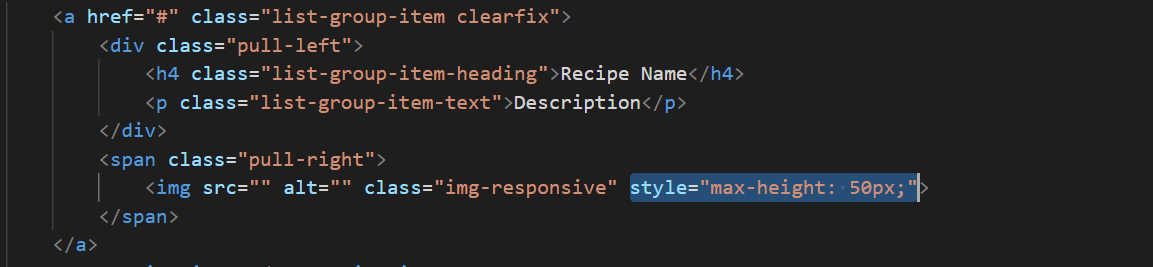
Of course, we won't be able to see anything for now. To see that, we should actually do something in the template of the recipe list component.

Here I'm right now only outputting my recipe item and I want to loop through all these items and use them to display my recipes.



1. **Outputting list of recipes with ngFor:**

* To repeat this part of the code as needed for all the data in the array, we will be using ngFor directive



* Here recipes is the array defined in the typescript file
* <a href="#" class="list-group-item clearfix" \*ngFor="let recipe of recipes">
* We can access properties of recipe object using String interpolation like this:
* <h4 class="list-group-item-heading">{{recipe.name}}</h4>
* <p class="list-group-item-text">{{recipe.description}}</p>

**Displaying image:**

* For displaying image we can use either string interpolation or property binding

**String interpolation:**

src="{{recipe.imagePath}}"

**Property Binding:**

* [src]="recipe.imagePath"
* In our code lets use property binding approach
* Thus with out entire code changes below:

Text

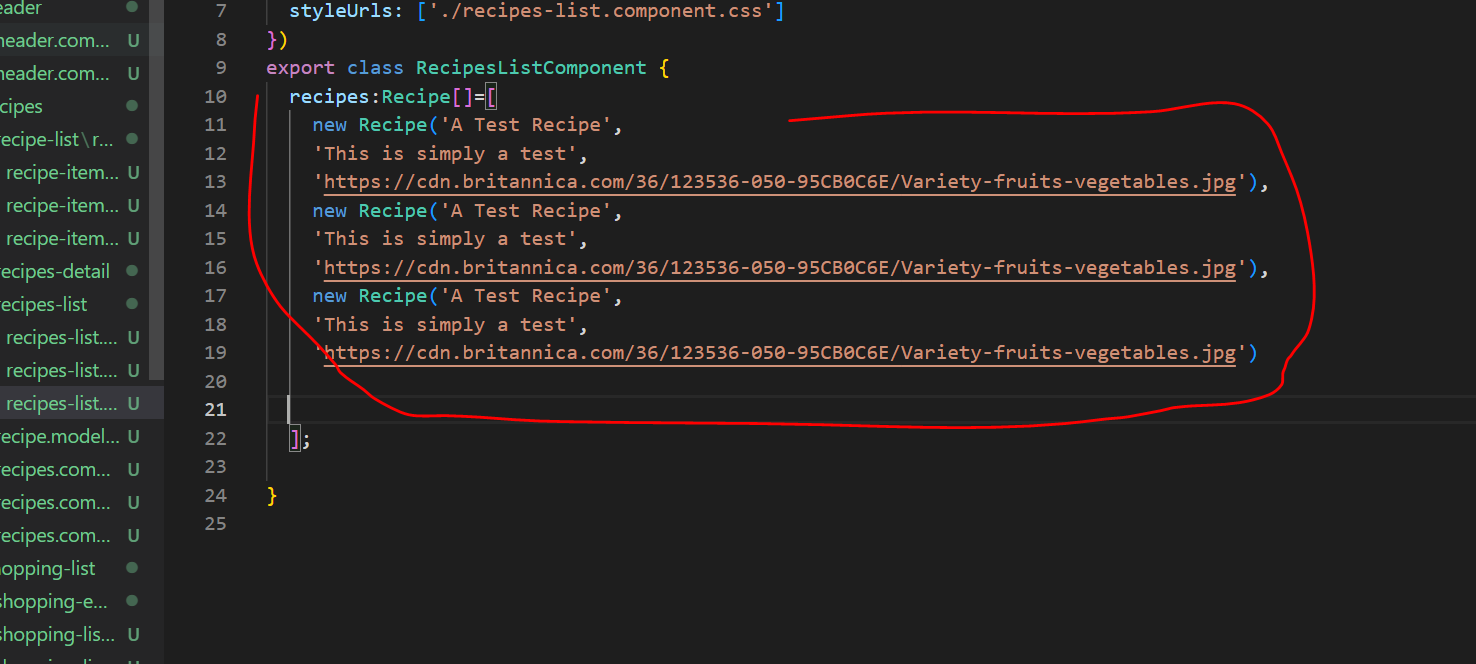
Description automatically generated

* We will see below output in the browser:

Graphical user interface, text, application

Description automatically generated

* So that's the list. And you should see that if you add more items to that list that it automatically grows.
* Eg:



* We can see the same in browser:

Graphical user interface, text, application

Description automatically generated