

# 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)

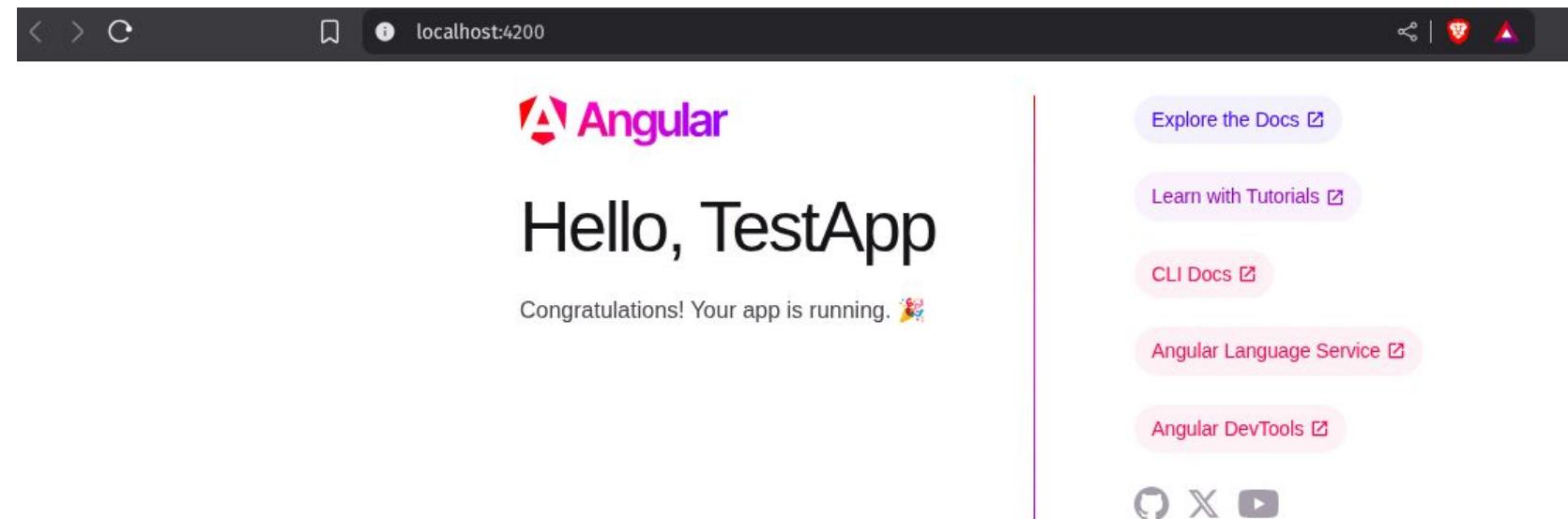
~/TestApp git:(master)~
```

# Running Angular

To run the application, use command

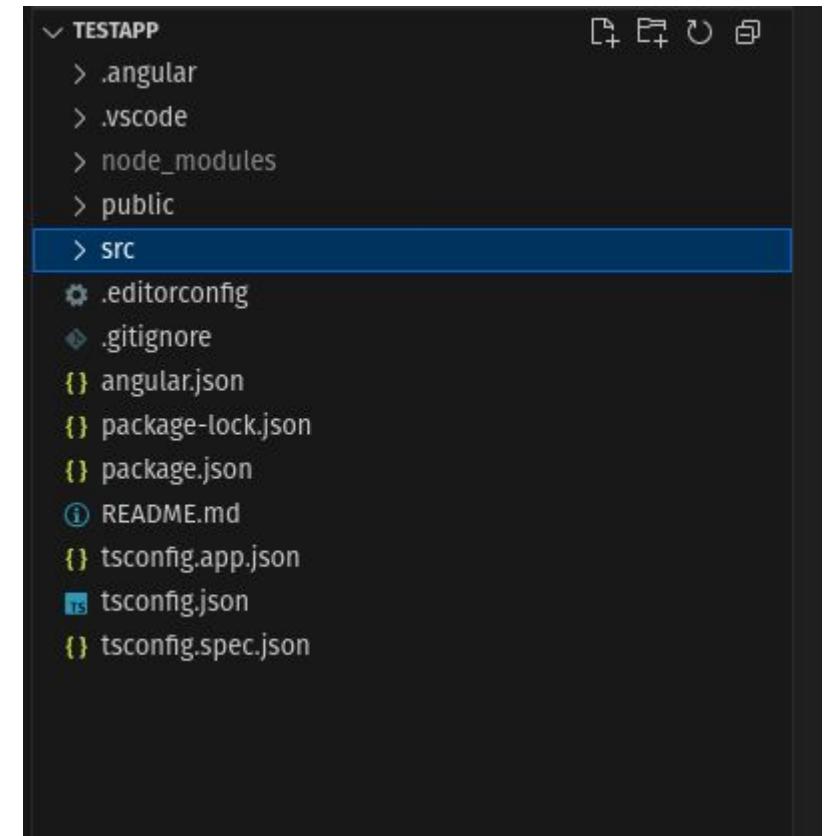
 **ng serve**

It will run angular  
application on port  
4200 by default.



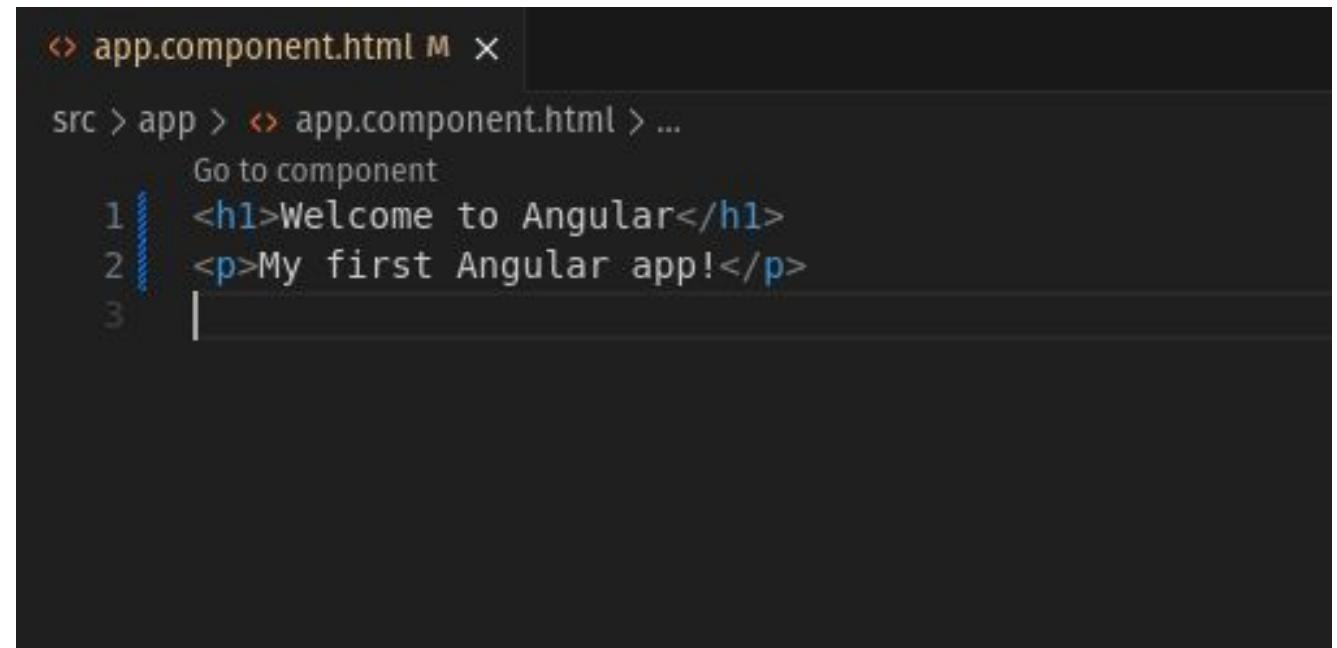
# Project Structure

- `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.



The screenshot shows a code editor window with a dark theme. The file is named `app.component.html`. The code contains:

```
<h1>Welcome to Angular</h1>
<p>My first Angular app!</p>
```

The code editor shows line numbers 1 and 2. A cursor is positioned at the end of line 2. A tooltip "Go to component" is visible above the code area. The status bar at the bottom of the editor shows "1 file" and "100%".

# Components

Generate component in angular using command:

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

```
~/TestApp git:(master)±1 1 file changed, 2 insertions(+), 336 deletions(-)
ng generate component Project
CREATE src/app/project/project.component.css (0 bytes)
CREATE src/app/project/project.component.html (22 bytes)
CREATE src/app/project/project.component.spec.ts (599 bytes)
CREATE src/app/project/project.component.ts (218 bytes)
```

# 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

```
└─ TESTAPP
    ├ .angular
    ├ .vscode
    ├ node_modules
    └ public
        └─ app
            └─ project
                # project.component.css
                <> project.component.html
                TS project.component.spec.ts
                TS project.component.ts
                # app.component.css
                <> app.component.html
                TS app.component.spec.ts
                TS app.component.ts
                TS app.config.ts
                TS app.routes.ts
                <> index.html
                TS main.ts
                # styles.css
                ⚙ .editorconfig
                ⚙ .gitignore
```

# Components

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

This is like declaring your component in the main app.

```
src > app > ts app.component.ts > AppComponent
  1 import { Component } from '@angular/core';
  2 import { RouterOutlet } from '@angular/router';
  3 | import { ProjectComponent } from './project/project.component';
  4 |
  5 @Component({
  6   selector: 'app-root',
  7   imports: [RouterOutlet, ProjectComponent],
  8   templateUrl: './app.component.html',
  9   styleUrls: ['./app.component.css']
 10 })
 11 export class AppComponent {
 12   title = 'TestApp';
 13 }
 14
```

# Components

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

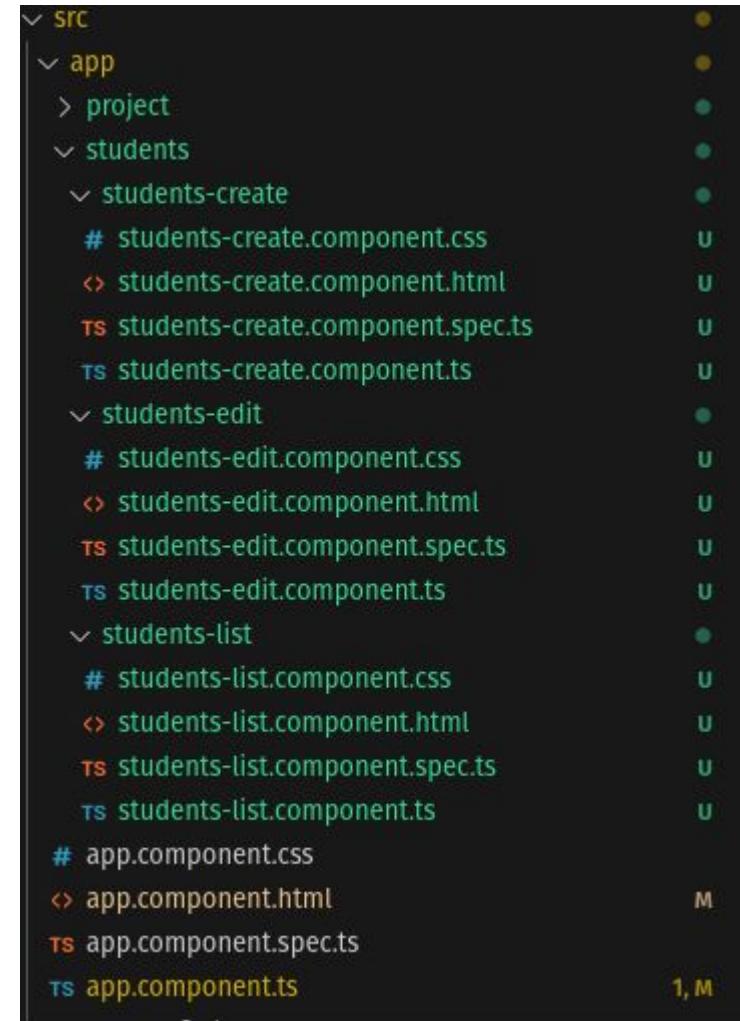
- ❑ `<app-project></app-project>`

Check the homepage now.

```
src > app > app.component.html > app-project
      Go to component
1  <h1>Welcome to Angular</h1>
2  <p>My first Angular app!</p>
3
4  <app-project></app-project>
```

# 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.



A screenshot of a file explorer window showing the directory structure for a component library. The structure is as follows:

- `src`: A folder containing:
  - `app`: A folder containing:
    - `project`
  - `students`: A folder containing:
    - `students-create`: A folder containing:
      - `# students-create.component.css`
      - `<> students-create.component.html`
      - `ts students-create.component.spec.ts`
      - `ts students-create.component.ts`
    - `students-edit`: A folder containing:
      - `# students-edit.component.css`
      - `<> students-edit.component.html`
      - `ts students-edit.component.spec.ts`
      - `ts students-edit.component.ts`
    - `students-list`: A folder containing:
      - `# students-list.component.css`
      - `<> students-list.component.html`
      - `ts students-list.component.spec.ts`
      - `ts students-list.component.ts`
  - `# app.component.css`
  - `<> app.component.html`
  - `ts app.component.spec.ts`
  - `ts app.component.ts`

# 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>

```
src > app > project > project.component.html > button
Go to component
1 <p>project works!</p>
2
3 <button>Click me!</button>
4
```

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

```
src > app > project > ts project.components.ts > ...
1 import { Component } from '@angular/core';
2 import { StudentsCreateComponent } from './students-create.component';
3
4 @Component({
5   selector: 'app-project',
6   imports: [StudentsCreateComponent],
7   templateUrl: './project.component.html',
8   styleUrls: ['./project.component.css']
9 })
10 export class ProjectComponent {
11
12   onClick() {
13     alert("Its working!!!")
14     console.log("Clicked")
15   }
16
17 }
18
```

# 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   name: string = "puskar"
12   isDisabled: boolean = false
13
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    isEnabled: boolean = false  
14    message = '';  
15  
16    onClick() {  
17      alert("Its working!!!")  
18      console.log("CLicked")  
19      this.message = 'Button clicked!';  
20    }  
21  
22  }  
23
```

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

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

Typing in the input updates **name** in the component instantly. **name** changing also re-renders 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  
23
```

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

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

# Directives - \*ngFor

## A “For loop” in html

```
21  <h2>*ngFor Demo</h2>
22  <ul>
23  |  <li *ngFor="let student of students; let i = index">
24  |    {{ i + 1 }}. {{ student }}
25  |  </li>
26 </ul>
27
28 <h2>Student List</h2>
29 <ul>
30 |  <li *ngFor="let student of students">
31 |    {{ student }}
32 |  </li>
33 </ul>
```

```
12  export class ProjectComponent {
13
14  name: string = "puskar"
15  isDisabled: boolean = false
16  message = '';
17  isLoggedIn = false;
18  students = ['Alice', 'Bob', 'Charlie', 'Diana'];
19
20  onClick() {
21    alert("Its working!!!")
22    console.log("CLicked")
23    this.message = 'Button clicked!';
24  }
25
26  toggleLogin() {
27    this.isLoggedIn = !this.isLoggedIn;
28  }
29
30 }
31
```

# 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
];
```

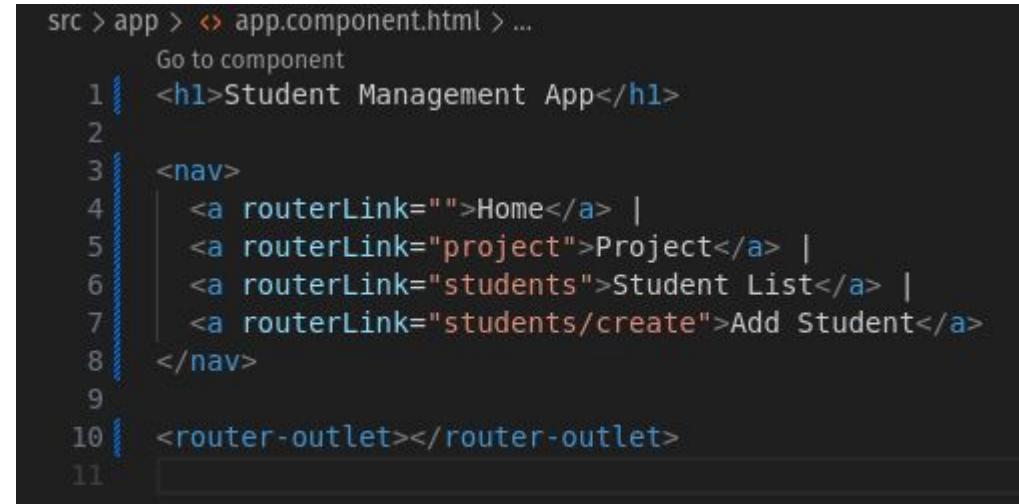
```
6  export const routes: Routes = [
7  {
8    path: 'project',
9    component: ProjectComponent
10 },
11 { path: 'students', component: StudentsListComponent },
12 { path: 'students/create', component: StudentsCreateComponent },
13 { path: '**', redirectTo: '' } // fallback for unknown paths
14 ];
15
```

# 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>
```



```
src > app > app.component.html > ...
Go to component
1 <h1>Student Management App</h1>
2
3 <nav>
4   <a routerLink="">Home</a> |
5   <a routerLink="project">Project</a> |
6   <a routerLink="students">Student List</a> |
7   <a routerLink="students/create">Add Student</a>
8 </nav>
9
10 <router-outlet></router-outlet>
11
```

# 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));
```

```
src > ts main.ts > ...
1  import { bootstrapApplication } from '@angular/platform-
2  import { AppComponent } from './app/app.component';
3  import { provideRouter } from '@angular/router';
4  import { routes } from './app/app.routes';
5
6  bootstrapApplication(AppComponent, {
7    providers: [
8      provideRouter(routes)
9    ]
10 }).catch(err => console.error(err));
11 |
```

# 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.*