Lab 10 – Angular via StackBlitz

Task 1: Creating an Angular project (5 marks)

Step 1: Create a new Angular project via StackBlitz.

- Visit https://stackblitz.com/.
- Login with your GitHub account (so you can submit your lab work via share).
- Create a new Angular project by clicking Angular TypeScript. It will direct to a web IDE where you can find the project workspace under "FILES" section in the left part of the page. The "app" folder under "src" folder contains initial skeleton files for this project. Focusing on the following files:
  - ***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)
  - ***app.module.ts***

Step 2: Angular feature: Interpolation Binding.

- The right part of the page 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.
- The middle part of page is an editor. Change the name of app in class file (app.component.ts) to "My Courses List with Info":

```
name = 'My Courses List with Info';
```

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

```
<h1>{{ name }}</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.

- Right click the "app" folder and hover over "Angular Generator", then click "Component". Name the new component "courses".
- 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 `CoursesComponent` class for a course named "Web Development.":

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

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

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

- Import and declare the courses component in ***app.module.ts***:
  - Add the following line above @NgModule:

    ```
    import { CoursesComponent } from './courses/courses.component';
    ```

  - Add declaration in @NgModule:

    ```
    declarations: [AppComponent, HelloComponent, CoursesComponent],
    ```

- 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>
```

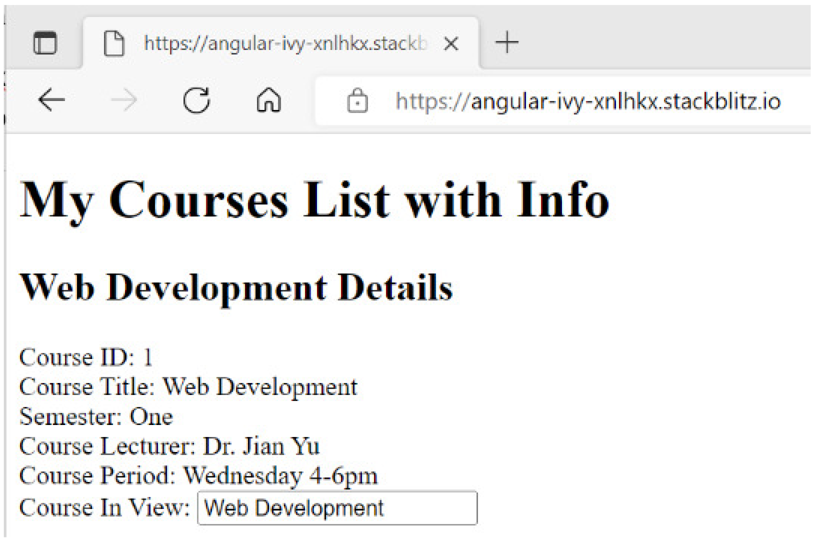
- Test in the browser to see what your app looks like.

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:

Web Development


https://angular-ivy-xnlhkx.stackb
https://angular-ivy-xnlhkx.stackblitz.io

My Courses List with Info

Web Development Details

Course ID: 1
Course Title: Web Development
Semester: One
Course Lecturer: Dr. Jian Yu
Course Period: Wednesday 4-6pm
Course In View: Web Development

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-----
];
```

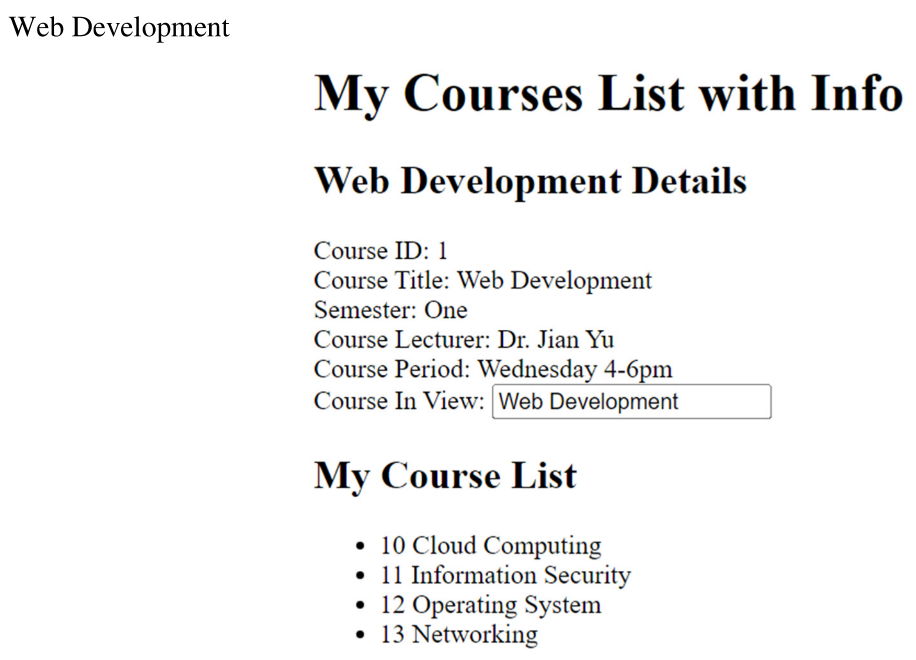
- Import the class `test-course` into the `CourseComponent` class file and add a courses property to the class that exposes these courses for binding:

```
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 like the following screenshot:


Web Development

My Courses List with Info

Web Development Details

Course ID: 1
Course Title: Web Development
Semester: One
Course Lecturer: Dr. Jian Yu
Course Period: Wednesday 4-6pm
Course In View: Web Development

My Course List

- 10 Cloud Computing
- 11 Information Security
- 12 Operating System
- 13 Networking

Step 2: Add CSS

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

```css
.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:

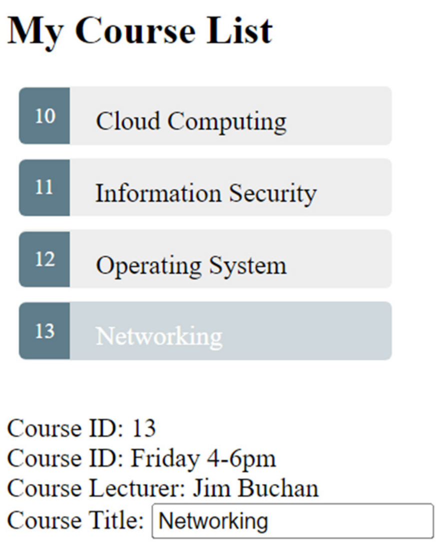
```
<div *ngIf="selectedCourse">
  <div><span>Course ID: </span>{{ selectedCourse.course_id }}</div>
  <div><span>Course ID: </span>{{ selectedCourse.period }}</div>
  <div><span>Course Lecturer: </span>{{ selectedCourse.lecturer }}</div>
  <div>
    <label>
      Course Title:
      <input [(ngModel)]="selectedCourse.course_title"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



My Course List

Course ID: 13
Course ID: Friday 4-6pm
Course Lecturer: Jim Buchan
Course Title: Networking

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.