Seneca | FACULTY OF CONTINUING EDUCATION AND TRAINING

SENECA WEB PROGRAMMING PROGRAM
SUMMER 2019 SESSION

Module Number: WEB 304
Assignment #4

Instructor: Tim Lai
Title: Angular Application
Assigned: Wednesday July 24
Draft Presentation Date: 6:30pm to 9:30pm Wednesday August 7
Draft Presentation Value: 5% of 40%
Final Due Date: 11:59pm Wednesday August 14
Final Value: 35% of 40%

Submission details: This assignment MUST be submitted in at least one of 2 ways, as specified below. If the assignment does not follow the specified instructions, it will be considered incomplete. *Failure to upload this assignment will result in a grade of zero for this assignment*.

1. This assignment should be uploaded to a **GitHub repository** which I will invite you to. When I download this GitHub repository, I must be able to access all your source files so that I can see your, TypeScript, JavaScript, JSON, HTML and CSS code and test your application. *DO NOT UPLOAD YOUR node_modules DIRECTORY BUT DO INCLUDE YOUR package.json file.*
2. If you have difficulty uploading to GitHub, then you may alternatively upload your assignment as a ZIP file to an online cloud storage site such as **Google Drive** or **WeTransfer** and send me a link at **tim.lai@senecacollege.ca**. *DO NOT EMAIL ME A ZIP FILE AS AN ATTACHMENT. I WILL NOT RECEIVE IT. DO NOT UPLOAD YOUR node_modules DIRECTORY BUT DO INCLUDE YOUR package.json file.*

This assignment has 2 submission points which correspond with the dates indicated above:

1. **Draft Presentation:** The first submission will be a rough draft of the entire assignment. The goal of this draft is to demonstrate that you understand the requirements and broad concepts of the assignment. It is not necessary for every part of the assignment to work and there will be no deductions for errors. You will present your draft to the class, explaining how your code works and what the purpose of it is. You will provide feedback on the draft assignments of your classmates and accept feedback on your own.
2. **Final:** The second submission will be the completed version of the assignment. This version should be fully-functional and there will be deductions for any errors as indicated in the rubric.

Instructions: You will be creating a **component-based MVC UI application** using **Angular 8** which allows users to view a list of **objects** which can belong to a **topic of your choice** (i.e. Spaceship, Dog, Clown etc.) *as long as it is NOT a Hero – be creative*. This application will store the objects in either a **TypeScript array** or **JSON file** which is loaded using either **Promises** or **RxJS Observables**. *DO NOT USE ONLY JAVASCRIPT OR A DIFFERENT JAVASCRIPT FRAMEWORK SUCH AS JQUERY, REACT, VUE, EMBER OR ESPECIALLY ANGULARJS*.

Your application should use **at least 1 model** in the form of a **TypeScript interface** which defines at least **5 properties** – an **id** which is a **number**, a **name** which is a **string**, an **image** which is a **string**, some **type of array** and a **Boolean**. Data types should be explicitly defined for every property in the model interface.

Your application should use **at least 1 service**. The Service should be used to transfer data from the model to the view using distinct **getObjects**, and **getObject** scripts to collect the object data from the model file and filter the data to display either all of the objects or a single object. You may use either **Promises** or **Observables** for AJAX functionality.

Your application should use **at least 3 components** – an app component, a component which displays all your objects and a component which displays a single object. The components should define methods to get either all the objects or a single object using the service as well as any other methods which are necessary for each component to work. Every **HTML** and **CSS** file for each component should include **at least some code**. You should demonstrate basic competency with HTML and CSS as well as an ability to style **at least 1 custom element**. Despite this, your application will not include entirely valid HTML code as a result of Angular's custom elements and attributes. *However, you should still try to use valid code, or you may lose some marks*.

The application should start on a page which displays the **name** of each of the objects which have been retrieved from the JSON file. The user should also be able to view an **image** which is associated with each object. There should be a distinct **link or button** associated with each object which displays a single object when clicked. When the user views a single object, they should see more information about that object, including a list of the data stored in the object's array property. They should also be able to use an **input** to alter the name of the object using the **@Input decorator** and the **FormsModule**. The user should also be able to navigate back to the previous page from this page. There should also be a router file which lists **at least 3 routes** using the **RouterModule** – a root directory redirect, a route to display all the objects and a route to display a single object.

The application should have an intuitive and functional **user interface**. Nothing complicated or fancy is required but the user interface should be easy to navigate and use. It should make use of **buttons**, and **colour-coding**. Basic accessibility considerations should be made for non-standard users (i.e. alt text on <img> tags). Responsiveness and use of libraries such as Bootstrap, Foundation and Font Awesome is NOT required but is strongly encouraged.

Deliverables:

- A model defined as a **interface** with at least **5 properties** including an **id**, a **name**, an **image**, an **array of some type** and a **boolean**
- A **TypeScript array** or **JSON** file which stores all this data, and which follows valid JSON formatting rules (if using JSON)
- A **Service** with distinct **getObjects** and **getObject** scripts which makes use of either **Promises** or **RxJS Observables**

- At least 3 components – an app component, a component which displays all your objects and a component which displays a single object
- At least 3 routes: a route to an **All Objects** page a route to a **Single Object** page and a root directory redirect (you should rename the pages to match your class i.e. All Dogs)
- A **button or link** which is associated with each object on the All Objects page which displays a single object when clicked
- An **input** which alters the name of the object using the **@Input decorator** and **FormsModule**
- CSS code for each component with at least 1 custom element styled
- A functional and intuitive **UI/UX design** which makes use of **buttons** and **colour-coding** (use of Bootstrap/Foundation/Font Awesome is encouraged)
- Accessibility considerations for non-standard users (blind, assistive devices, etc.)

Grading breakdown:

- Presentation of first draft **(5%)**
- Creation of data and a model interface with properties and type checking **(10%)**
- Creation of a service which retrieves objects using Promises or Observables **(20%)**
- Creation of a components which transfer data and methods to templates **(25%)**
- Creation of templates which display objects using HTML and CSS **(20%)**
- Creation of routes which load components **(10%)**
- Good coding/file organization practices (property names, bracket indentation etc.) **(10%)**

Your files will not use entirely valid HTML code as a result of Angular's custom elements and attributes. Therefore, you do not need to validate your HTML code and will not lose 2% per type of validation error. *However, you should still try to use valid code, or you may lose some marks*. Under most normal circumstances, you should not have any TypeScript compilation errors. *You will lose 2% per type of TypeScript compilation error which could have been fixed*. Under normal circumstances, you should not have any JavaScript errors in your console. *You will lose 2% per type of JavaScript console error which could have been fixed*. You MUST have valid JSON code according to JSONLint (https://jsonlint.com/). *You will lose 2% per JSON error*.

Late Policy

All late assignments will be given a grade of zero.

Plagiarism

There are serious penalties for cheating and plagiarism offences and you are expected to be aware of our Academic Honesty Policy. Please refer to the Academic Policy at http://www.senecacollege.ca/ academic-policy/acpol-09.html for more information.