

USER INTERFACE DESIGN

Laying out and styling responsive Webpages with HTML, CSS, and Bootstrap

Introduction

This is the first of several related assignments that build on each other to create an online course management system where faculty can author courses and students can register for online tutorials. In this assignment HTML, CSS, and Bootstrap are used to create a responsive, static Web site that will serve as the prototype for a dynamic Web application.

Several examples are provided throughout the assignment. All examples are illustrative and not meant to be copied and pasted verbatim for your own solution. You are responsible for using the code snippet examples for what they are, just examples. Do not expect the examples to work as is. You are meant to write all your code from scratch, using the code snippets provided here and in class as guidance. Ask your instructors for further clarification if needed.

The assignment requires the use of various naming conventions and other common best practices that you are required to employ throughout all assignments. If you are familiar with alternative conventions and industry standards you might have acquired from co-ops or your professional practice, feel free to use those if you feel they are an improvement over the ones suggested here. Confirm with your instructor(s) and TA(s) as it may affect your assignment score.

The wireframes, styles, and look and feel used through the assignment are a minimum requirement and are only illustrative suggestions. You are free to improve on the styling, layout, and look and feel, as long as it is a clear improvement over the illustrations provided. Equivalent use of white space, justification, wrapping, padding, and margins is required. All pages must be responsive and adapt to the size of the browser. Confirm with your instructor(s) and TA(s) if you have doubts.

Features

The following features will be implemented in this assignment:

- (20pts) Home page and course list
- (10pts) Registration allows anyone to register as a user
- (10pts) Login allows anyone to login
- (20pts) Course list
- (40pts) Course editor

Learning objectives

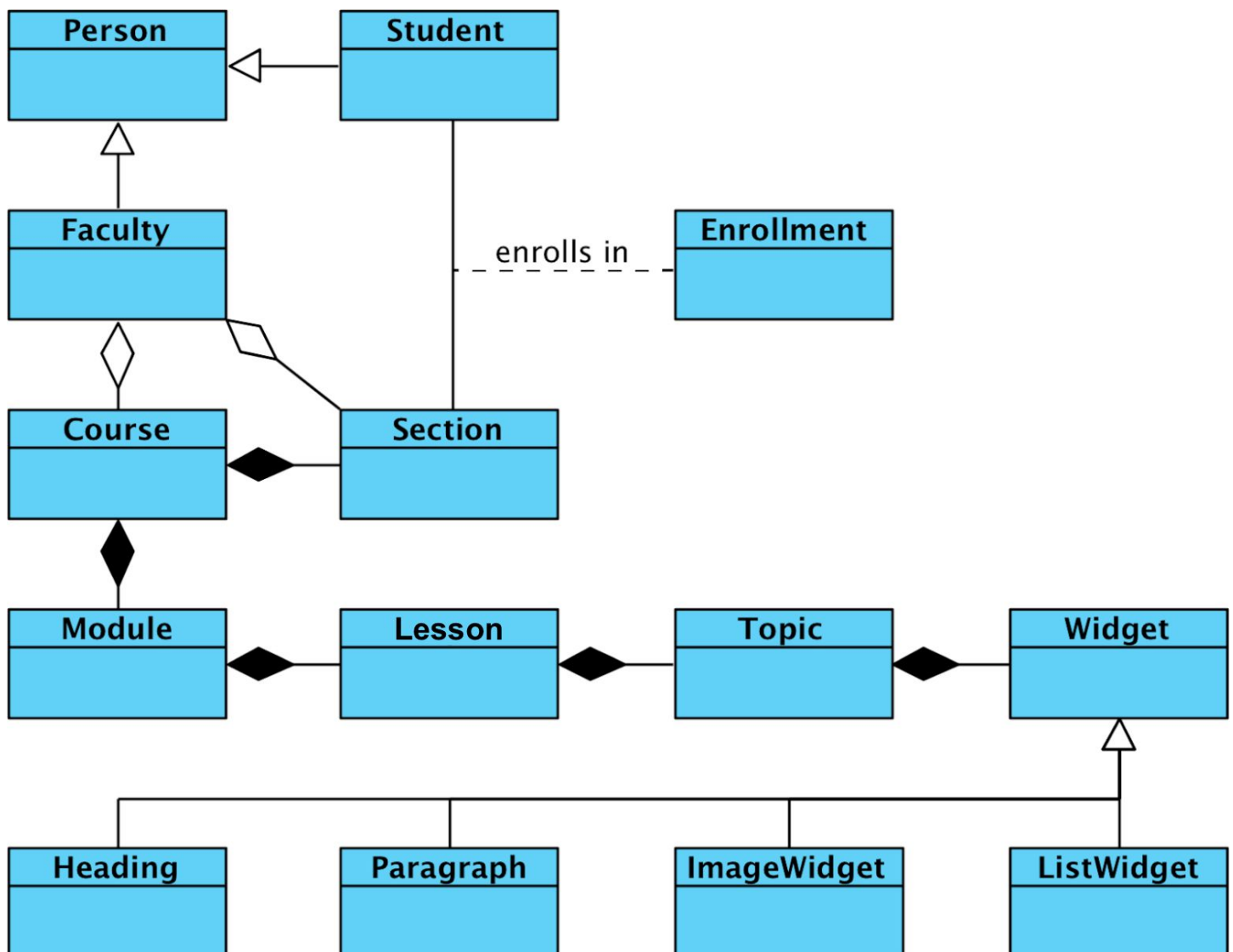
This assignment will introduce the student to the following concepts and skills:

- Creating webpages with the Hypertext Markup Language (HTML)

- Styling HTML webpages with Cascading Style Sheet (CSS) library
- Laying out responsive webpages with the Bootstrap CSS library
- Implementing a Java HTTP Web server using Spring boot
- Data modeling with the Unified Modeling Language (UML)

Design

The UML class diagram below shows the relation between several entities implemented in this assignment. Entities describe related attributes of an object or concept that we wish to store in a database. The base class Person captures common attributes describing all users of the system. The Person class contains attributes such as id (which uniquely identifies a Person's record), username, password, firstName, lastName, dateOfBirth, role, email, and phone. Faculty and Student classes inherit attributes from Person, capturing additional, more specific, attributes of faculty and students. We say that Person is a more general form, or a generalization of Faculty and Student. The inverse is also true, that is, faculty and student are special cases of a person, or they are specializations. Student class captures attributes such as gpa and graduation year. Faculty captures attributes such as whether the faculty is tenured, and their office.



The class diagram further captures the one to many relationship “authors” between faculty and course. That is, one faculty can author many courses. Similarly, there’s a one to many relationship teaches between faculty and section, describing the fact that one faculty can teach several sections. There’s also a one to many relation between courses and sections, that is, one course may have many sections associated with it. And finally, enrollment describes a many to many relation between sections and students, capturing the fact that a student might be enrolled in many sections and that many sections might have many students enrolled in it. The enrollment association can further describe the relation between a particular student and a particular section such as the grade the student received when enrolled in that section.

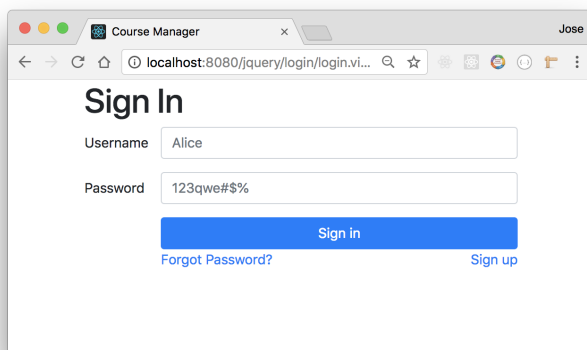
Wireframes

Wireframes capture the content and layout intent of a visual user interface. The styling shown here are only for illustration purposes. You are free to improve on the styling suggested here, or implement the proposed style.

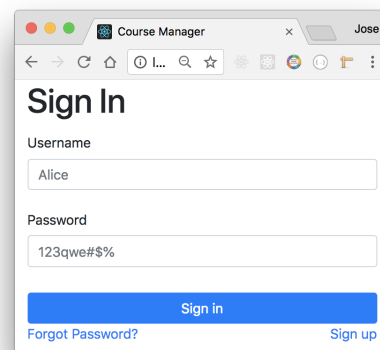
Use CSS libraries such as bootstrap and fontawesome to provide a consistent look and feel across the application. These libraries provide generic example code snippets that can be copied and tailored for a specific use. Although reusing code from well known authoritative sources is encouraged, avoid copying whole entire pages or algorithms. Make sure to provide clear references.

Login requirements

The login page allow users to login into their personal profiles. Clicking on the sign in button navigates the user to their profile page. The login page, as many other pages must be responsive to the size of the screen. The images below illustrate the login page as rendered in a desktop and in a smaller mobile screen. You are free to make improvements on the design and styling as long as the basic functionality is supported.



Desktop login wireframe



Mobile login wireframe

Clicking on the Sign up link navigates to the registration page. The forgot password feature is not required for this assignment. Clicking on cancel (not shown) navigates to the home page. Implement the user login page in the following files in the **webapp/login** directory.

File	Description
------	-------------

login.template.client.html	implements the user interface using HTML
login.style.client.css	implements the styling of the login page using CSS

Use the login template below as a starting point. The template renders a sign in title, username and password labels and inputs, a sign in button, and links to forgot password and sign up. The sign up link navigates to the sign up page described elsewhere. Here's a snippet of code to get you started on the login page. Note: all code provided is for illustration purposes only, you are responsible for the actual final code submitted for any assignment

```
webapp/login/login.template.client.html
```

```
<div class="container">
  <h1>Sign In</h1>
  <form>
    <div class="form-group row">
      <label for="username" class="col-sm-2 col-form-label">
        Username </label>
      <div class="col-sm-10">
        <input class="form-control wbdv-field wbdv-username"
          id="username"
          placeholder="Alice">
      </div>
    </div>
    <div class="form-group row">
      <label for="password" class="col-sm-2 col-form-label">
        Password </label>
      <div class="col-sm-10">
        <input type="password" class="form-control wbdv-field wbdv-password"
          id="password" placeholder="123qwe#$%">
      </div>
    </div>
    <div class="form-group row">
      <label class="col-sm-2 col-form-label"></label>
      <div class="col-sm-10">
        <a href="profile.template.client.html" class="btn btn-primary btn-block wbdv-login">Sign in</a>
        <div class="row">
          <div class="col-6">
            <a href="#">Forgot Password?</a>
          </div>
          <div class="col-6">
            <a href="#" class="float-right">Sign up</a>
          </div>
        </div>
      </div>
    </div>
  </form>
</div>
```

Implement styling specific to the login page in a CSS file called `login.style.client.css` and link to it from the template file.

Commit your work into source control

As you complete a significant amount of work, and reach a stable state of your project, commit your work into source control. Add and commit the files created in this section. Use a commit command and message similar to the one below.

Git commit command and message

```
git add .
git commit -m 'Created login web page'
```

Do the same for every section that follows where a new file is created, or new functionality is added.

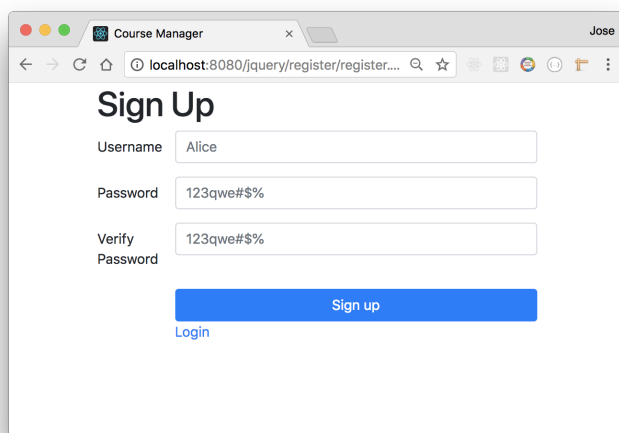
User signup requirements

The user signup feature allow users to register so that they can login into their personal profiles. The user signup front end will provide a form for users to enter their preferred username and password. Implement the feature in the following files in the `webapp/register` folder.

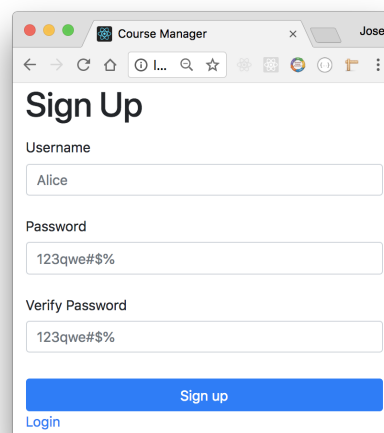
File	Description
<code>register.template.client.html</code>	implements the user interface using HTML
<code>register.style.client.css</code>	implements the styling of the user admin page using CSS

User signup wireframes

Users can sign up using the registration page show below. The page is responsive having a desktop and a mobile version.



A desktop browser window titled 'Course Manager' showing a 'Sign Up' form. The form has three input fields: 'Username' with the value 'Alice', 'Password' with the value '123qwe#\$', and 'Verify Password' with the value '123qwe#\$', each with a label to its left. Below the fields is a blue 'Sign up' button and a blue 'Login' link.



A mobile browser window titled 'Course Manager' showing a 'Sign Up' form. The form has three input fields: 'Username' with the value 'Alice', 'Password' with the value '123qwe#\$', and 'Verify Password' with the value '123qwe#\$', each with a label above it. Below the fields is a blue 'Sign up' button and a blue 'Login' link.

Clicking on the Login link navigates to the login page. Clicking on the Sign up button, navigates to the profile page, with the newly registered user currently logged in. Clicking on cancel (not shown) navigates back to home. Below is a snippet of code for to get you started. Add additional markup and styling as shown in the login template. Note: all code provided is for illustration purposes only, you are responsible for the actual final code submitted for any assignment

```
webapp/register/register.template.client.html
```

```
<h1>Register</h1>
Username: <input id="usernameFld"/>
Password: <input id="passwordFld"/>
Verify Password: <input id="verifyPasswordFld"/>
<button id="registerBtn">Register</button>
```

Implement any register specific styling in a file called `register.style.client.css` in the same folder.

Commit your work into source control

As you complete a significant amount of work, and reach a stable state of your project, commit your work into source control. Add and commit the files created in this section. Use a commit command and message similar to the one below.

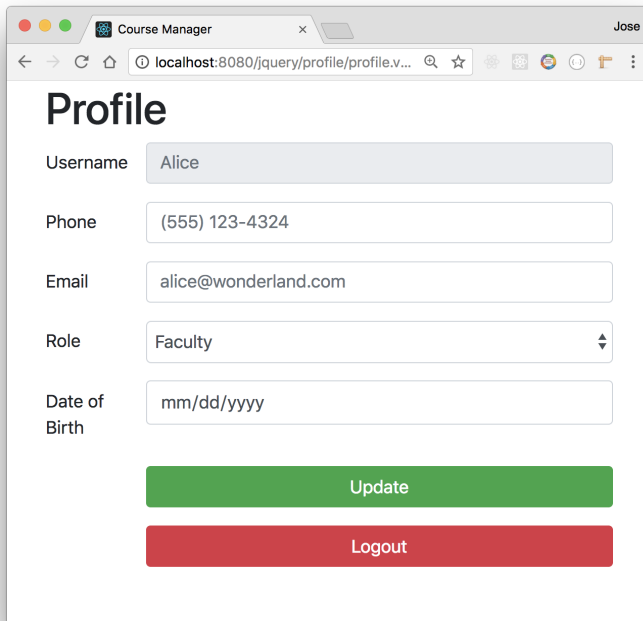
```
git commit command and message
```

```
git add .
git commit -m 'Created user signup web page'
```

Do the same for every section that follows where a new file is created, or new functionality is added. This is the last reminder.

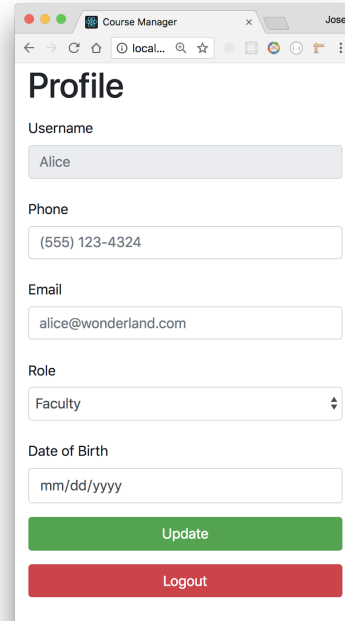
Profile requirements

Once a user registers, or logs in, they can view or edit their personal information from a profile webpage. The profile page shows a user's personal information once they've logged in or registered. The page is responsive having a desktop and mobile version as shown below.



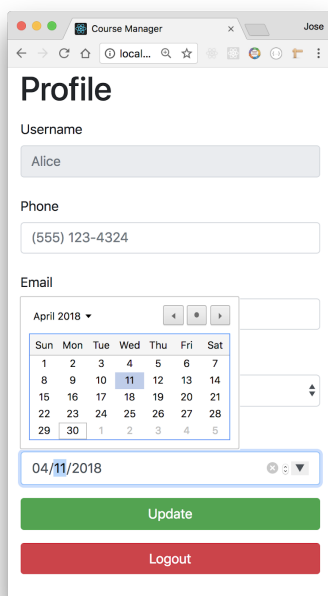
A desktop browser window titled 'Course Manager' showing a 'Profile' page. The page has a header with the title 'Profile'. Below it, there are input fields for 'Username' (containing 'Alice'), 'Phone' (containing '(555) 123-4324'), 'Email' (containing 'alice@wonderland.com'), 'Role' (a dropdown menu with 'Faculty' selected), and 'Date of Birth' (containing 'mm/dd/yyyy'). At the bottom, there are two buttons: a green 'Update' button and a red 'Logout' button.

Desktop profile wireframe

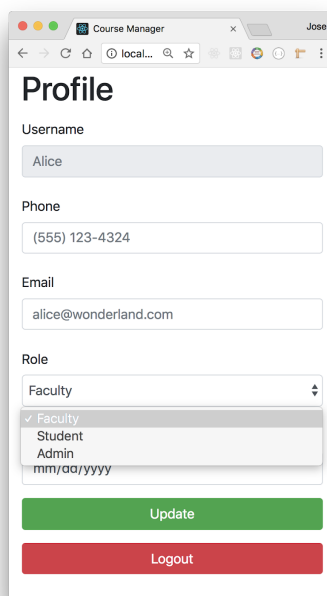


A mobile browser window titled 'Course Manager' showing a 'Profile' page. The page has a header with the title 'Profile'. Below it, there are input fields for 'Username' (containing 'Alice'), 'Phone' (containing '(555) 123-4324'), 'Email' (containing 'alice@wonderland.com'), 'Role' (a dropdown menu with 'Faculty' selected), and 'Date of Birth' (containing 'mm/dd/yyyy'). At the bottom, there are two buttons: a green 'Update' button and a red 'Logout' button.

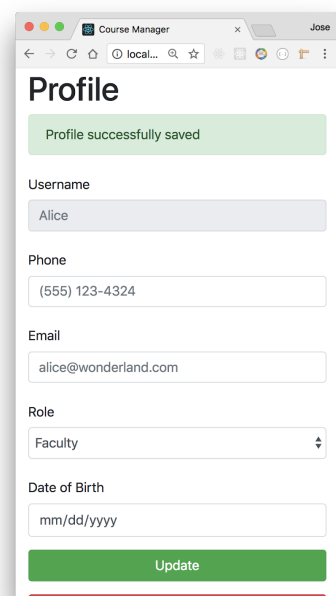
Mobile profile wireframe



A mobile browser window titled 'Course Manager' showing a 'Profile' page. The page has a header with the title 'Profile'. Below it, there are input fields for 'Username' (containing 'Alice'), 'Phone' (containing '(555) 123-4324'), 'Email' (containing 'alice@wonderland.com'), 'Role' (a dropdown menu with 'Faculty' selected), and 'Date of Birth' (containing 'mm/dd/yyyy'). At the bottom, there are two buttons: a green 'Update' button and a red 'Logout' button. A calendar widget is visible, showing the date '04/11/2018'.



A mobile browser window titled 'Course Manager' showing a 'Profile' page. The page has a header with the title 'Profile'. Below it, there are input fields for 'Username' (containing 'Alice'), 'Phone' (containing '(555) 123-4324'), 'Email' (containing 'alice@wonderland.com'), 'Role' (a dropdown menu with 'Faculty' selected), and 'Date of Birth' (containing 'mm/dd/yyyy'). At the bottom, there are two buttons: a green 'Update' button and a red 'Logout' button. The role dropdown menu is open, showing options: 'Faculty', 'Student', 'Admin', and 'mm/dd/yyyy'.



A mobile browser window titled 'Course Manager' showing a 'Profile' page. The page has a header with the title 'Profile'. Below it, there is a green message box that says 'Profile successfully saved'. Below the message box, there are input fields for 'Username' (containing 'Alice'), 'Phone' (containing '(555) 123-4324'), 'Email' (containing 'alice@wonderland.com'), 'Role' (a dropdown menu with 'Faculty' selected), and 'Date of Birth' (containing 'mm/dd/yyyy'). At the bottom, there is a green 'Update' button.

Profile date of birth date input

Profile role options

Profile successfully saved alert
message

Clicking on the Logout button navigates to the Home page. Clicking on the Update button saves the changes, stays in the same page, and a success alert appears at the top of the screen. The success alert can be static for now, and made dynamic in a later assignment. The date of birth input field is of type date and renders as a date picker. The username field is readonly and can not be modified. The role field is a select input field with options Faculty, Student, and Admin. The role field would typically be read only so that users can not change their own role. Role management would be an admin only use case, but let's allow users to change their role for now. Later assignments will change it to readonly as necessary.

Implement the profile template in a file called profile.template.client.html in a directory called webapp/profile. Use the wireframes, the sign in, and the sign up HTML code samples to get started.

File	Description
profile.template.client.html	implements the user interface using HTML
profile.style.client.css	implements the styling of the webpage using CSS













Implement styling specific to the profile template in a file called profile.style.client.css in a directory called webapp/profile. The template should render the title, labels, input fields, and buttons. Use the wireframe as a guidance. Feel free to improve on the layout and styling. Remember to make good use of white space (padding, margins), text wrapping, justification, and alignment throughout all your webpages.

webapp/profile/profile.template.client.html













```
<h1>Profile</h1>
Username: <input id="usernameFld"/>
Phone: <input id="phoneFld"/>
Email: <input id="emailFld"/>
Date of Birth: <input id="dobFld"/>
Role: <input id="roleFld"/>
  <select>
    <option value="Faculty"></option>
    <option value="Student"></option>
    <option value="Admin"></option>
  </select>
Update: <button id="updateBtn">Update</button>
Update: <button id="logoutBtn">Logout</button>
<div class="alertFld">
</div>
```


Home page and course list requirements













Faculty use the web application to author courses. The *course list view* lists all the courses authored by a faculty. The list of courses is the default landing page that serves as the home page of the application. Selecting a course navigates to that particular course rendered in a separate *course editor view*. Courses can be edited in the course editor where faculty can rename the course, add modules, lessons, topics, and content widgets. The following is a proposed wireframe for the course list. Notice the columns, headers, white space, and padding. You can use a different look and feel, but the functionality must be the same.

	<input type="text" value="New Course"/>	
Title ^		 
 Unselected Course		
 Selected Course		 
 Unselected Course		
 Editing Course	<input type="text" value=""/>	
 Unselected Course		

Portrait Layout

	<input type="text" value="New Course"/>	
Title ^	Owned by	 
 Unselected Course	me	
 Selected Course	me	 
 Unselected Course	me	
 Editing Course	me	
 Unselected Course	me	

Medium Landscape Layout

	Course Manager	<input type="text" value="New Course"/>	
Title ^	Owned by	Last modified	 
 Unselected Course	me	1/1/2020	
 Selected Course	me	1/1/2020	 
 Unselected Course	me	1/1/2020	
 Editing Course	me	1/1/2020	
 Unselected Course	me	1/1/2020	

Large Landscape Layout

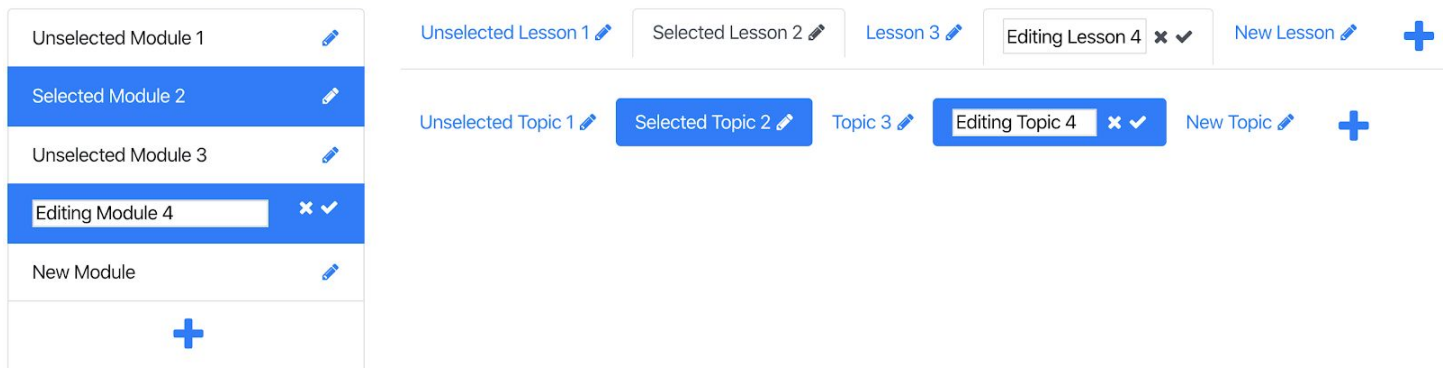
The course list view displays a list of courses. Each course row displays 4 columns: the title of the course, the owner of the course, the date and time last modified. An icon is shown to the left of the title. Selecting a course navigates to the course editor. The table layout must be responsive as shown above, hiding and showing columns as the width of the screens shrinks and widens. Two of the rows appear with a background color indicating that they are "active". The first one illustrates the row has been selected, whereas the second row illustrates the course title is being edited. Implement the course list layout and look and feel statically. There's no need to use JavaScript yet for this assignment.

Course editor requirements

A course is made up of many content or learning modules. This section describes how the application supports adding modules to a course. Selecting a course navigates to a course's **CourseEditor** webpage. The **CourseEditor** displays as a horizontal split showing a list of modules to the left, and lessons and topics to the right. Selecting a course shows the list of modules that make up that course.

Selecting *new module* adds a new module with a default name or the name in the *module name* input field. Selecting *delete module* displays a delete module confirmation (not shown). In later assignments, confirming delete module actually deletes the module and the module list updates. Selecting a module makes it the current module, displays the lessons for that module, and highlights the current module. On the right hand side of the course editor there tabs and pills representing the lessons and topics that make up a module. A topic contains many content widgets displaying various types of content such as slides, movies, HTML, etc.

✕ Web Dev Selected Course



Various buttons and controls allow managing the list of content widgets. Widgets are not required for this assignment. These include buttons to add and remove widgets, buttons to reorder widgets, dropdowns to select the widget type, and each widget provides different configuration fields specific to the widget type.

Source control requirements

Assignments and final project will be submitted through github source control. Use the school's github account. If you don't have one, create a free repository at github.com. Invite all instructors and TAs as collaborators to your github repository. They will need full access to clone your repository and help debug issues. There should be a commit for every single new file, and every new function or feature added to an already existing file. A point will be deducted for every missing commit with a maximum of 5 points.

Deliverables

As a deliverable, check in all your work into a github source control repository. Deploy your project to a remote public server on Heroku or AWS. Submit the URL of the repository and the remote server in blackboard. Tag the commit you want graded as assignment1. TAs will clone your repository and grade the tagged commit.

Name your repository and remote server using the course, term, section, your name, and role of the project. For instance, if this is the webdev course, spring 2019, section 1, your name is Ada Lovelace, and this is the Java server, an appropriate name for your github repository would be

wbdv-su19-ada-lovelace-server-java

wbdv-f19-jannunzi-server-java

References

[jQuery](#)

[Spring Tool Suite™ Downloads](#)

[Sign up for free Heroku account](#)

[Deploying Spring Boot Applications to Heroku](#)

[Deploying Spring Boot Applications to Heroku \(Slides\)](#)

Review

What do the following abbreviations and acronyms mean and what are they used for?

HTML, HTTP, CSS, UML, URL, CLI, STS

What are the following technologies used for?

HTML, CSS, Bootstrap, Spring Boot

What do the following terms mean?

Server, Client, Browser, One to many, Many to many, Inheritance, Cardinality, Relationship, Primary key, Foreign key, Private, Public, Protected, Web service, Path Parameter, Query Parameter, HTTP BODY, Cookie, Session