Lecture 05 - Getting started with Angular

Pre check

1. Make sure we are on the Node version ^20.9.x by running the following command.

```
node –v
```

2. Check angular version by running the following command

```
ng version
```

3. If it's 17.x.x, then it's fine, otherwise run the following commands.

```
# Uninstall existing version
npm uninstall –g @angular/cli

# Install the specific version
npm install –g @angular/cli@17.3.9
```

1. Create a new project

1. Start a new Project
2. Create a folder where you want to create the new project.
3. Open Terminal or Command prompt and navigate inside the created folder/directory.
4. Run the following command

```
ng new CourseAdminSystemAngular
```

5. Select CSS
6. For SSR, Choose 'N'.
7. The project will be created inside a new folder named **CourseAdminSystemAngular**.
8. Navigate into the new folder by following command.

```
cd CourseAdminSystemAngular
```

9. Run the following commands to fetch the required libraries

```
npm install
```

10. Open Visual Studio Code.
11. Open Folder and browse to the folder named **CourseAdminSystemAngular** and select it.
12. Go to the command prompt/terminal and run the following command.

```
ng serve
```

13. Look for app.component.html and remove everything up to <router-outlet>. Experiment with some html.

2. Create a new Component

1. Open Terminal inside Visual Studio Code.
2. Make sure you are in the root folder of your application.
3. Run the following command to generate a new component e.g. Student

```
ng generate component student
# OR using shorter commands
ng g c student
```

3. View/Edit the newly created files in VS Code.

3. Create a new Class/Interface

1. Create new folder **model** under the **src/app** folder in Visual Studio Code.
2. Make sure you are in the root folder of your application.
3. Run the following command (interface is preferable over class).

```
# Create an interface called Student by running either of the following commands. It will create it under the folder 'model'
ng generate interface model/Student
ng g i model/Student


# If creating a class, use either of the following commands
ng generate class model/Student
ng g cl model/Student
```

4. Add properties to the Student interface as defined in the database model e.g.

```
export interface Student {
    id: number;
    firstName: string;
    lastName: string;
    studyProgram: number;
    dob: Date;
    email: string;
    phone: string;
}
```

4. Configure Component to work with Model

1. Go to *student.component.ts* and create an instance of student like the following.

```
// It's important to fill out all properties if they are not nullables
student?: Student = {
    id: 1,
    firstName: 'Jane',
    lastName: 'Doe',
```

```
    studyProgram: 1,
    dob: new Date(2000, 1, 1),
    email: 'jane.doe@mailinator.com',
    phone: '+4511111111'
}
```

2. Go to *student.component.html* and display all student properties as required. Experiment with html tags. An example is as follows

```
@if (student) {
    <div>
     <b>id:</b><span>{{student.id}}</span>
    </div>
    <div>
     <b>First name:</b><span>{{student.firstName}}</span>
    </div>
    <div>
     <b>Last name:</b><span>{{student.lastName}}</span>
    </div>
    ...
    ...
}
@else {
    <p>Nothing to display</p>
}
```

3. Update *app.component.ts* to include *StudentComponent* to the *imports* list as follows.

```
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, StudentComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
```

4. Update *app.component.html* so that it can display the student component by using the following code.

```
<app-student></app-student>
```

5. [Exercise] Create a Teacher component

Create a Teacher component following the previous steps.
Use the properties defined for the Teacher previously when creating it in the Postgres database.

6. Create a StudentList component

Now we add a new component which would display multiple students.

1. Run the following command (make sure we are at the **src/app** directory).

```
ng generate component StudentList
```

2. Go to *student-list.component.ts* and create an array of students and populate it with a few records as follows.

```
students: Student[] = [
    {
        id: 1,
        firstName: 'Jane',
        lastName: 'Doe',
        studyProgram: 1,
        dob: new Date(2000, 1, 1),
        email: 'jane.doe@mailinator.com',
        phone: '+4511111111'
    },
    {
        id: 2,
        firstName: "Super",
        lastName: "Man",
        studyProgram: 2,
        dob: new Date(2002, 2, 28),
        email: "super.man@mailinator.com",
        phone: "+4522222222"
    },
    {
        id: 3,
        firstName: "Super",
        lastName: "Woman",
        studyProgram: 1,
        dob: new Date(2001, 7, 1),
        email: "super.woman@mailinator.com",
        phone: "+4533333333"
    }
];
```

3. Go to *student-list.component.html* and display all the students in the array using a for loop as in the example below

```
@for (student of students; track $index) {
    <div>
     <b>id:</b><span>{{student.id}}</span>
```

```
    </div>
    <div>
     <b>First name:</b><span>{{student.firstName}}</span>
    </div>
    <div>
     <b>Last name:</b><span>{{student.lastName}}</span>
    </div>
    ...
    ...
}
```

4. Update the *app.component.html* to display the StudentListComponent by using the following code.

```
<app-student-list></app-student-list>
```

5. Check the updated app in the browser to verify everything is working fine.

7. Refactor StudentList to use Student component

1. Go to the *student.component.ts* and modify the student property as follows.

```
@Input() student?: Student;
```

2. Go to the *student-list.component.html* and update the code as follows.

```
@for (student of students; track $index) {
    <app-student [student]="student"></app-student>
}
```

3. Verify that everything works as expected.
4. Experiment with modifying the *student.component.html* to display student data in some other way, e.g. as a table.

```
<table>
    <tr>
        <td>Id</td>
        <td>{{student.id}}</td>
    </tr>
    <tr>
        <td>First name</td>
        <td>{{student.firstName}}</td>
    </tr>
    <tr>
        <td>Last name</td>
        <td>{{student.lastName}}</td>
    </tr>
```

```
    ...
    ...
</table>
```

5. Prettify the table by adding some css to *student.component.css* e.g. as follows.

```
table, th, td {
   border: 1px solid black;
}
```

6. Add an option to toggle between the two views by modifying the code as follows in the *student.component.ts*.

```
mode = 0; // 0 will display div mode, 1 will display table mode
```

7. In *student.component.html*, wrap the views in an if/else statement as follows

```
@if (mode == 0) {
    <div>
        <b>id:</b><span>{{student.id}}</span>
    </div>
    <div>
        <b>First name:</b><span>{{student.firstName}}</span>
    </div>
    <div>
        <b>Last name:</b><span>{{student.lastName}}</span>
    </div>
    ...
    ...
}
@else {
    <table>
    <tr>
        <td>Id</td>
        <td>{{student.id}}</td>
    </tr>
    <tr>
        <td>First name</td>
        <td>{{student.firstName}}</td>
    </tr>
    <tr>
        <td>last name</td>
        <td>{{student.firstName}}</td>
    </tr>
    ...
```

```
    ...
</table>
}
```

8. [Exercise] Create a TeacherList component

Create a TeacherList component following the previous step (6).

9. Basic Routing

When we have multiple components added, we need a way to move between them. This can be achieved by routing. A very simple routing can be achieved by the following steps.

1. Go to *app.routes.ts* and update the code as follows.

```
export const routes: Routes = [
   { path: 'students', component: StudentListComponent },
];
```

2. Go to the *app.component.ts* and update the *imports* array as follows.

```
imports: [.., "RouterLink", ..]
```

3. Go to the *app.component.html* and add a link which routes to the StudentList component when clicked. This can be done as follows.

```
<ul>
  <li>
    <a routerLink="/students">Student list</a>
  </li>
</ul>
```

10. [Exercise] Create a route to the TeacherList component

1. Add a new route to the previously added TeacherList component.
2. Try adding a few other components e.g. a Home component which just displays a welcome text.
3. Update the routes with the newly added components.

Some useful commands using terminal/command prompt

```
NOTE: Everything including and after the # is a comment and is NOT a part of the command
and hence NOT to be used


# Move to a sub-folder within the current folder
cd name_of_the_folder


# Move to the parent folder of the current folder
cd ..
```

```
# View the contents of the current folder
ls # For Mac
dir # For Windows



# If you are getting some permission errors, run the following commands

# Windows: If scripts are not allowed to run
set-ExecutionPolicy RemoteSigned -Scope CurrentUser

# Mac: Make folder writeable for all users
sudo chmod -R 777 <project_dir_name>
```