Angular

By: Puskar Adhikari

Installation

From the official website

- Angular

Create a New Project

Make a new angular app using the command

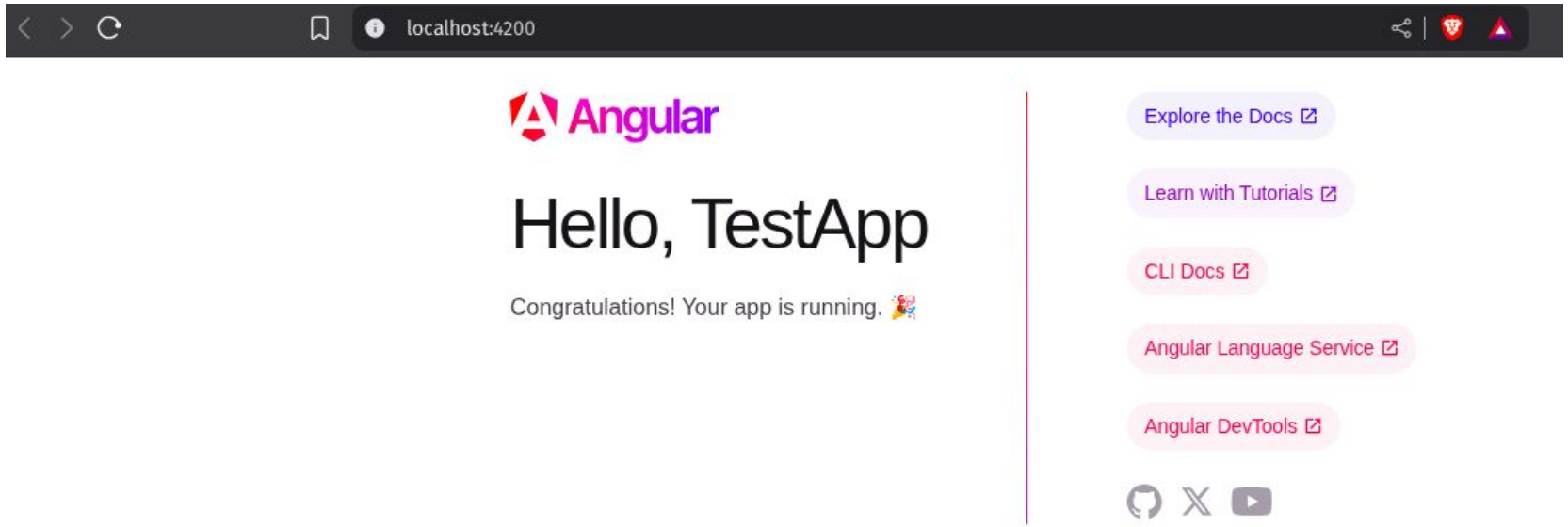
- ng new <app_name>

```
~ (2m 17.88s)
ng new TestApp
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this co
hint:
hint:   git branch -m <name>
    Successfully initialized git.


~ (0.028s)

~/TestApp  git:(master)
```

Running Angular

To run the application, use command

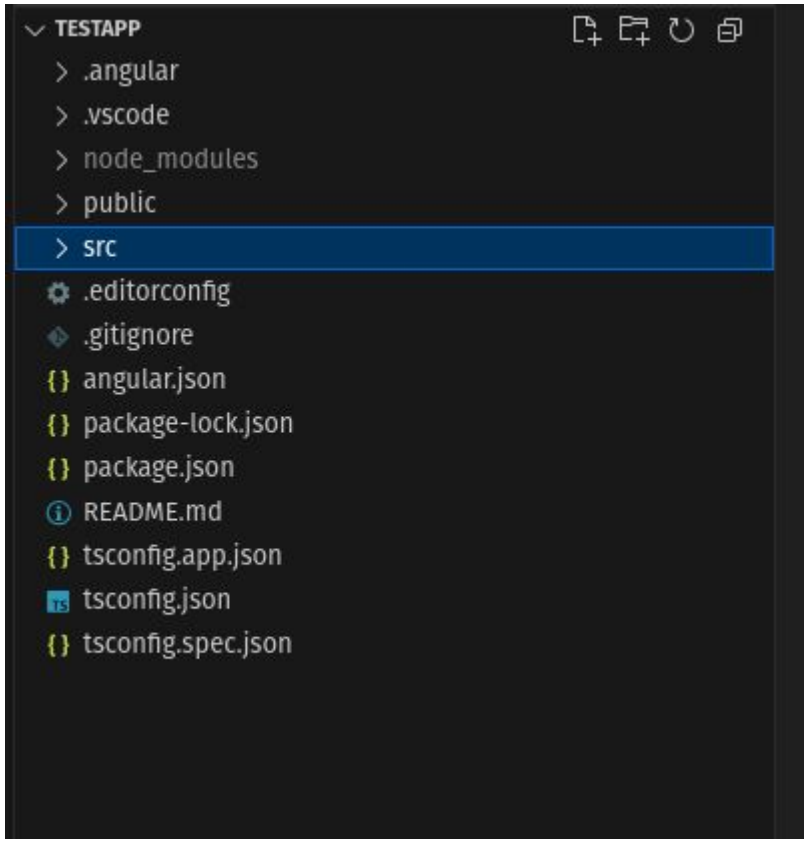
- ng serve



It will run angular application on port 4200 by default.

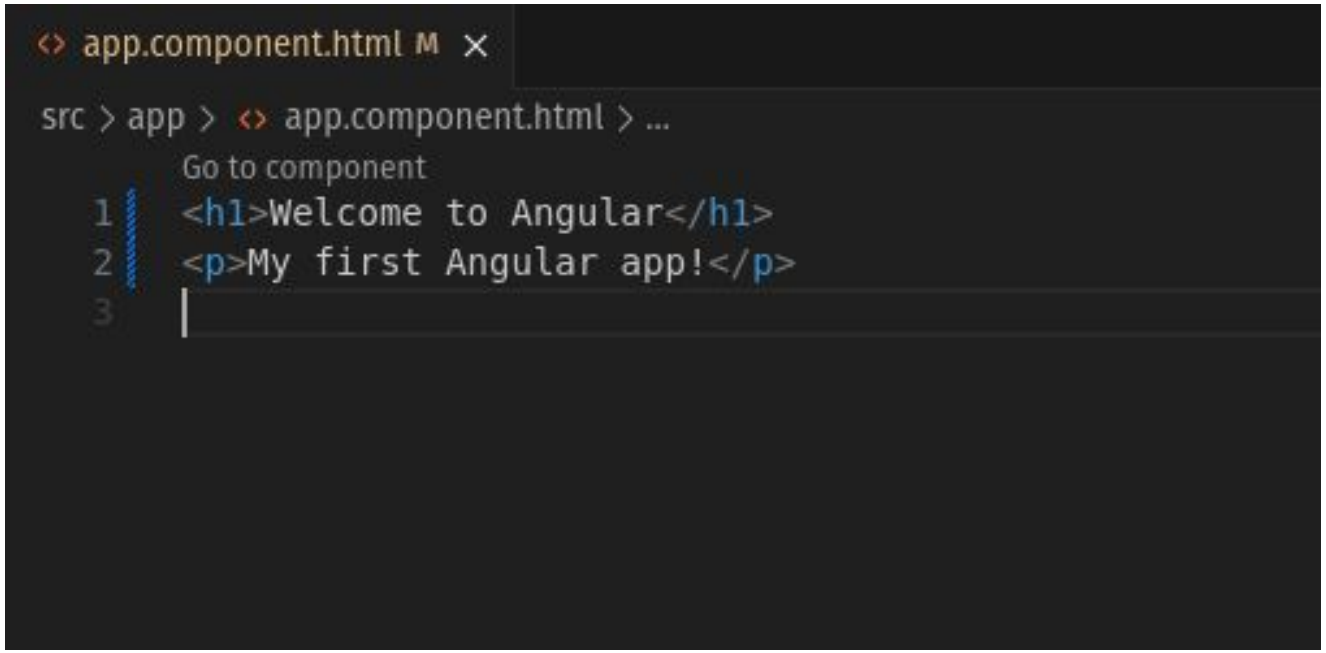
Project Structure

- src/app: main codebase
- app.component.ts/html/css: root component
- app.module.ts: module declarations
- main.ts: entry point



HomePage

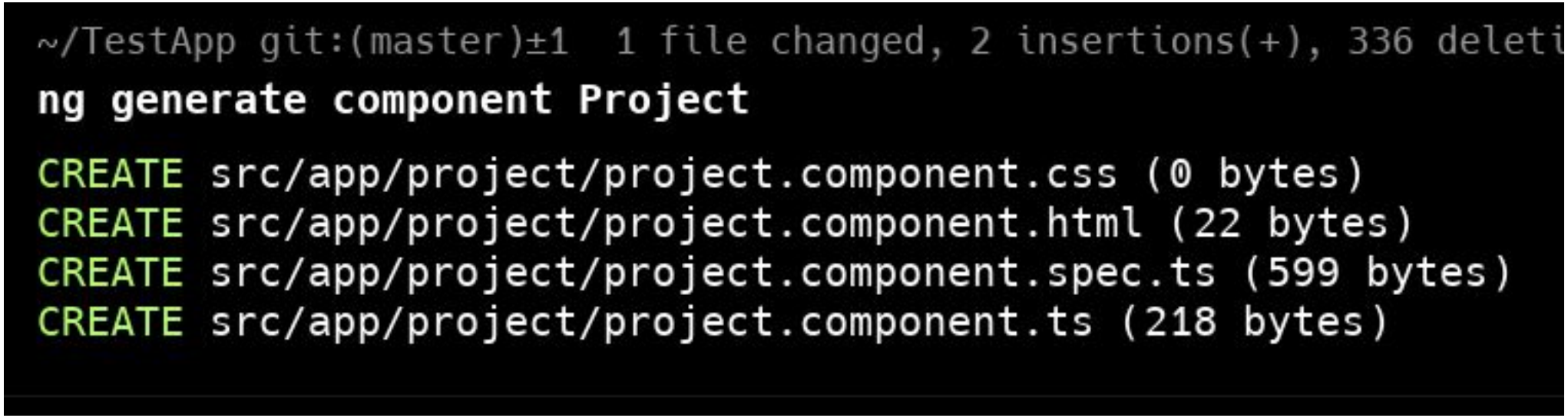
Replace all the contents of src/app/app.component.html with some simple html like this.



Components

Generate component in angular using command:

- ng generate component <component-name>
- ng g c <component-name>

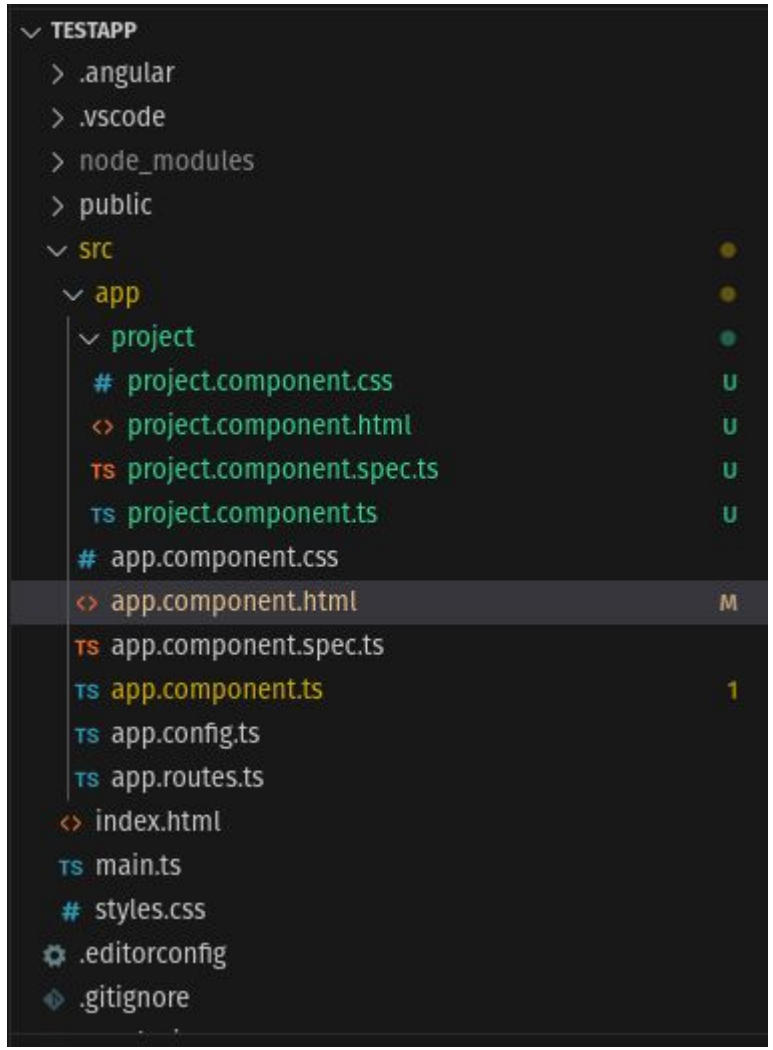


Components

You will now see a project folder with css, html and ts files there.

A component in angular generally has a css file, typescript file and a html file.

- CSS File: For styling
- TS File: Logics and Functions
- Html File: Html syntax and templates



Components

In src/app/app.component.ts, add *ProjectComponent* in the imports field.

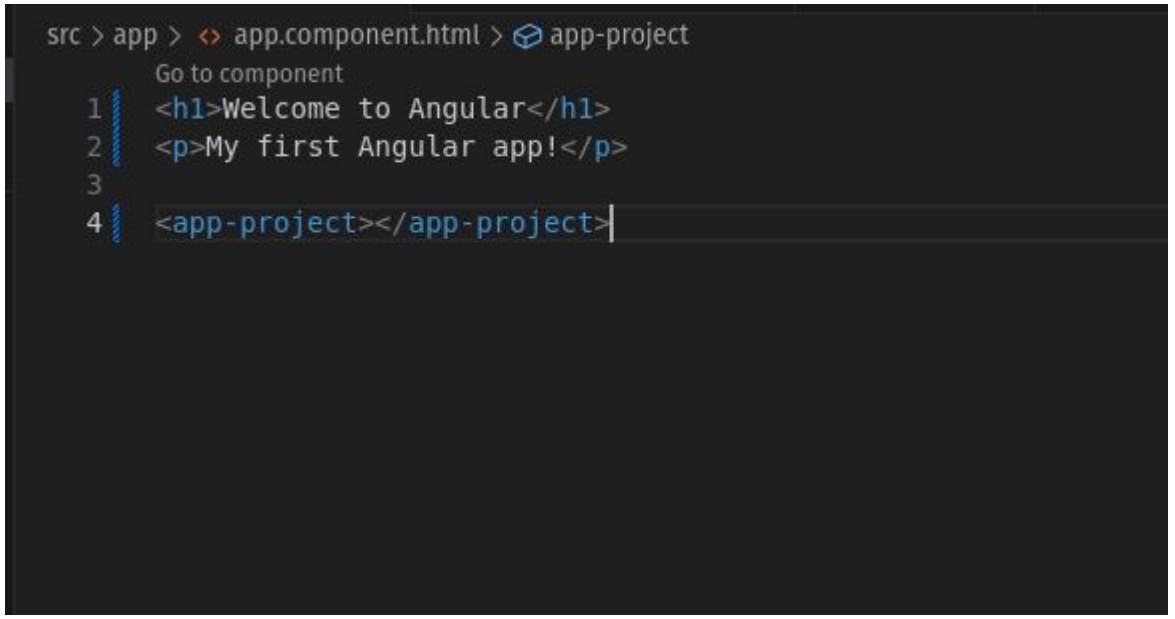
This is like declaring your component in the main app.

Components

In src/app/app.component.html, Add the line:

- <app-project></app-project>

Check the homepage now.

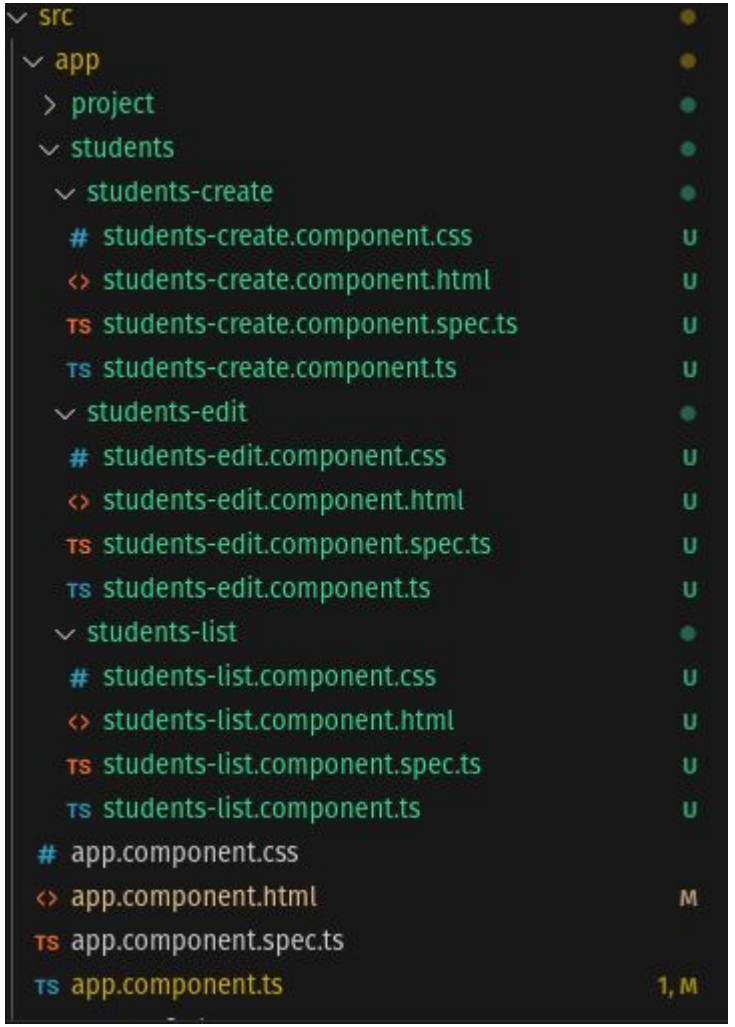


Components

- Make a new folder called *students* inside src/app.
- Generate components students-create,

  students-list, students-edit inside the students folder.

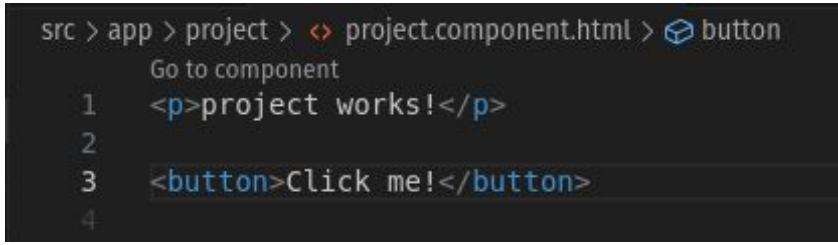
- Try to add them to projects or the whole app.

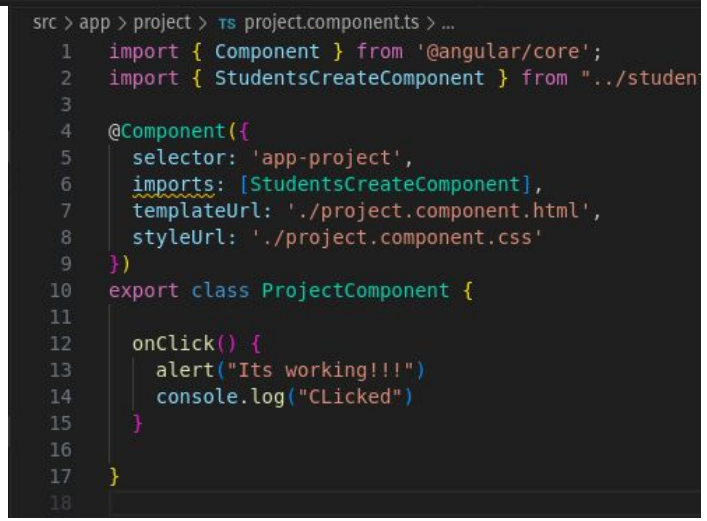


Project Component

- Add a button in project.component.html
- Add a property click to the button and bind it to a method “onClick”.
- Add a method onClick() in project.component.ts

https://v17.angular.io/guide/binding-syntax





Interpolation - {{ ... }}

- Used to **display data** from the component in the template.
- {{ name }} pulls value from the component and shows it as text in the DOM.

```
export class ProjectComponent {

  name: string = "puskar"

  onClick() {
    alert("Its working!!!")
    console.log("CLicked")
  }

}
```

```
src > app > project > project.component.html > p
      Go to component
1     <p>project works!</p>
2
3     <button (click)="onClick()">Click me!</button>
4
5     <p>Hello {{ name }}</p>
```

Property Binding - [property]="..."

- [disabled]="isDisabled" binds the button's disabled property to your component variable.
- If isDisabled is true, the button is disabled.

src > app > project > project.component.html > ...

Go to component

```
1  <p>project works!</p>
2
3  <button (click)="onClick()">Click me!</button>
4
5  <p>Hello {{ name }}</p>
6
7  <h2>Property Binding</h2>
8  <button [disabled]="isDisabled">Property Binding Button</button>
9
```

```
10 export class ProjectComponent {
11
12   name: string = "puskar"
13   isDisabled: boolean = false
14
15   onClick() {
16     alert("Its working!!!")
17     console.log("CLicked")
18   }
19
20 }
21
```

Event Binding - (event)="method()"

- Used to **listen to DOM events** and call component methods.
- (click)="onClick()" calls the method when user clicks the button.

- It updates message, which re-renders automatically in the template.

```
10  export class ProjectComponent {
11
12    name: string = "puskar"
13    isDisabled: boolean = false
14    message = '';
15
16    onClick() {
17      alert("Its working!!!")
18      console.log("CLicked")
19      this.message = 'Button clicked!';
20    }
21
22  }
```

Two-Way Data Binding - [(ngModel)]="..."

- Used to **bind both ways**: from component → template and template → component.
- First, import FormsModule in project.component.ts

```
10  export class ProjectComponent {
11
12    name: string = "puskar"
13    isDisabled: boolean = false
14    message = '';
15
16    onClick() {
17      alert("Its working!!!")
18      console.log("CLicked")
19      this.message = 'Button clicked!';
20    }
21
22  }
```

Typing in the input updates name in the component instantly. name changing also re-renders in the template.

```
11  <h2>Two-Way Binding</h2>
12  <input [(ngModel)]="name" placeholder="Enter name" />
13  <p>Hello, {{ name }}!</p>
```

Directives - *ngIf

Click the button to show/hide messages dynamically.

```
14
15  <h2>*ngIf Demo</h2>
16  <p *ngIf="isLoggedIn">Welcome back, {{ name }}!</p>
17  <p *ngIf="!isLoggedIn">Please log in.</p>
18
19  <button (click)="toggleLogin()">Toggle Login</button>
20
```

```
6   @Component({
7     selector: 'app-project',
8     imports: [StudentsCreateComponent, FormsModule, CommonModule],
9     templateUrl: './project.component.html',
10    styleUrl: './project.component.css'
11  })
12  export class ProjectComponent {
13
14    name: string = "puskar"
15    isDisabled: boolean = false
16    message = '';
17    isLoggedIn = false;
18
19    onClick() {
20      alert("Its working!!!")
21      console.log("CLicked")
22      this.message = 'Button clicked!';
23    }
24
25    toggleLogin() {
26      this.isLoggedIn = !this.isLoggedIn;
27    }
28
29  }
30
```

Checkpoint (Optional)

```
<p>project works!</p>

<button (click)="onClick()">Click me!</button>
<p>{{ message }}</p>

<p>Hello {{ name }}</p>

<h2>Property Binding</h2>
<button [disabled]="isDisabled">Property Binding
Button</button>

<h2>Two-Way Binding</h2>
<input [(ngModel)]="name" placeholder="Enter name"
/>
<p>Hello, {{ name }}!</p>
```

```
import { Component } from '@angular/core';
import { StudentsCreateComponent } from
"../students/students-create/students-create.component";
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

@Component({
 selector: 'app-project',
 imports: [StudentsCreateComponent, FormsModule,
CommonModule],
 templateUrl: './project.component.html',
 styleUrl: './project.component.css'
})
export class ProjectComponent {
```

Directives - *ngFor

A “For loop” in html

```
<h2>*ngFor Demo</h2>
<ul>
  <li *ngFor="let student of students; let i = index">
    {{ i + 1 }}. {{ student }}
  </li>
</ul>
```

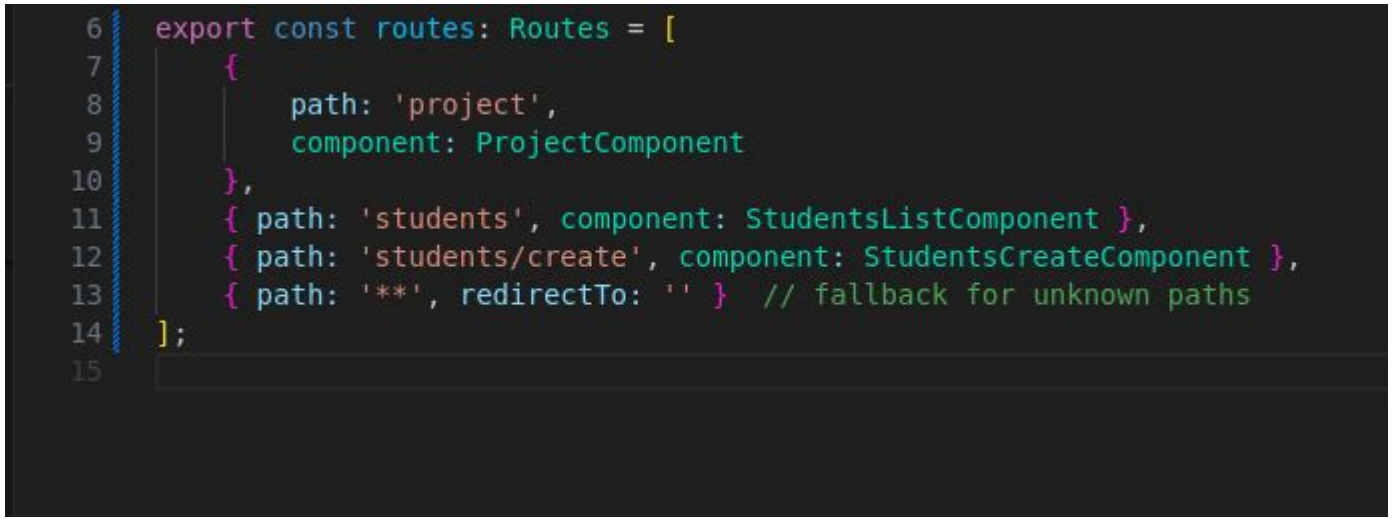
```
<h2>Student List</h2>
<ul>
  <li *ngFor="let student of students">
    {{ student }}
  </li>
</ul>
```

```
export class ProjectComponent {

  name: string = "puskar"
  isDisabled: boolean = false
  message = '';
  isLoggedIn = false;
  students = ['Alice', 'Bob', 'Charlie', 'Diana'];

  onClick() {
    alert("Its working!!!")
    console.log("CLicked")
    this.message = 'Button clicked!';
  }

  toggleLogin() {
    this.isLoggedIn = !this.isLoggedIn;
  }

}
```

Angular Routing

Update your app.routes.ts

```
export const routes: Routes = [
  {
     path: 'project',
     component: ProjectComponent
  },
  { path: 'students', component: StudentsListComponent },
  { path: 'students/create', component:
StudentsCreateComponent },
  { path: '**', redirectTo: '' }  // fallback for unknown paths
];
```

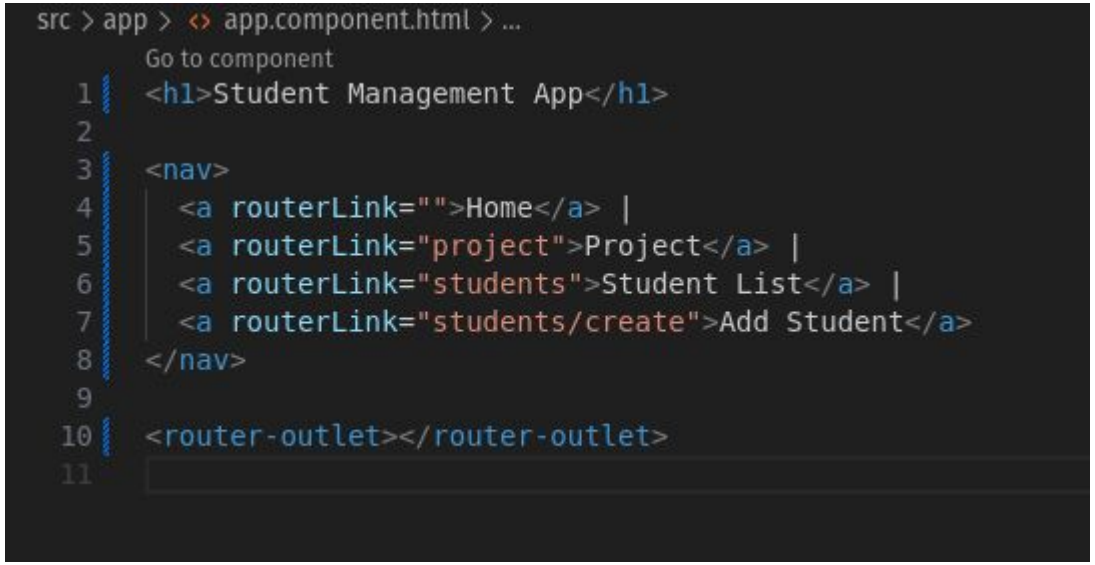


Angular Routing

Replace everything in

app.component.html with:

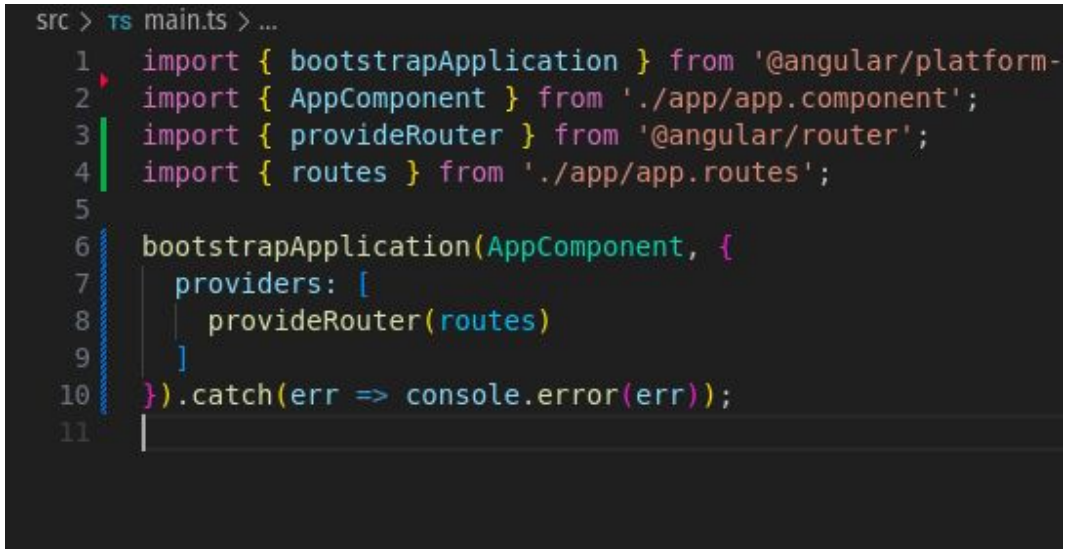
```
<h1>Student Management App</h1>
<nav>
 <a routerLink="">Home</a> |
 <a routerLink="project">Project</a> |
 <a routerLink="students">Student List</a> |
 <a routerLink="students/create">Add Student</a>
</nav>

<router-outlet></router-outlet>
```



Angular Routing

Add providerRouter in your main.ts file

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app/app.component';
import { provideRouter } from '@angular/router';
import { routes } from './app/app.routes';

bootstrapApplication(AppComponent, {
 providers: [
   provideRouter(routes)
 ]
}).catch(err => console.error(err));
```



Templates

- https://angular.dev/guide/templates

Assignment

Home Page: Welcome message, Toggle Login

Project Page: Your project info

Student List Page: List of students (or empty message) (Via Array)

Navigation: Links between pages work, Add a goto students page link in project page (Use routerlink or navigate)

Bonus: Delete student & success messages

Submit the url of your gitlab repo with all the relevant screenshots

in a README file.