Lab 10 – Angular via local device

Task 1: Creating an Angular project (5 marks)

To do so:

1. Launch NodeJS command line terminal from the Windows Start Menu "Node.js->Node.js command prompt" (or search "Node.js command prompt")

   Note: Angular requires a current, Active LTS (long time support) or maintenance version of Node.js (https://nodejs.org/en/download/). **The version of Node.js installed on AUT computers may not be latest so we strongly recommend you install it on your own machine.**
2. Change to `C:` drive in the terminal by typing: `C:`
3. Go to the "Desktop" directory by typing:

   ```
   cd C:\Users\[your user name]\Desktop
   ```
4. Do this **only if you are using an AUT computer**:
   Set up the proxy:
   ```
   npm config set proxy http://cache.aut.ac.nz:3128
   ```
5. Install Angular CLI globally:
   ```
   npm install -g @angular/cli
   ```

   You can also install specific version of CLI, e.g., install version 11.2.8 using the command: `npm install -g @angular/cli@11.2.8`
   It is recommended that you install latest version of angular CLI.

6. Create a folder called *'AngularLab'* and create a new angular project name *courses-info-app* with the angular CLI command in this folder:
   - `mkdir AngularLab`
   - `cd AngularLab`
   - `%AppData%/npm/ng new course-info-app` (Run `ng new course-info-app` instead if you are using your PC or using the node.js command prompt).
   - The '`ng new`' command prompts you for information about the features to include in the initial app. Accept the defaults by pressing **Enter** or **Return key**.
   - The Angular CLI installs the necessary Angular `npm packages` and other dependencies. Please wait until this process is finished.
   - Now, an Angular root module containing a default component (the name is `app`) with a default HTML template.
   - It also creates the following workspace and starter project files:
     - A new workspace, with a root folder named course-info-app.
     - An initial skeleton app project (in the src subfolder).
     - An end-to-end test project (in the e2e subfolder).
     - Related configuration files.

Step 2: Angular feature: Interpolation Binding.

- From the project directory, launch the app from the server (Angular CLI includes a server, so you can build and serve you app locally):
  - `cd course-info-app`
  - `ng serve` (Run `%AppData%/npm/ng serve` instead if you are using Windows Command prompt)
- Use chrome to visit http://localhost:4200/.
- The page you will see is the application shell. The shell is controlled by an Angular component named ***AppComponent.*** Components are the fundamental building blocks of Angular applications. They display data on the screen, listen for user input, and act based on that input.

- *Note: as IE is not working property, please use the Chrome browser.*

- Study the files generated in the project created using an editor.

- Focus on the following in `\AngularLab\course-info-app\src\app:`
  - ***app.components.ts*** (the component class code, written in type script),
  - ***app.component.html*** (the component template, written in html)
  - ***app.component.css*** (the component's private CSS styles) ***and***
  - ***app.module.ts***

- Change the title of app in class file (app.component.ts) *to* ***My Courses List with Info.***
  - `title = 'My Courses List with Info';`
- Replace everything in ***app.component.html*** with:
- `<h1> {{ title }} </h1>`
  - Note that the double curly braces are Angular's *interpolation binding* syntax.
  - This interpolation binding presents the component's title property value inside the HTML header tag.
  - The browser refreshes and displays the new application title.

Step 3: Creating a new component and add it to the default component.

- Create the courses component using Angular CLI command below:

```
ng generate component courses
```

NB: You need to open a new terminal and navigate to `<User\Desktop\AngularLab\course-info-app>`, then run the command above.

- In the ***courses.component.ts*** class file, study the three-metadata properties generated *(Selector, templateUrl and styleUrls).* Note that the selector '`app-courses`' matches the name of the HTML element that identifies this component within a parent component's template.
- Add a course property to the CourseComponent for a course named "Web Development."

```
course='Web Development';
```

- Replace everything in ***courses.component.html*** with:

```
<h2> {{course}} </h2>
```

- Append `<app-courses></app-courses>` to `app.component.html`.
- Check your browser for your course property display.

Step 4: Creating a new class.

- Create a `Course` class in a separate ***.ts*** file to capture all your courses with attributes. Put the file in src/app directory.
- Part of the file has been provided below, but you need to complete the file by giving each class attribute a correct type. Please refer to the webpage https://www.typescriptlang.org/docs/handbook/basic-types.html
- And fill in the attribute types.

```
export class Course {
  course_id!: _____;
  course_title!: _____;
  semester!: _____;
  period!:  _____;
  lecturer!: _____;
}
```

- Import this class in course component and change the type of course in the component to this class type.
  Note: The page won't display properly before you finish changing the course from a string to an object.
- Replace the content of your course.component.html with the following:

```
<h2>{{ course.course_title }} Details</h2>
<div><span>Course ID: </span>{{ course.course_id }}</div>
<div><span>Course Title: </span>{{ course.course_title }}</div>
<div><span>Semester: </span>{{ course.semester }}</div>
<div><span>Course Lecturer: </span>{{ course.lecturer }}</div>
<div><span>Course Period: </span>{{ course.period }}</div>
```

- Test in the browser to see what your app looks like.

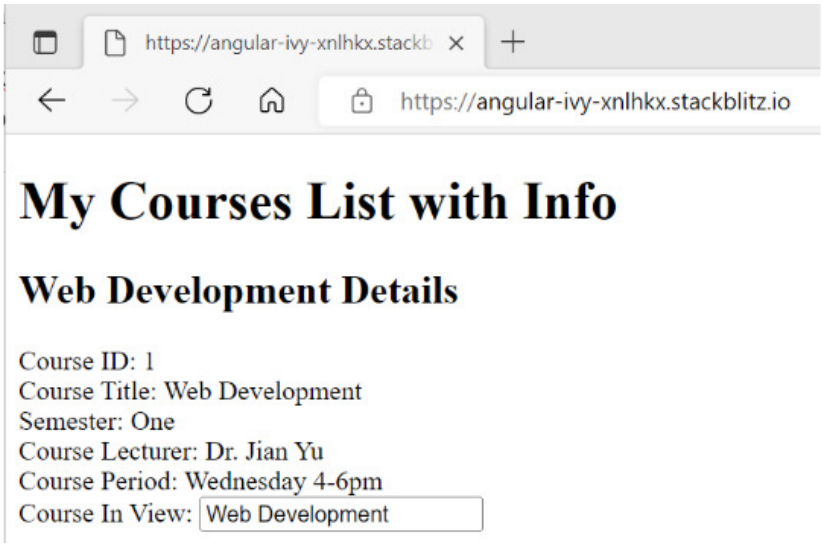
Step 5: Show the course object information in the template.

- Include two-way binding by appending:

```
<div><label>Course In View:
<input [(ngModel)]="course.course_title" placeholder="name" /></label></div>
```

  Note: For this to work you must make sure FormsModule is included in NgModule.

- Open AppModule (***app.module.ts***) and **import the FormsModule** from the @angular/forms library. Your output should look like the following screenshot:



Task 2: Displaying List of Courses (3 marks)

Step 1: Use *ngFor to populate the template.

- Create a file called ***test-course.ts*** in the *src/app/* folder. Define a `COURSES` constant as an array of 5 courses and export it. The file should look like this:

```
import {Course} from './courses';
export const COURSES: Course[] = [
  // ----insert array of five courses with ID here-----
];
```

- Import the class `test-course` into the `CourseComponent` class file and add a courses property to the class that exposes these courses for binding:

```
courses= COURSES;
```

- List your courses with *ngFor as follows:

```
<h2>My Course List</h2>
<ul class="courses">
  <li *ngFor="let course of courses">
    <span class="badge">{{course.course_id}}</span> {{course.course_title}}
  </li>
</ul>
```

- Fill in the empty {{}} to show the course id and title:

Step 2: Add CSS

Add the CSS style below to the ***course.component.css*** file:

```
.selected {
  background-color: #cfd8dc !important;
  color: white;
}
.courses {
  margin: 0 0 2em 0;
  list-style-type: none;
  padding: 0;
  width: 15em;
}
.courses li {
  cursor: pointer;
  position: relative;
  left: 0;
  background-color: #eee;
  margin: 0.5em;
  padding: 0.3em 0;
  height: 1.6em;
  border-radius: 4px;
}
.courses li.selected:hover {
  background-color: #bbd8dc !important;
  color: white;
}
.courses li:hover {
  color: #607d8b;
  background-color: #ddd;
  left: 0.1em;
}
```

```
.courses .text {
  position: relative;
  top: -3px;
}
.courses .badge {
  display: inline-block;
  font-size: small;
  color: white;
  padding: 0.8em 0.7em 0 0.7em;
  background-color: #607d8b;
  line-height: 1em;
  position: relative;
  left: -1px;
  top: -4px;
  height: 1.8em;
  margin-right: 0.8em;
  border-radius: 4px 0 0 4px;
}
```

Task 3: Add Click Event and Event Handler (2 marks)

Step 1:

- Add a click event binding to the <li> like the following:

  ```
  <li *ngFor="let course of courses" (click) = "onSelect(course)">
  ```

- Create an instance of `Course` named `selectedCourse` and add it to `CoursesComponent` class.

  ```
  selectedCourse!:_______;
  ```

- Create the `onSelect()` method, which assigns the clicked course from the template to the component's `selectedCourse`. Add the method to `CoursesComponent` class.

  ```
  onSelect(course: _______): void { this.selectedCourse = ______; }
  ```

Step 2:

- Update the component.html with the new inclusion:

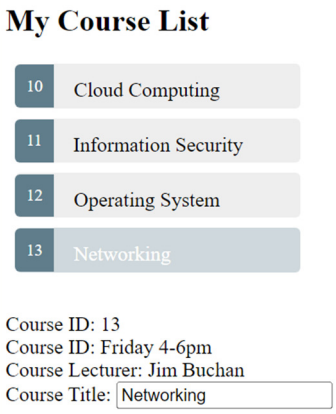
```
<div *ngIf="selectedCourse">
  <div><span>Course ID: </span>{{ selectedCourse.course_id }}</div>
  <div><span>Course ID: </span>{{ selectedCourse.period }}</div>
  <div><span>Course Lecturer: </span>{{ selectedCourse.lecturer }}</div>
  <div>
    <label>
      Course Title:
      <input [(ngModel)]="selectedCourse.course_title" placeholder="title" /
>
    </label>
  </div>
</div>
```

Note: the HTML enclosed in `*ngIf="selectedCourse"`. This implies the component should only display the selected course details if the `selectedCourse` exists.

- Append `[class.selected]="course === selectedCourse"` to the ***course.component.html*** to highlight the selected course.

```
<li
  *ngFor="let course of courses"
  (click)="onSelect(course)"
  [class.selected]="course === selectedCourse"
 >
```

The highlight effect looks like this:



Task 4: Create another component to separate the Details of Course (0 marks)

Step 1:

- Generate another component called `course-detail`. Cut the HTML template for this component from the ***course.component.html*** .i.e. the selected Course html:

```
<div *ngIf="selectedCourse">
  <div><span>Course ID: </span>{{ selectedCourse.course_id }}</div>
  <div><span>Course ID: </span>{{ selectedCourse.period }}</div>
  <div><span>Course Lecturer: </span>{{ selectedCourse.lecturer }}</div>
  <div>
    <label
      >Course Title:
      <input [(ngModel)]="selectedCourse.course_title" placeholder="title" /
>
    </label>
  </div>
</div>
```

Step2:

- Import Course class into `course-detail` and add the @Input ( ) decorator to bind course property with this component:

```
import { Course } from '../course';
import { Component, OnInit, Input } from '@angular/core';
```

- Append the line below to the ***course.component.html***

```
<app-course-detail [course] = "selectedCourse"></app-course-detail>
```

- Add a course property, preceded by the @Input ( ) decorator:

```
@Input() course: Course;
```

Extra Challenge:

Extend your application by retrieving and updating data on MongoDB.