Implementing User Interfaces with HTML

1 Introduction

This assignment describes how to install the development environment for creating and working with Web applications we will be developing this semester. We will add new content every week, pushing the code to a GitHub source repository, and then deploying the content to a remote server hosted on Netlify.

Web pages consist of text documents that contain plain text formatted with ***HTML*** (***HyperText Markup Language***) tags embedded within the text. HTML is a computer language used to format the content displayed in Web pages. The formatting consists of configuring the foreground and background color, adding white spaces between text, aligning text, configuring font, creating lists, tables, and forms. In this assignment we will learn how to use HTML to format plain text into Web pages.

Assignments in this course contain three main sections: ***Lab***, ***Kanbas***, and ***Graduates***. In the ***Lab*** sections we hand hold you through several exercises intended for you to practice various skills. The ***Lab*** section in this assignment will give you an opportunity to practice the concepts described in this assignment, i.e., HTML. Once you've had a chance to practice with HTML, in the ***Kanbas*** section you'll be asked to apply what you've learned to building a Website inspired on a popular learning management system. The ***Graduates*** section is meant only for graduate students.

1.1 Learning Objectives

- Integrated Development Environments - Installing IntelliJ and VS Code
- Web Server - Installing ***Node.js***
- Front end framework - Creating ***React.js*** Web applications
- Source control - Pushing source code to ***GitHub.com***
- Hosting a Web application - Deploying Web applications to ***Netlify***
- Creating Web content with the ***HyperText Markup Language*** (***HTML***)
- Formatting Web content with ***HTML tags***
- Interacting with Web pages with ***HTML form tags***
- Navigating between Web pages with ***HTML anchor tags***

2 Lab

This section walks you through several exercises to familiarize yourself with HTML. First you'll setup a development environment and then practice several HTML exercises. Copy the examples into an HTML document as instructed and confirm that they render as intended. After practicing with the exercises you will be asked to apply the skills to create ***Kanbas***, a Web application inspired by a popular learning management system site. Using your favorite ***IDE***, such as ***IntelliJ*** or ***Visual Studio Code***, open the React.js project created earlier.

2.1 Web Servers

In this course will be creating Web servers from scratch using the ***JavaScript*** programming language running on local and remote computers. To run ***JavaScript*** we will need ***Node.js***, a popular ***JavaScript runtime environment***. Navigate to the URL below and download Node.js for your operating system. Download the recommended version and install it on your local computer.

https://nodejs.org/

2.2 Front End Frameworks

When you visit a Webpage on your browser, you are viewing a document formatted using the ***HyperText Markup Language*** (HTML). ***HTML*** documents are plain text documents that can be edited with any text editor. We refer to the software layer that interacts directly with a human as the ***front end***. As we demanded more from our Webpages, the industry evolved techniques, tools and frameworks to deal with the growing complexity. Many techniques were borrowed from other engineering fields. Today's front end frameworks implement advanced software engineering techniques and design patterns such as: Object oriented programming, Component technology, Singletons, Services, Model view controller.

There are many front end frameworks to choose from and they all implement some or all of the techniques above: Angular, Ember, Sail, Meteor, React.js, Vue.js, Dojo.

This course will focus on ***React.js*** since it has become one of the most popular front end libraries in the industry. Using ***npx***, a ***Node.js*** tool part of the Node.js installation, create a React.js project where we will be learning all about Web development. From the ***desktop***, or the ***root*** of your hard drive, or your ***home*** directory, create a directory for this semester, and then another directory under that for this course. Below are examples of creating directories from your home directory of your file system using a macOS and then Windows OS

On macOS, start the ***Terminal*** application. On Windows OS, start the console application ***PowerShell***. On either operating system, type the following to create the directory ***~/2024/spring/webdev***

```
$ cd  ~                                 # navigate to your home directory in your file system
$ mkdir  2024                           # creates a directory called 2024
$ mkdir  2024/spring                    # creates a directory called spring inside 2024
$ mkdir  2024/spring/webdev             # creates a directory called webdev inside spring/2024
$ cd  2024/spring/webdev                # navigates to the directory webdev inside spring/2024
```

You are free to choose other places in your file system, but if you do, please make sure all directory names:

- are all lowercase
- do not have spaces in them
- are inside directories that also meet these criteria

Still from the terminal or console, navigate to the directory you created for this course and type the following to create a brand new React.js project called ***kanbas-react-web-app*** using the ***create-react-app*** Node.js module.

```
$ npx  create-react-app  kanbas-react-web-app  --template  typescript
```

A new directory called ***kanbas-react-web-app*** will be created and lots of libraries and code will be downloaded into the new directory. Wait for the process to complete and then navigate into the new directory and run the project.

```
$ cd   kanbas-react-web-app
$ npm  start
```

Once you've confirmed the React application is working fine, let's break it for now by commenting the whole ***index.tsx*** file under the ***src*** directory. Comment all but the React import statement. This will allow us to focus on learning plain HTML, without the added complexity of ***JavaScript*** and ***TypeScript***. We'll uncomment the file in the next assignment. Using your favorite IDE, e.g., ***Visual Code***, open the ***kanbas-react-web-app*** directory, and then the ***public*** directory. Do all your work under the ***public*** directory of your project. Starting from this assignment, you will create a separate ***GitHub*** branch for each assignment. Commit your work for this assignment into a branch called ***a1***. Commit work in subsequent assignments 2, 3, 4, and 5 into branches ***a2***, ***a3***, ***a4***, and ***a5.*** Once you get a grade for the assignment, you can merge it into the ***main*** branch. We will configure Netlify so that each branch will deploy to a separate URL. Let's start learning how to use HTML to create a ***User Interface***. Follow along in Chapters 1 and 2 of the ***Full Stack Developer*** book assigned for this course.

2.3 Heading Tags

Text documents are often broken up into several sections and subsections. Each section is usually prefaced with a short title or ***heading*** that attempts to summarize the topic of the section it precedes. For instance this paragraph is preceded by the heading ***Heading Tags***. The font of the section headings are usually larger and bolder than the plain text and their subsection headings. This document uses headings to introduce topics such as HTML Documents, HTML Tags, Heading Tags, etc. HTML ***heading tags*** can be used to format plain text so that it renders in a browser as large headings. There are 6 heading tags: ***h1***, ***h2***, ***h3***, ***h4***, ***h5***, and ***h6***. Tag ***h1*** is the largest heading and ***h6*** is the smallest heading.

To practice using the heading tags we are going to create several headings and subheadings to introduce the topics we will cover in this assignment. Under the ***public*** directory, create directory ***labs/a1*** where you will practice several HTML exercises for assignment 1. Under the ***labs/a1*** directory, create an HTML file called ***index.html***. Copy the HTML below into the ***<body>*** tag of this new file.

Then, after the ***Heading Tags*** heading (highlighted in red here on the right), copy and paste the first paragraph of this section highlighted in yellow. To see the content of the Webpage, open the file with your browser. The file will open in a browser window and the content should look similar to the content highlighted yellow at the beginning of this section. Note how the text surrounded by the ***<h1>*** tag is larger and bolder than the text surrounded by the ***<h2>*** tag, and both are larger than the text that has no tags around it. The ***index.html*** document will be part of your deliverable and will contain the exercises that follow. The ***index.html*** file should now look as shown below. Confirm by visiting http://localhost:3000/labs/a1/index.html

public/labs/a1/index.html	How the browser renders

`<html>`
`  <head>`
`  <title>Lab 1</title>`
`  </head>`
`  <body>`
`  <h1>HTML Examples</h1>`
`  <h2>Heading Tags</h2>`
`  </body>`
`</html>`
	HTML Examples
	Heading Tags

public/labs/a1/index.html

```
<html>
  <head>
    <title>Lab 1</title>
  </head>
  <body>
    <h1>HTML Examples</h1>
    <h2>Heading Tags</h2>
    Text documents are often broken up into several sections and subsections. Each section is usually prefaced with a short title or heading that attempts to summarize the topic of the section it precedes. For instance this paragraph is preceded by the heading Heading Tags. The font of the section headings are usually larger and bolder than their subsection headings. This document uses headings to introduce topics such as HTML Documents, HTML Tags, Heading Tags, etc. HTML heading tags can be used to format plain text so that it renders in a browser as large headings. There are 6 heading tags: h1, h2, h3, h4, h5, and h6. Tag h1 is the largest heading and h6 is the smallest heading.

  <!-- Do the rest of the exercises here. Make sure it's all inserted before the closing body tab below -->

  </body>
</html>
```

2.4 Paragraph Tag

Browsers ignore white spaces such as tabs and newlines. To add space between different paragraphs we can use the paragraph tag ***<p>***. Wrap text with the paragraph tag to add vertical spacing. To practice using the paragraph tag, copy the code on the right at the end of the ***index.html***, but still within the ***<body>*** tag, e.g., before the closing ***</body>*** tag. Below is another example of how the browser renders HTML text on the left column. Note how the browser ignores line breaks and other white space

```
<h2>Paragraph Tag</h2>
<p>
   This is a paragraph. We often separate a long set of sentences with vertical spaces to make the text easier to read. Browsers ignore vertical white spaces and render all the text as one single set of sentences. To force the browser to add vertical spacing, wrap the paragraphs you want to separate with the paragraph tag
</p>
```

formatting like tabs and content just flows from left to right and then wraps when there's no more horizontal space. This style of rendering is referred to as ***inline***. Inline content flows from left to right horizontally the whole width of its parent container and then wraps vertically when there's no more space.

public/labs/a1/index.html	How the browser renders

`This is the first paragraph. The paragraph tag is used to format vertical gaps between long pieces of text like this one.`
 	
`This is the second paragraph. Even though we added a deliberate gap between the paragraph above and this paragraph, by default browsers render them as one contiguous piece of text as shown here on the right.`
 	
`This is the third paragraph. Wrap each paragraph with the paragraph tag to tell browsers to render the gaps.`
	This is the first paragraph. The paragraph tag is used to format vertical gaps between long pieces of text like this one. This is the second paragraph. Even though we added a deliberate gap between the paragraph above and this paragraph, by default browsers render them as one contiguous piece of text as shown here on the right. This is the third paragraph. Wrap each paragraph with the paragraph tag to tell browsers to render the gaps.

Applying the paragraph tags below lets the browser know we want to keep the vertical spacing.

public/labs/a1/index.html	How the browser renders

`<p>`
`This is the first paragraph. The paragraph tag is used to format vertical gaps between long pieces of text like this one.`
`</p>`
`<p>`
`This is the second paragraph. Even though there is a deliberate white gap between the paragraph above and this paragraph, by default browsers render them as one contiguous piece of text as shown here on the right.`
`</p>`
`<p>`
`This is the third paragraph. Wrap each paragraph with the paragraph tag to tell browsers to render the gaps.`
`</p>`
	This is the first paragraph. The paragraph tag is used to format vertical gaps between long pieces of text like this one.
	
	This is the second paragraph. Even though there is a deliberate white gap between the paragraph above and this paragraph, by default browsers render them as one contiguous piece of text as shown here on the right.
	
	This is the third paragraph. Wrap each paragraph with the paragraph tag to tell browsers to render the gaps.

Copy the HTML above on the left to the end of the ***index.html*** document in the ***Paragraph Tag*** section, but still within the ***<body>*** tag, e.g., before the closing ***</body>*** tag. Remember to keep all your content within the ***body*** tag. Refresh the Webpage and confirm it renders as shown on the right. Note how the paragraphs are now spaced vertically from one another. Both the paragraph and heading tags add vertical space and we refer to this style of rendering as ***block***. By controlling the inline and block styles of laying out content, we can achieve all sorts of useful layouts.

2.5 List Tags

2.5.1 Ordered List Tag

List tags are used to create lists of related items. There are two types of lists: ***ordered*** and ***unordered***. Ordered list tags are useful for listing items in a particular order. Here's a list of steps to join the 1% in 4 "easy" steps.

public/labs/a1/index.html	How the browser renders

`<h2>List Tags</h2>`
`<h3>Ordered List Tag</h3>`
`How to make pancakes:`
`1. Mix dry ingredients.`
`2. Add wet ingredients.`
`3. Stir to combine.`
`4. Heat a skillet or griddle.`
`5. Pour batter onto the skillet.`
`6. Cook until bubbly on top.`
`7. Flip and cook the other side.`
`8. Serve and enjoy!`
	List Tags
	
	Ordered List Tag
	
	How to make pancakes: 1. Mix dry ingredients. 2. Add wet ingredients. 3. Stir to combine. 4. Heat a skillet or griddle. 5. Pour batter onto the skillet. 6. Cook until bubbly on top. 7. Flip and cook the other side. 8. Serve and enjoy!

Note that in the HTML text on the left we explicitly wrote the numbers 1., 2., etc., but the nice formatting is lost when the browser renders it on the right. Instead of rendering a list of items, each in its own line, they are instead all rendered on the same line. To achieve the desired format we'll use the ordered list tag. The ordered list tag actually consists of a pair of tags

- `<ol>` declares the beginning of the list
- `<li>` declares an item in the list

Here's the same example from earlier, but now applying the ordered list tags to achieve the intended formatting.

public/labs/a1/index.html	**How the browser renders**

```
<h2>List Tags</h2>
<h3>Ordered List Tag</h3>
How to make pancakes:
<ol>
   <li>Mix dry ingredients.</li>
   <li>Add wet ingredients.</li>
   <li>Stir to combine.</li>
   <li>Heat a skillet or griddle.</li>
   <li>Pour batter onto the skillet.</li>
   <li>Cook until bubbly on top.</li>
   <li>Flip and cook the other side.</li>
   <li>Serve and enjoy!</li>
</ol>
```

List Tags

Ordered List Tag

How to make pancakes:

1. Mix dry ingredients.
2. Add wet ingredients.
3. Stir to combine.
4. Heat a skillet or griddle.
5. Pour batter onto the skillet.
6. Cook until bubbly on top.
7. Flip and cook the other side.
8. Serve and enjoy!

Copy the HTML above to the end of ***index.html*** file and confirm it renders as shown on the right.

2.5.2 Unordered List Tag

The unordered list tag is similar to it's ordered version with the difference that the items are not numbered and instead bullets decorate each line item. The unordered list tag is ***<ul>***, but the list item tag is still ***<li>*** as shown below. Unordered lists are great for displaying a list of items in no particular order. Here's an example of an unordered list of my favorite books in no particular order. Add the example HTML code above to the end of the index.html document to include it in your deliverable.

public/labs/a1/index.html	**How the browser renders**

```
<h3>Unordered List Tag</h3>
My favorite books (in no particular order)
<ul>
   <li>Dune</li>
   <li>Lord of the Rings</li>
   <li>Ender's Game</li>
   <li>Red Mars</li>
   <li>The Forever War</li>
</ul>
```

Unordered List Tag

My favorite books (in no particular order)

- Dune
- Lord of the Rings
- Ender's Game
- Red Mars
- The Forever War

2.6 Table Tags

HTML began as a tool for sharing research results between scientists. These documents often consisted of data points captured as a result of some experiment. Each data point might have several attributes associated. An effective way to display or visualize these results were formatted in a data table with a row for each data point and a column for each attribute. The ***<table>*** tag allows formatting data into a table with rows and columns. For instance, consider capturing grade results for several quizzes you might have taken over a semester. These might be captured using the following table.

Quiz	Topic	Date	Grade

Q1	HTML	2/3/21	85
Q2	CSS	2/10/21	90
Q3	JavaScript	2/17/21	95
Average			90

Several things to note:

1. The first row is formatted as headings for each column
2. There are 3 data points, one for each quiz, one in each row
3. Each data point has the same data types for each of the columns, e.g, Quiz, Topic, Date, Grade
4. The last row is formatted as a footer
5. The three first columns of the last row are merged into a single cell and unlike the 3 data rows

HTML ***table*** tag can be used to format the data with the following tags:

- ***table*** - declares the start of a table
- ***tr*** - declares the start of a row
- ***td*** - declares a table data cell
- ***thead*** - declares a row of headings
- ***tbody*** - declares the main data content rows of the table
- ***tfoot*** - declares a row as a footer
- ***th*** - declares a table cell as a heading

To practice using ***table*** tag, copy the HTML below to the end of index.html. The code implements the table shown earlier. You can ignore the comments on the right.

```
<h2>Table Tag</h2>
<table border="1" width="100%">                    <!-- declares the table, sets border and width -->
   <thead>                                         <!-- declares the table heading section -->
   <tr>                                            <!-- declares the headings row -->
       <th>Quiz</th>                               <!-- declares heading for first column -->
       <th>Topic</th>                              <!-- declares heading for second column -->
       <th>Date</th>                               <!-- declares heading for third column -->
       <th>Grade</th>                              <!-- declares heading for fourth column -->
   </tr>
   </thead>
   <tbody>                                         <!-- declares the table's main content -->
   <tr>                                            <!-- declares the first row -->
       <td>Q1</td>                                 <!-- declares data for first row, first column -->
       <td>HTML</td>                               <!-- declares data for first row, second column -->
       <td>2/3/21</td>                             <!-- declares data for first row, third column -->
       <td>85</td>                                 <!-- declares data for first row, fourth column -->
   </tr>
   <tr>                                            <!-- declares the second row -->
       <td>Q2</td>                                 <!-- declares data for second row, first column -->
       <td>CSS</td>                                <!-- declares data for second row, second column -->
       <td>2/10/21</td>                            <!-- declares data for second row, third column -->
       <td>90</td>                                 <!-- declares data for second row, fourth column -->
   </tr>
   <tr>                                            <!-- declares the third row -->
       <td>Q3</td>                                 <!-- declares data for third row, first column -->
       <td>JavaScript</td>                         <!-- declares data for third row, second column -->
       <td>2/17/21</td>                            <!-- declares data for third row, third column -->
       <td>95</td>                                 <!-- declares data for third row, fourth column -->
   </tr>
   </tbody>
   <tfoot>                                         <!-- declares the table footer section -->
   <tr>                                            <!-- declares the footer row -->
       <td colspan="3">Average</td>                <!-- declares data spans 3 columns -->
       <td>90</td>                                 <!-- declares data for fourth column -->
   </tr>
   </tfoot>
</table>
```

Content in a cell can be aligned to the top of the cell using the `valign` attribute set to `"top"`, e.g., `<td valign="top">`.

2.7 Image Tag

Use the image tag to render pictures in your HTML documents. The images can be anywhere on the internet, or a local image document in your local file system.

```
<img src="my-picture.jpg"
    width="200px"
    height="300px"/>
```

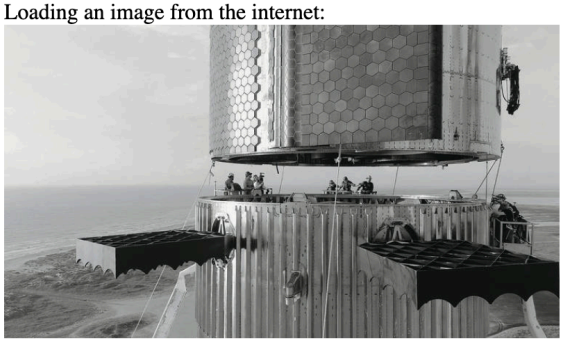
```
<!-- Use img tag to embed pictures in HTML documents.
src attributes references image file either locally or remotely. width and height
attributes configure the image size. If only width or height is provided, the
other scales proportionally -->
```

To practice using the image tag, copy the code below at the end of ***index.html***. The first image tag embeds an image from a remote server. The second one assumes you have a local image file called ***teslabot.jpg***. Search for Tesla Bot on the internet, and download an image that looks similar to the one shown below. Name the image ***teslabot*** keeping the original file extension.

```
<h2>Image tag</h2>

Loading an image from the internet:<br/>

<img width="400px"

src="https://www.staradvertiser.com/wp-content/uploads/2021/08/web
1_Starship-gap2.jpg"/>
<br/>

Loading a local image:<br/>

<img src="teslabot.jpg"
    height="200px"/>
```

Image tag

Loading an image from the internet:



Loading a local image:

2.8 Form Tags

Form tags are useful for entering data. Let's take a look at the most common ones: ***form***, ***input***, ***select***, ***textarea***, ***radio***, ***checkbox***.

2.8.1 Text fields

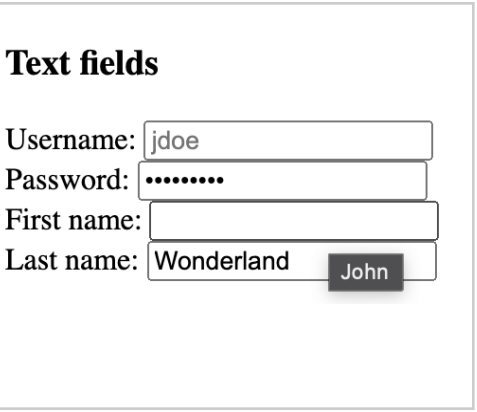
Text fields are the most common form elements allowing entering a single line of text.

```
<input  type="text"
       placeholder="hint"
       title="tooltip"
       value="COMEDY"/>
```

```
<!--    use input tag's text type to declare a single line input field text is
default if type is left out. Use placeholder and title to give a hint of what
information you're expecting. Optionally initialize the value of the field with
value attribute-->
```

To practice using text fields, add the following example at the end of ***index.html***. It creates a set of input fields for entering some personal information. The ***label*** tags below associate descriptive text with each form element. The is established by setting a ***label***'s ***for*** attribute to the ***id*** attribute of the related form field.

```
<h2>Text fields</h2>
<form id="text-fields">
   <label for="text-fields-username">Username:</label>
   <input id="text-fields-username" placeholder="jdoe"/><br/>
   <label for="text-fields-password">Password:</label>
   <input type="password" id="text-fields-password"
          value="123@#$asd"/><br/>
   <label for="text-fields-first-name">First name:</label>
   <input type="text" id="text-fields-first-name"
          title="John"/><br/>
   <label for="text-fields-last-name">Last name:</label>
   <input type="text" id="text-fields-last-name"
          placeholder="Doe" value="Wonderland"/>
   <!-- copy rest of form elements here -->
</form>
```



2.8.2 Textareas

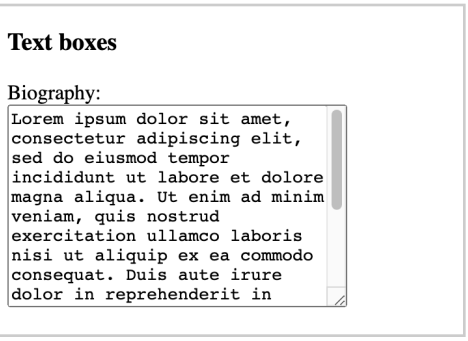
The ***textarea*** tag is useful for entering long form text such as someone's biography data, or a blog post.

```
<textarea cols="20" rows="25"
          placeholder="Biography"
          title="tooltip">Some text
</textarea>
```

```
<!--    use textarea tag for long form text configure its width and
height with attributes cols and rows. Use placeholder and tooltip to
give hints. Note default value is in tag's body -->
```

To practice using the ***textarea*** tag, add the following example to the end of index.html. It creates a ***textarea*** useful for entering your biography. You can get a sample of the dummy text at https://www.lipsum.com/.

```
<h2>Text boxes</h2>
<form id="textarea">
   <label>Biography:</label><br/>
   <textarea cols="30" rows="10">Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.</textarea>
</form>
```



2.8.3 Buttons

Buttons allow invoking actions executed by the browser. To practice creating buttons, copy the code below at the end of ***index.html***.

```
<h3>Buttons</h3>
<button type="button">Click me!</button>
```

2.8.4 File upload button

Use the file type for the input tag to choose a file for upload. We won't be able to upload just yet until later in the course, but for now let's practice adding a file upload tag as shown below. Clicking the button pops up a file choose where you can navigate to the file you want to upload. To practice using the file selector, copy the code below to the end of ***index.html***. We'll learn how to upload files later in the course.

```
<h2>File upload</h2>
<input type="file"/>
```

2.8.5 Radio buttons

Radio buttons allow selecting a single choice from multiple alternative options

```
<input type="radio" name="NAME1"
        value="OPTION1"/>
<input type="radio" name="NAME1"
        value="OPTION2" checked/>
```

```
<!--    use the input tag's checkbox type to declare a checkbox give the checkbox a value -->
```

To practice using radio buttons, add the following example at the end of ***index.html***.

```
<h2>Radio buttons</h2>

<label>Favorite movie genre:</label><br/>

<input type="radio" value="COMEDY"
      name="radio-genre" id="radio-comedy"/>
<label for="radio-comedy">Comedy</label><br/>
<input type="radio" value="DRAMA"
      name="radio-genre" id="radio-drama"/>
<label for="radio-drama">Drama</label><br/>
<input type="radio" value="SCIFI"
      name="radio-genre" id="radio-scifi" checked/>
<label for="radio-scifi">Science Fiction</label><br/>
<input type="radio" value="FANTASY"
      name="radio-genre" id="radio-fantasy"/>
<label for="radio-fantasy">Fantasy</label>
```

2.8.6 Checkboxes

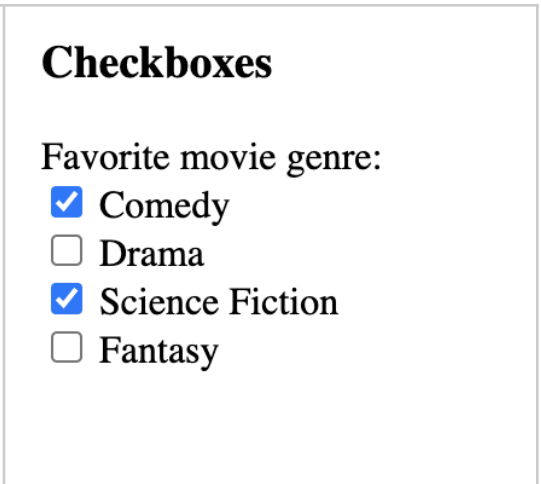
Checkboxes allow selecting multiple choices

```
<input type="checkbox" name="NAME2"
       value="OPTION1" checked/>
<input type="checkbox" name="NAME2"
       value="OPTION2"/>
<input type="checkbox" name="NAME2"
       value="OPTION3" checked/>
```

```
<!--    use the input tag's checkbox type to declare a checkbox give the checkbox a value       -->
```

To practice using checkboxes, add the following example to the end of ***index.html***. It creates a set of checkbox buttons to select all your favorite movie genres, which there might be more than one.

```
<h2>Checkboxes</h2>
<label>Favorite movie genre:</label>
<br/>
<input type="checkbox" value="COMEDY"
      name="check-genre" id="chkbox-comedy" checked/>
<label for="chkbox-comedy">Comedy</label>    <br/>

<input type="checkbox" value="DRAMA"
      name="check-genre" id="chkbox-drama"/>
<label for="chkbox-drama">Drama</label>    <br/>

<input type="checkbox" value="SCIFI"
      name="check-genre" id="chkbox-scifi" checked/>
<label for="chkbox-scifi">Science Fiction</label> <br/>

<input type="checkbox" value="FANTASY"
      name="check-genre" id="chkbox-fantasy"/>
<label for="chkbox-fantasy">Fantasy</label>
```



2.8.7 Dropdowns

Dropdowns are useful for selecting one or more options from a list of possible values. The default version displays a set of values from which you can choose a single value.

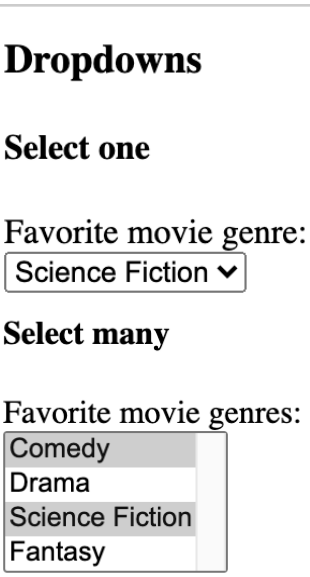
```
<select>                                                    <!--    Wrap several option tags in a select tag.
   <option value="VAL1">Value 1</option>                    Optionally provide option's value, otherwise the
   <option value="VAL2" selected>Value 2</option>           option's text is the value of the select element.
   <option value="VAL3">Value 3</option>                    Optionally use selected attribute to select default.
</select>                                                           -->
```

Adding the optional ***multiple*** attribute converts the dropdown into a list of options that can be selected.

```
<select multiple>                                       <!--    Alternatively use attribute multiple to
   <option value="VAL1" selected>Value 1</option>       allow selecting more than one option.    Use
   <option value="VAL2">Value 2</option>                ctrl+click to select more than one option -->
   <option value="VAL3" selected>Value 3</option>
</select>
```

To practice using the ***select*** tag, add the following example to the end of ***index.html***. It creates a dropdown and a list of options.

```
<h2>Dropdowns</h2>

<h3>Select one</h3>
<label for="select-one-genre"> Favorite movie genre: </label><br/>
<select id="select-one-genre">
   <option value="COMEDY">Comedy</option>
   <option value="DRAMA">Drama</option>
   <option selected value="SCIFI">
       Science Fiction</option>
   <option value="FANTASY">Fantasy</option>
</select>

<h3>Select many</h3>
<label for="select-many-genre"> Favorite movie genres: </label><br/>
<select id="select-many-genre" multiple>
   <option selected value="COMEDY">Comedy</option>
   <option value="DRAMA">Drama</option>
   <option selected value="SCIFI">
       Science Fiction</option>
   <option value="FANTASY">Fantasy</option>
</select>
```



2.8.8 Other HTML field types

The input tag's ***type*** attribute has several other possible values: ***date***, ***email***, ***number***, and ***range***. They can be used to enter text information with a specific format. To practice these other formats add the following example under the last input field you worked on earlier, but inside the ***form*** tag. The fields should look as shown below on the right.

```
<h2>Other HTML field types</h2>
<label for="text-fields-email"> Email: </label>
<input type="email"
      placeholder="jdoe@somewhere.com"
      id="text-fields-email"/><br/>
<label for="text-fields-salary-start"> Starting salary:
</label>
<input type="number"
      id="text-fields-salary-start"
      placeholder="1000"
      value="100000"/><br/>
<label for="text-fields-rating"> Rating: </label>
```

```
<input type="range" id="text-fields-rating"
       placeholder="Doe"
       max="5"
       value="4"/><br/>
<label for="text-fields-dob"> Date of birth: </label>
<input type="date"
       id="text-fields-dob"
       value="2000-01-21"/><br/>
```

2.9 Anchor Tag

The anchor tag allows navigating to other websites or other pages within the same website.

```
<a href="aa.com">                    <!--    Use the href attribute to refer to the location of the website or other
American Airlines</a>                page in the same website. Click on the body text to navigate -->
```

To practice using anchor tags, add the following example at the end of ***index.html***. It creates two hyperlinks. One navigates to ***lipsum.com***, a website that contains dummy text, and the other link navigates to another document located in the same website. Create the ***other-page.html*** document in the same directory as ***index.html*** and fill it with some dummy text. Click the ***other page*** link and confirm the navigation works.

```
<h2>Anchor tag</h2>

Please
<a href="https://www.lipsum.com">click here</a>
to get dummy text<br/>

Checkout my <a href="other-page.html">other page</a>
```

In the existing ***public/index.html*** file, create hyperlinks to navigate to the labs as shown below. Make sure to comment out the root div as shown.

public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head> ...
  </head>
  <body>
    <h1>Welcome to Web Development!</h1>
    <ul>
      <li><a href="/labs/a1/index.html">Assignment 1</a></li>
      <li><a href="/labs/a2/index.html">Assignment 2</a></li>
      <li><a href="/labs/a3/index.html">Assignment 3</a></li>
      <li><a href="/Kanbas/Courses/Home/screen.html">Kanbas</a></li>
    </ul>
    <!--
      <noscript>You need to enable JavaScript to run this app.</noscript>
      <div id="root"></div>
    -->
  </body>
</html>
```

```
//
```

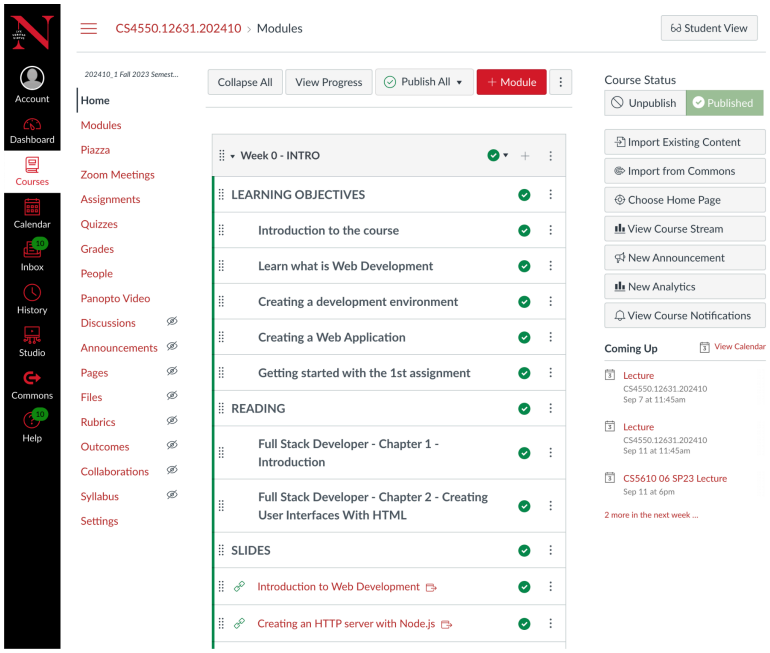
2.10 Exercises

In an ***index.html*** file in ***public/labs/a1/index.html***, complete the following exercises as described in the sections 2.1 through 2.2 listed below

1. Create a React.js application as described in section 2.1 and 2.2
2. Implement the headings as described in section 2.3
3. Implement the paragraphs as described in section 2.4
4. Implement the lists as described in section 2.5
5. Implement the tables as described in section 2.6
6. Implement the images as described in section 2.7
7. Implement the form tags as described in section 2.8
8. Implement the anchor tags as described in section 2.9

3 Implementing the Kanbas User Interface with HTML

Now that you've had a chance to practice various aspects of HTML, we're going to use them to build ***Kanbas***, a Web site based on a popular online ***Learning Management System*** (LMS). Over the next several assignments we're going to implement several screens giving us an opportunity to practice the skills discussed in lectures. We'll start with some simple versions of the more common screens and we'll improve on them over several assignments. Do your work under a new ***public/Kanbas*** directory.



3.1 Implementing the Course Home Screen

Let's start with the ***Home*** screen shown here on the right. The home screen displays a list of modules that contain a course's material. The home screen is accessible by clicking the ***Home*** link on the left navigation sidebar. The screen shot here on the right illustrates the first module ***Week 0 - INTRO***, introducing the ***LEARNING OBJECTIVES***, ***READING***, as well as providing links to the ***SLIDES***. In this assignment we're only going to capture the main content using just tables, lists, headings, paragraphs, and anchors. In upcoming assignments we will revisit these screens and apply styling with CSS. The first column renders a list of hyperlinks (anchors) to navigate across multiple courses in the ***Kanbas*** application. Users can navigate to the home screen by clicking ***Courses*** on the ***Kanbas Navigation*** side bar shown above in the first column. Then users click the ***Home*** on the ***Course Navigation*** side bar in the second column. The third column renders the home screen displaying the course content. The last column on the right provides additional information. Implement the ***Kanbas Navigation*** sidebar and ***Course Navigation*** sidebars in ***/public/Kanbas/Navigation/index.html*** and ***/public/Kanbas/Courses/Navigation/index.html*** respectively, shown in Figures 3.1.1 and 3.1.2.

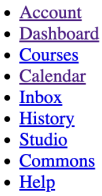


Figure 3.1.1 - Kanbas Navigation

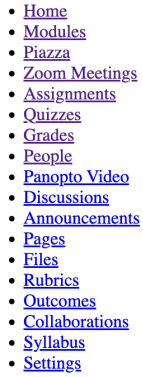


Figure 3.1.2 - Course Navigation

Here's an example of what the ***Kanbas*** and ***Course Navigation*** files might look like:

/public/Kanbas/Navigation/index.html

```
<ul>
 <li><a href="/Kanbas/Account/Navigation/index.html">
    Account</a></li>
 <li>
  <a href="/Kanbas/Dashboard/screen.html">Dashboard</a>
 </li>
 <li>
  <a href="/Kanbas/Courses/Home/screen.html">Courses</a>
 </li>
 <li>
  <a href="/Kanbas/Calendar/index.html">Calendar</a></li>
 <li><a href="/Kanbas/Inbox/index.html">Inbox</a></li>
</ul>
```

/public/Kanbas/Courses/Navigation/index.html

```
<ul>
  <li>
    <a href="/Kanbas/Courses/Home/screen.html">
      Home</a></li>
  <li><a href="/Kanbas/Courses/Modules/screen.html">
      Modules</a></li>
  <li><a href="/Kanbas/Courses/Piazza/index.html">
      Piazza</a></li>
  <li><a href="/Kanbas/Courses/Grades/screen.html">
      Grades</a></li>
  <li><a href="/Kanbas/Courses/Assignments/screen.html">
      Assignments</a></li>
</ul>
```

In ***/public/Kanbas/Courses/Home/index.html***, implement the main content of the ***Home*** screen as shown in Figure 3.1.3. The left side has buttons ***Collapse All***, ***View Progress***, ***Module***, and the ***Publish All*** dropdown. Below the buttons we have nested bullet lists listing ***Week 0***, ***Week 1***, and ***Learning Objectives*** under each. The right hand side contains some headings, buttons and hyperlinks as shown below. Feel free to change the text, except the button labels. The links and buttons don't need to do anything yet.

Figure 3.1.3 - Home

You can use a table to layout the two columns in the ***Home*** screen as shown below on the left. You can nest unordered lists as shown below on the right to implement the ***Weeks*** and nested ***Learning Objectives***.

Use Tables to Layout Columns

```
<table>
  <tbody>
    <tr>
      <td>Left Side Column</td>
      <td>Right Side Column</td>
    </tr>
  </tbody>
</table>
```

Nested Lists

```
<ul>
  <li> Week 1
    <ul>
      <li>Learning Objectives
      <li>Introduction to the course</li>
      <li>Learn what is Web Development</li>
    </ul>
  </li>
  <li> Week 2
    <ul>
      <li>Learning Objectives</li>
    </ul>
  </li>
</ul>
```

Once you have completed the ***Kanbas Navigation*** sidebar, ***Course Navigation*** sidebar, and ***Home*** to your satisfaction, combine the content of the three into a ***Home Screen*** in a new file ***/public/Kanbas/Courses/Home/screen.html***, as shown below Figure 3.1.4. Use a table with three columns. Copy the content of the ***Kanbas Navigation*** into the first column, then copy the content of the ***Course Navigation*** into the second column, and then copy the content of ***Home*** into the third column. In later chapters we will be using JavaScript to do this programmatically instead of manually. If you know how to, feel free to give it a try. Confirm you can navigate to this Web page from http://localhost:3000/index.html.

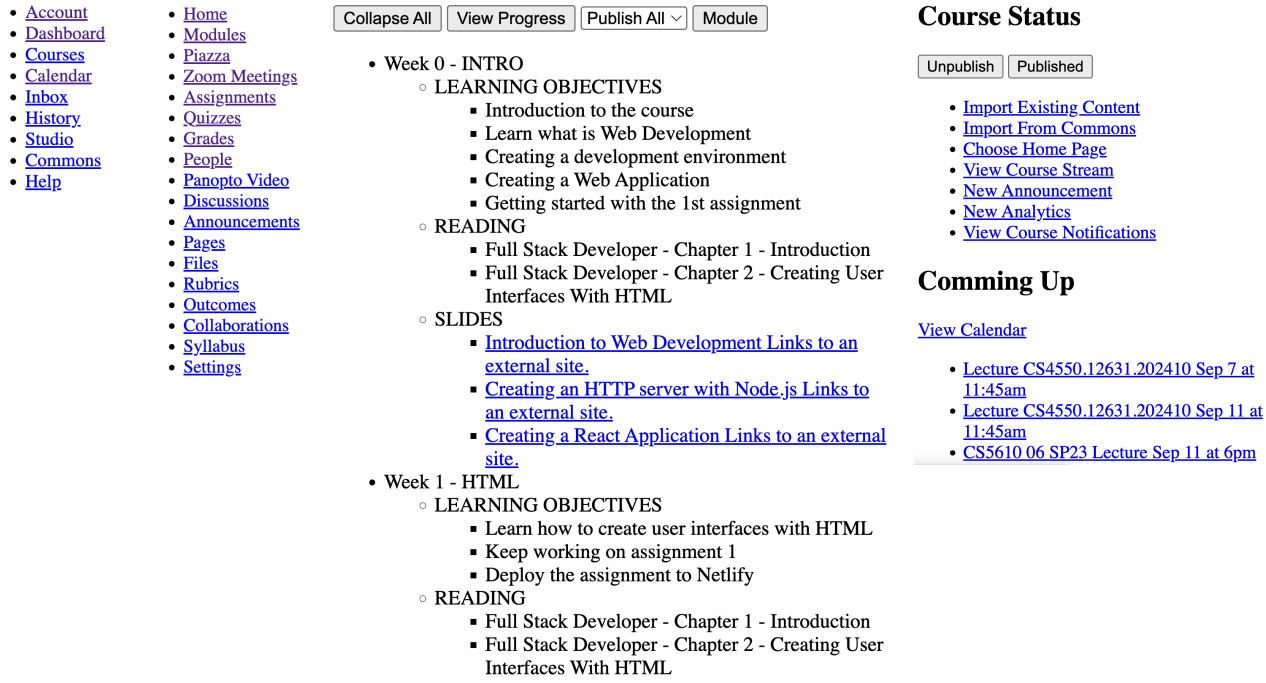


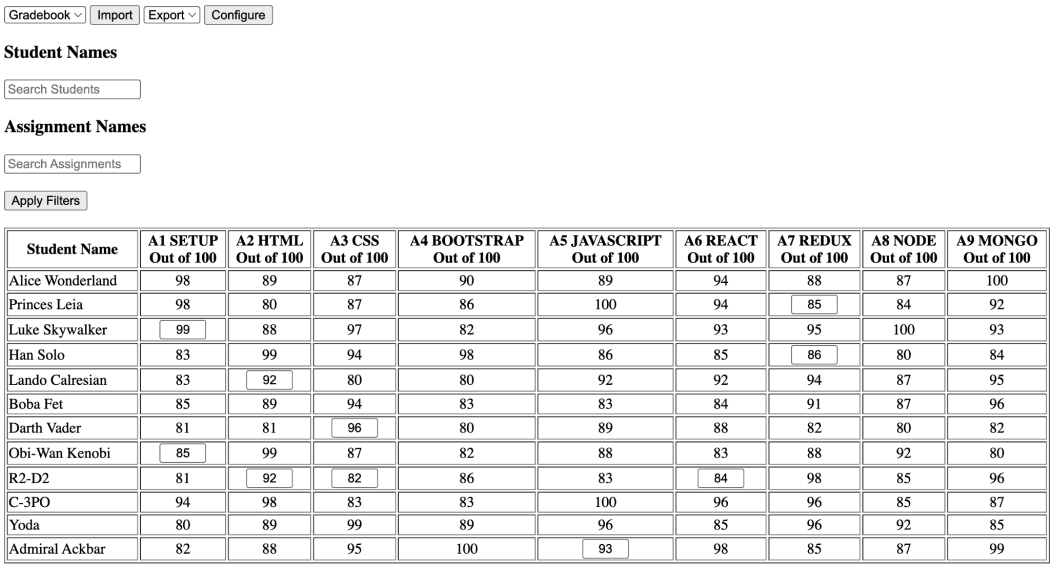
Figure 3.1.4 - Home Screen

3.2 Grades screen

The ***Grades*** screen displays the scores students have received in the various evaluations throughout the course. Each student is represented as a separate row with one column for each of the assignments, quizzes, and projects they worked on. Users can navigate to the grades screen by clicking on ***Courses*** in the ***Kanbas Navigation*** side bar and then on ***Grades*** in the ***Course Navigation*** side bar. Clicking on a grade allows faculty and TAs to set the grade for a particular assignment.

For this assignment we are just going to capture the general layout and content of the screen. We will revisit this screen in later assignments to apply CSS styling so that it renders closer to the screen shot shown here. In

public/Kanbas/Courses/Grades/index.html implement the main content of the ***Grades*** screen consisting of the form elements and table containing the assignments, students, and their grades as shown in Figure 3.2.1.



Gradebook | Import | Export | Configure

Student Names

Search Students

Assignment Names

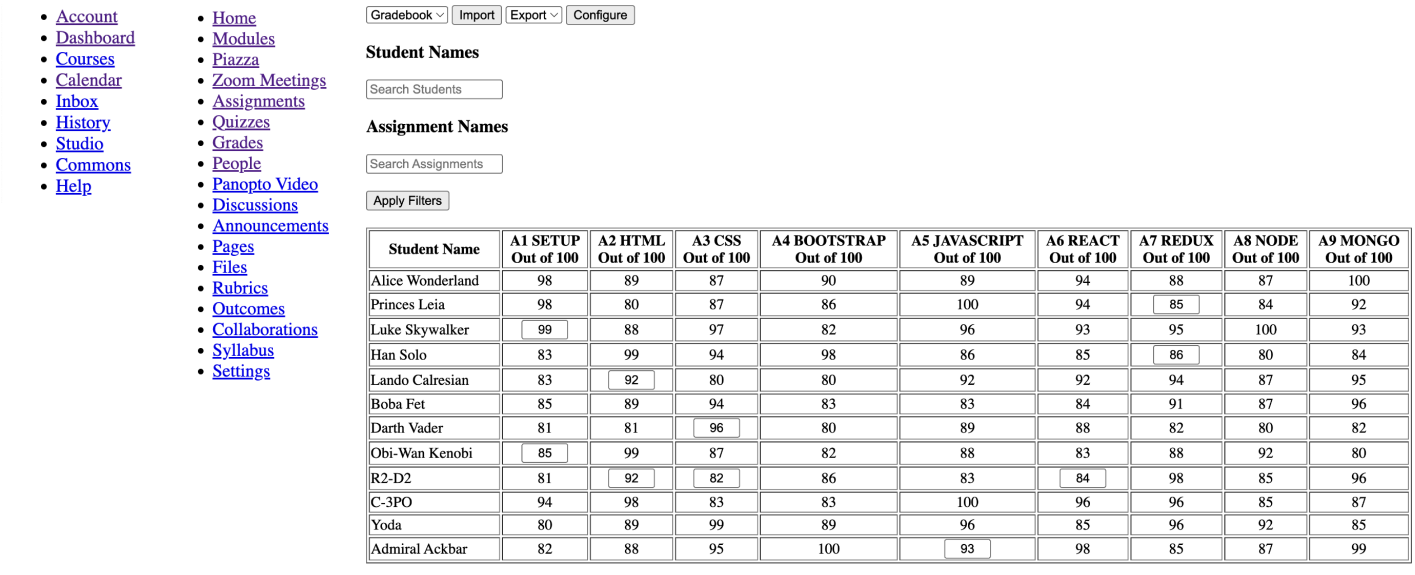
Search Assignments

Apply Filters

Student Name	A1 SETUP Out of 100	A2 HTML Out of 100	A3 CSS Out of 100	A4 BOOTSTRAP Out of 100	A5 JAVASCRIPT Out of 100	A6 REACT Out of 100	A7 REDUX Out of 100	A8 NODE Out of 100	A9 MONGO Out of 100
Alice Wonderland	98	89	87	90	89	94	88	87	100
Princes Leia	98	80	87	86	100	94	85	84	92
Luke Skywalker	99	88	97	82	96	93	95	100	93
Han Solo	83	99	94	98	86	85	86	80	84
Lando Calresian	83	92	80	80	92	92	94	87	95
Boba Fet	85	89	94	83	83	84	91	87	96
Darth Vader	81	81	96	80	89	88	82	80	82
Obi-Wan Kenobi	85	99	87	82	88	83	88	92	80
R2-D2	81	92	82	86	83	84	98	85	96
C-3PO	94	98	83	83	100	96	96	85	87
Yoda	80	89	99	89	96	85	96	92	85
Admiral Ackbar	82	88	95	100	93	98	85	87	99

Figure 3.2.1 - Grades

Implement the grades screen in ***public/Kanbas/Courses/Grades/screen.html*** by combining the content of the ***Kanbas Navigation***, ***Course Navigation*** and ***Grade*** so that it renders as shown in Figure 3.2.1. Use a table with three columns and copy the ***Kanbas Navigation*** into the 1st column, the ***Course Navigation*** into the 2nd column, and the ***Grades*** into the last column. Users can navigate to the grades screen by clicking ***Courses*** on the ***Kanbas Navigation*** sidebar in the first column below. Then they click on ***Grades*** on the ***Course Navigation*** sidebar in the second column. Use HTML tables to layout the columns as shown. Use HTML lists and anchors to render the navigation links as shown. The grades screen renders in the third column below. Feel free to make up your own users/students, assignments, and grades. You only need 4 students and 4 assignments. Some of the grades must be input fields as shown. Use ***select*** element to render the ***Gradebook*** and ***Export*** dropdowns below. Don't worry about options not shown. Make sure the ***Student Names*** and ***Assignment Names*** input fields have the ***placeholder*** shown. The buttons ***Import***, ***Configure***, and ***Apply Filters***, don't need to work yet. Only the links ***Account***, ***Profile***, ***Courses***, ***Home***, ***Assignments***, and ***Grades*** need to work. Configure other links with `href="#"` for now, since we don't yet care where they navigate to. Confirm you can navigate between ***Home*** and ***Grades***.



- Account
- Dashboard
- Courses
- Calendar
- Inbox
- History
- Studio
- Commons
- Help

- Home
- Modules
- Piazza
- Zoom Meetings
- Assignments
- Quizzes
- Grades
- People
- Panopto Video
- Discussions
- Announcements
- Pages
- Files
- Rubrics
- Outcomes
- Collaborations
- Syllabus
- Settings

Gradebook | Import | Export | Configure

Student Names

Search Students

Assignment Names

Search Assignments

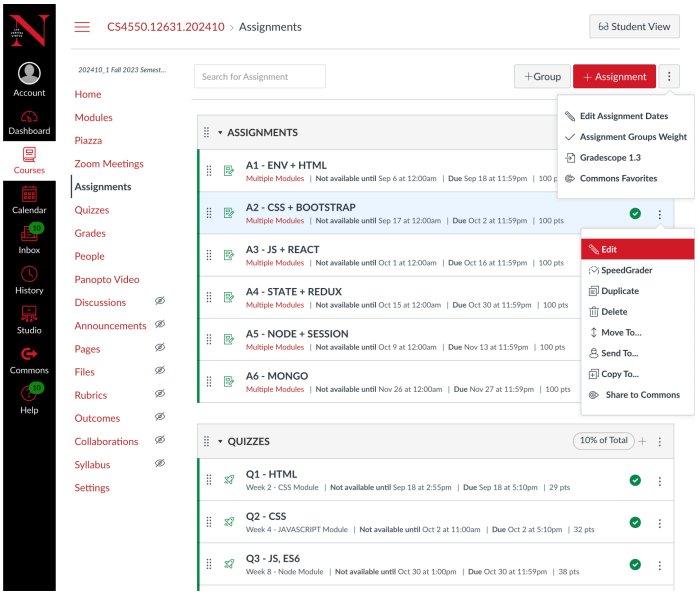
Apply Filters

Student Name	A1 SETUP Out of 100	A2 HTML Out of 100	A3 CSS Out of 100	A4 BOOTSTRAP Out of 100	A5 JAVASCRIPT Out of 100	A6 REACT Out of 100	A7 REDUX Out of 100	A8 NODE Out of 100	A9 MONGO Out of 100
Alice Wonderland	98	89	87	90	89	94	88	87	100
Princes Leia	98	80	87	86	100	94	85	84	92
Luke Skywalker	99	88	97	82	96	93	95	100	93
Han Solo	83	99	94	98	86	85	86	80	84
Lando Calresian	83	92	80	80	92	92	94	87	95
Boba Fet	85	89	94	83	83	84	91	87	96
Darth Vader	81	81	96	80	89	88	82	80	82
Obi-Wan Kenobi	85	99	87	82	88	83	88	92	80
R2-D2	81	92	82	86	83	84	98	85	96
C-3PO	94	98	83	83	100	96	96	85	87
Yoda	80	89	99	89	96	85	96	92	85
Admiral Ackbar	82	88	95	100	93	98	85	87	99

Figure 3.2.2 - Grades Screen

3.3 Assignments screen

The ***Assignments*** screen lists all the assignments, quizzes, exams, and projects students need to complete throughout the semester. An example of the assignments screen is shown here on the right. To navigate to the assignment screen, users click on ***Courses*** on the left ***Kanban Navigation*** sidebar in the first column, then on ***Assignments*** on the ***Course Navigation*** sidebar in the second column. Assignments are grouped into categories such as ***ASSIGNMENTS***, ***QUIZZES***, ***EXAMS***, and ***PROJECT***, and are shown in the third column as shown below. For this assignment, we're just going to focus on capturing the main content and rough layout of the ***Assignments*** screen. In later assignments we'll come back to style this screen to look more like the screen shot shown here. For now, implement the third column of the assignments screen in ***index.html*** in a new folder ***public/Kanbas/Courses/Assignments*** as shown in Figure 3.3.1. Then in a separate file called ***screen.html***, combine the content of ***Assignments/index.html***, the ***Kanbas Navigation*** and ***Course Navigation*** as shown in FIgure 3.3.2. The input fields, buttons, and dropdowns should display as shown, but they don't need to do anything yet. Do make sure that the text fields render the ***placeholders*** as shown. The ***Group*** and ***Assignment*** buttons don't need to do anything yet. The ***Multiple Modules*** link can reference # for now.



Search for Assignments | Group | Assignment | Edit Assignment Dates

ASSIGNMENTS 40% of Total

- A1 - ENV + HTML
  Multiple Modules | Not availab
- A2 - CSS + BOOTSTRAP
  Multiple Modules | Not availab
- A3 - JS + REACT
  Multiple Modules | Not availab
- A4 - STATE + REDUX
  Multiple Modules | Not available yet
- A5 - NODE + SESSION
  Multiple Modules | Not available yet
- A6 - MONGO + MONGOOSE
  Multiple Modules | Not available yet

Edit
Speed Grader
Duplicate
Delete
Move To...
Send To...
Copy To...
Share to Commons

QUIZZES 10% of Total Edit

- Q1 - HTML
  Multiple Modules | Not available yet
- Q2 - CSS
  Multiple Modules | Not available yet
- Q3 - JS, ES6
  Multiple Modules | Not available yet
- Q4 - NODE
  Multiple Modules | Not available yet
- Q5 - MONGO
  Multiple Modules | Not available yet

EXAMS 20% of Total Edit

- Midterm
  Multiple Modules | Not available yet
- Final
  Multiple Modules | Not available yet

PROJECT 30% of Total Edit

- Project
  Multiple Modules | Not available yet

Figure 3.3.1

- Account
- Dashboard
- Courses
- Calendar
- Inbox
- History
- Studio
- Commons
- Help

- Home
- Modules
- Piazza
- Zoom Meetings
- Assignments
- Quizzes
- Grades
- People
- Panopto Video
- Discussions
- Announcements
- Pages
- Files
- Rubrics
- Outcomes
- Collaborations
- Syllabus
- Settings

Search for Assignments | Group | Assignment | Edit Assignment Dates

ASSIGNMENTS 40% of Total

- A1 - ENV + HTML
  Multiple Modules | Not availab
- A2 - CSS + BOOTSTRAP
  Multiple Modules | Not availab
- A3 - JS + REACT
  Multiple Modules | Not availab
- A4 - STATE + REDUX
  Multiple Modules | Not available yet
- A5 - NODE + SESSION
  Multiple Modules | Not available yet
- A6 - MONGO + MONGOOSE
  Multiple Modules | Not available yet

Edit
Speed Grader
Duplicate
Delete
Move To...
Send To...
Copy To...
Share to Commons

QUIZZES 10% of Total Edit

- Q1 - HTML
  Multiple Modules | Not available yet
- Q2 - CSS
  Multiple Modules | Not available yet
- Q3 - JS, ES6
  Multiple Modules | Not available yet
- Q4 - NODE
  Multiple Modules | Not available yet
- Q5 - MONGO
  Multiple Modules | Not available yet

EXAMS 20% of Total Edit

- Midterm
  Multiple Modules | Not available yet
- Final
  Multiple Modules | Not available yet

PROJECT 30% of Total Edit

- Project
  Multiple Modules | Not available yet

Figure 3.3.2

Clicking on any of the assignment link navigates to the ***Assignment Edit*** screen shown in Figure 3.3.3, which allows editing the assignment. For now implement the main content in ***public/Kanbas/Courses/Assignments/Edit/index.html*** and then

combine that content with the ***Kanbas Navigation*** and ***Course Navigation*** into ***public/Kanbas/Courses/Assignments/Edit/screen.html*** as shown in Figure 3.3.4. In later assignments we will come back and style the screen to look more like the screen shot on the left. Use ***input*** fields of type ***date*** to implement the fields ***Due***, ***Available from***, and ***Until***. Use checkboxes to implement fields under ***Online Entry Options***, ***Group Assignments***, and ***Peer Reviews***. Use text input fields for all other fields. Clicking on the ***Cancel*** or ***Save*** buttons should navigate back to the assignment screen.

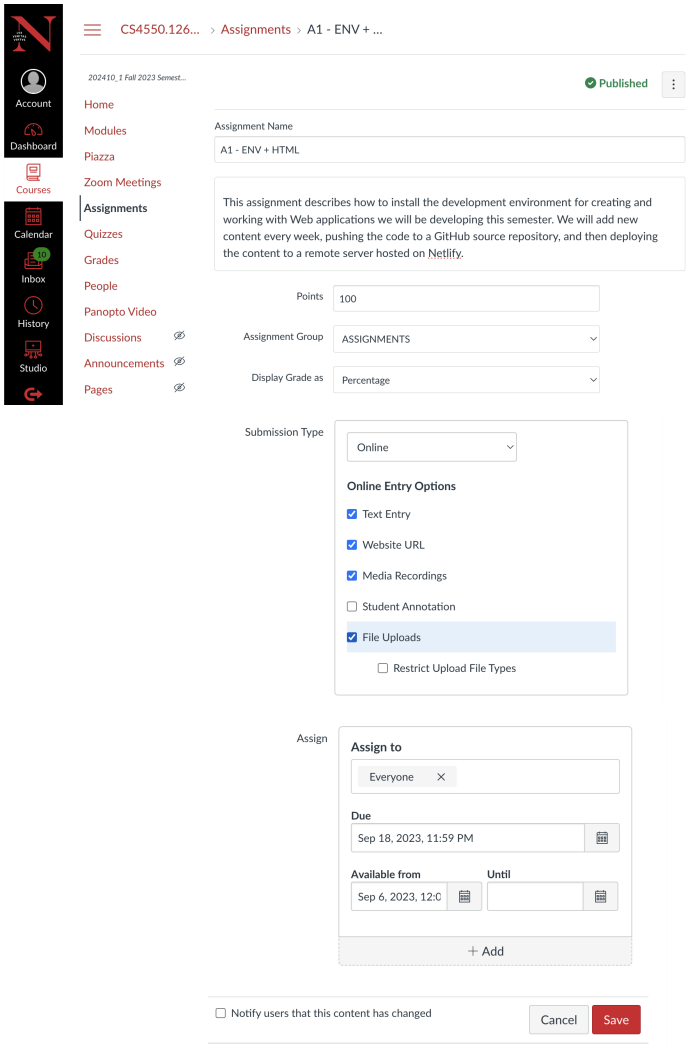


Figure 3.3.3

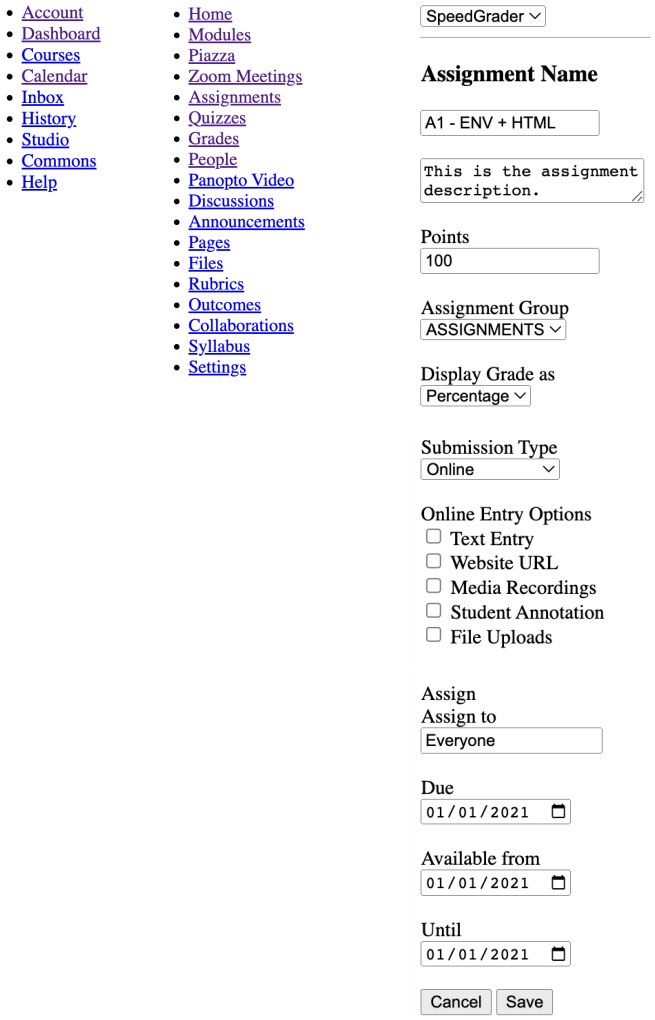
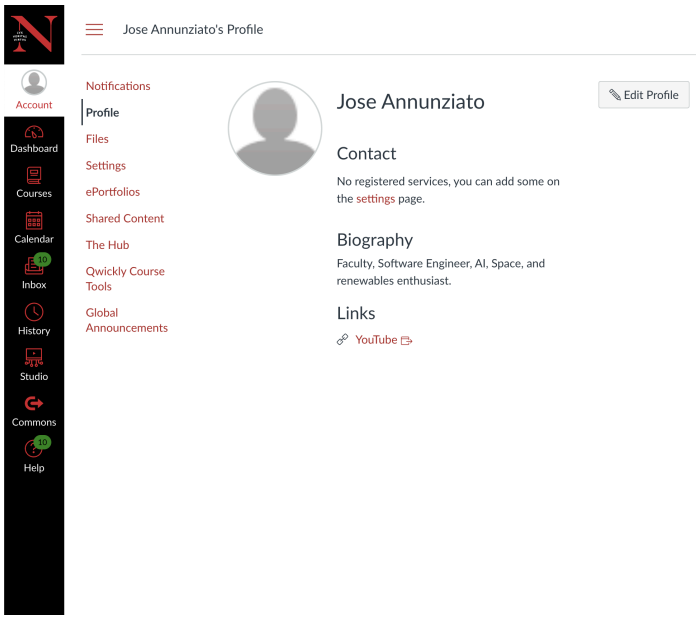
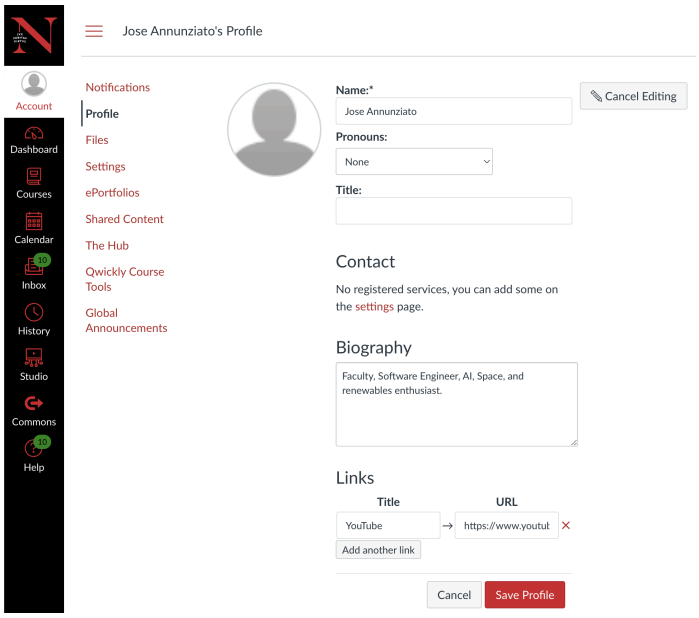


Figure 3.3.4

3.4 Profile screen

The ***Profile*** screen allows users to edit their personal information. To navigate to the profile, users click on ***Account*** and the ***Profile*** screen is shown by default where they can see their ***name***, ***contact***, ***biography***, and ***links*** as shown below on the left. Users can edit their profile by clicking on ***Edit Profile*** which displays the ***Edit Profile*** screen shown below on the right. Users can save their changes by clicking ***Save Profile***, or cancel by clicking on ***Cancel Editing*** which navigates back to the ***Profile*** screen.





For this assignment we'll just implement the content and navigation, but leave the styling to later assignments. In ***/public/Kanbas/Account/Navigation/index.html***, implement ***Account Navigation*** links as shown in Figure 3.4.1. In ***/public/Kanbas/Account/Profile/index.html***, implement the ***Profile*** content shown in Figure 3.4.2 below.

- Notifications
- Profile
- Files
- Settings
- ePortfolios
- Shared Content
- The Hub
- Qwickly Course Tools
- Global Announcements

Figure 3.4.1 - Account Navigation

Jose Annunziato's Profile

Edit Profile

Jose Annunziato

Contact

No registered services, you can add some on the settings page.

Biography

Faculty, Software Engineer, AI, Space, and renewables enthusiast.

Links

YouTube

Figure 3.4.2 - Profile

Then, in ***/public/Kanbas/Account/Profile/screen.html***, combine the ***Profile*** with the ***Kanbas Navigation*** and ***Account Navigation*** to create the ***Profile*** screen as shown in Figure 3.4.3. Clicking ***Profile*** in ***Account Navigation*** and ***Account*** in ***Kanbas Navigation*** should navigate to ***Profile/screen.html***. The other links can just point to # for now.

- Account
- Dashboard
- Courses
- Calendar
- Inbox
- History
- Studio
- Commons
- Help

- Notifications
- Profile
- Files
- Settings
- ePortfolios
- Shared Content
- The Hub
- Qwickly Course Tools
- Global Announcements

Jose Annunziato's Profile

Edit Profile

Jose Annunziato

Contact

No registered services, you can add some on the settings page.

Biography

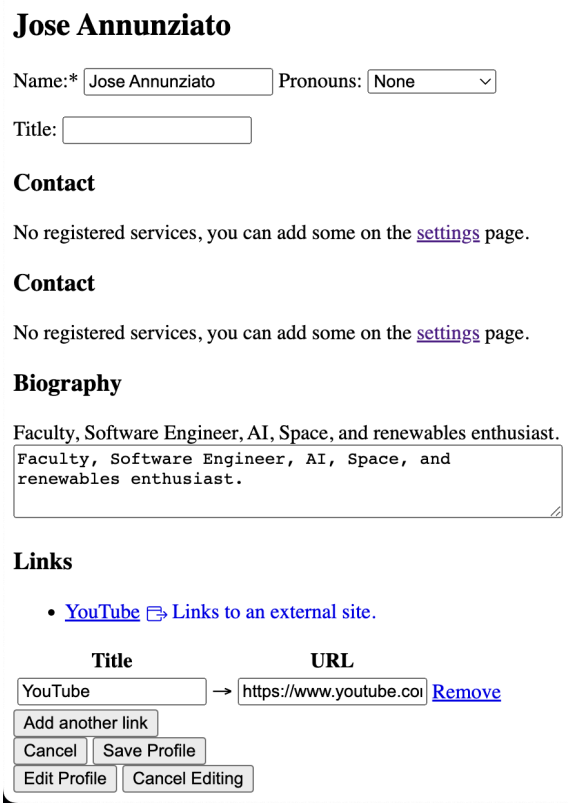
Faculty, Software Engineer, AI, Space, and renewables enthusiast.

Links

YouTube

Figure 3.4.3 - Profile Screen

Now lets focus on implementing the ***Edit Profile*** screen. In ***/public/Kanbas/Account/Profile/Edit/index.html***, implement the form shown here on the right. In the same directory, in ***/public/Kanbas/Account/Profile/Edit/screen.html***, combine the form in ***index.html*** with the ***Kanbas Navigation*** and ***Account Navigation*** to create the ***Edit Profile*** screen. Clicking the ***Edit Profile*** button should navigate to ***Edit/screen.html***. Clicking ***Save Profile*** or ***Cancel Editing*** navigates back to ***Profile/screen.html***. Use HTML tables to layout the columns as shown. Use HTML lists and anchors to render the navigation links as shown. Only the links ***Account***, ***Profile***, ***Courses***, ***Home***, ***Assignments***, and ***Grades*** need to work. Configure other links with `href="#"` for now, since we don't yet care where they navigate to. Feel free to use your own username, first name, last name, contact, biography, title, URL, etc.



3.5 Exercises

Implement the ***Kanbas*** Web pages described in section 3 listed below

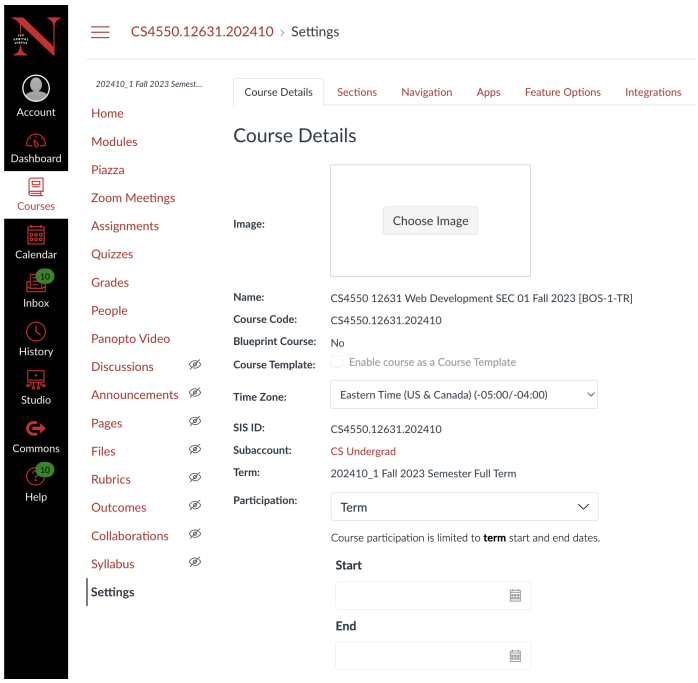
1. Implement Home screen as described in section 3.1
2. Implement Grades screen as described in section 3.2
3. Implement the Assignment and Edit Assignment screens as described in section 3.3
4. Implement the profile and edit profile screens as described in section 3.4

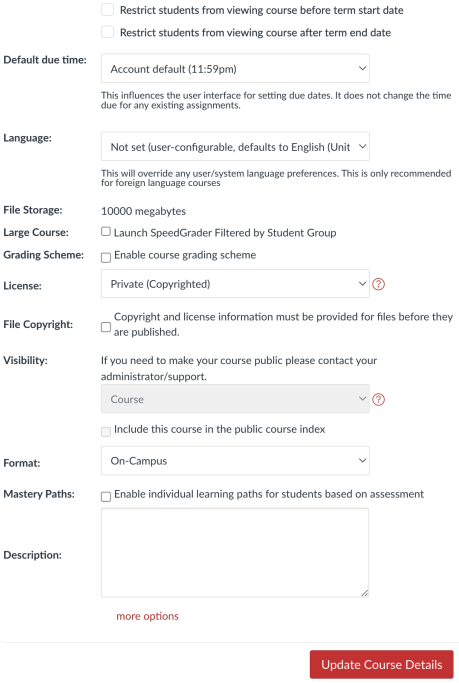
4 Graduate Students Only

Graduate students are required to do additional work to complete this assignment and the course overall. Implement the ***Settings*** screen allowing faculty to configure the behavior and organization of courses. Users can navigate to the ***Settings*** screen by clicking on ***Courses*** in the ***Kanbas Navigation*** sidebar in the first column, and then on ***Settings*** in the ***Course Navigation*** in the second column. The ***Settings*** screen provides several tabs to navigate to various settings screens. In this assignment, we'll focus on the ***Course Details*** and ***Navigation*** tabs under the ***Settings*** screens. Let's first tackle the ***Course Details*** tab.

4.1 Course Details Tab

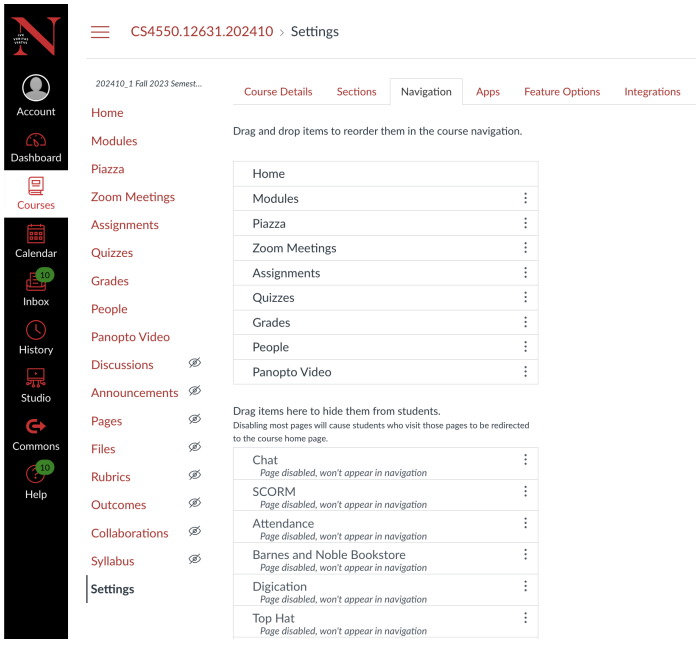
The ***Course Details Tab*** allows faculty to configure several course attributes such as the ***time zone***, ***course start date***, and ***course end date***. Screenshots of the tab is shown below split into two columns since it was too long. Implement the column layout using tables as described in earlier exercises. Implement the ***Kanbas Navigation*** and ***Course Navigation*** links as lists of anchors as described in earlier exercises. Implement the dropdowns, input fields, checkboxes, date fields, as described in earlier exercises. The resulting screen doesn't need to look like the screenshots, but make sure to capture the main content and rough layout. Later assignments will revisit this screen and use CSS to style the screens to look more like the screenshots.

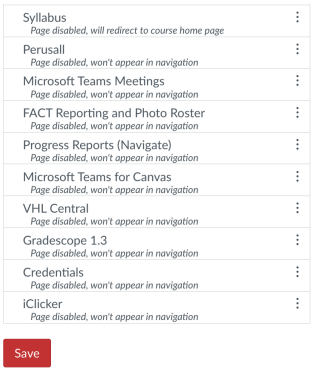




4.2 Course Navigation Tab

The ***Navigation Tab*** allows faculty to configure the navigation of the course. Screenshots of the tab is shown below split into two columns since it was too long. Implement the column layout using tables as described in earlier exercises. Implement the ***Kanbas Navigation*** and ***Course Navigation*** links as lists of anchors as described in earlier exercises. Later assignments will revisit this screen and use CSS to style the screens to look more like the screenshots.





5 Source Control

5.1 Install a git client

On macOS you already get a command line git client. You can fully interact with ***github.com*** from the terminal. On Windows OS, you'll need to install a git client from where you will be able to issue the same commands from a console. Download git for windows from https://git-scm.com/download/win, run the installer and follow the instructions. At the end of the installation you should be able to execute git commands from new console instances. All examples in this course assumes you have git installed in your OS.

You can also install a graphical git client if you prefer. There are a lot of alternatives, but the author prefers Sourcetree since it works well and consistently in both macOS and Windows. To install Sourcetree download from https://www.sourcetreeapp.com/, install, and follow instructions. We will not be covering how to use Sourcetree, but you are free to use it if you wish. All examples in this course will be using the command line git client.

5.2 Ignoring files and directories

Git can keep track of all the files in your project, but there are some files or directories that you don't want to keep track of, for instance, compiled classes, libraries, etc. You can configure which files and directories you want git to ignore by listing them in a file called ***.gitignore*** at the root of your project, which should already exist. Create the file if it does not already exist. With a text editor or from IntelliJ, edit the file ***.gitignore***. Towards the top of the file, in a blank line, type the following. ***Note*** the period at the beginning of the file!!

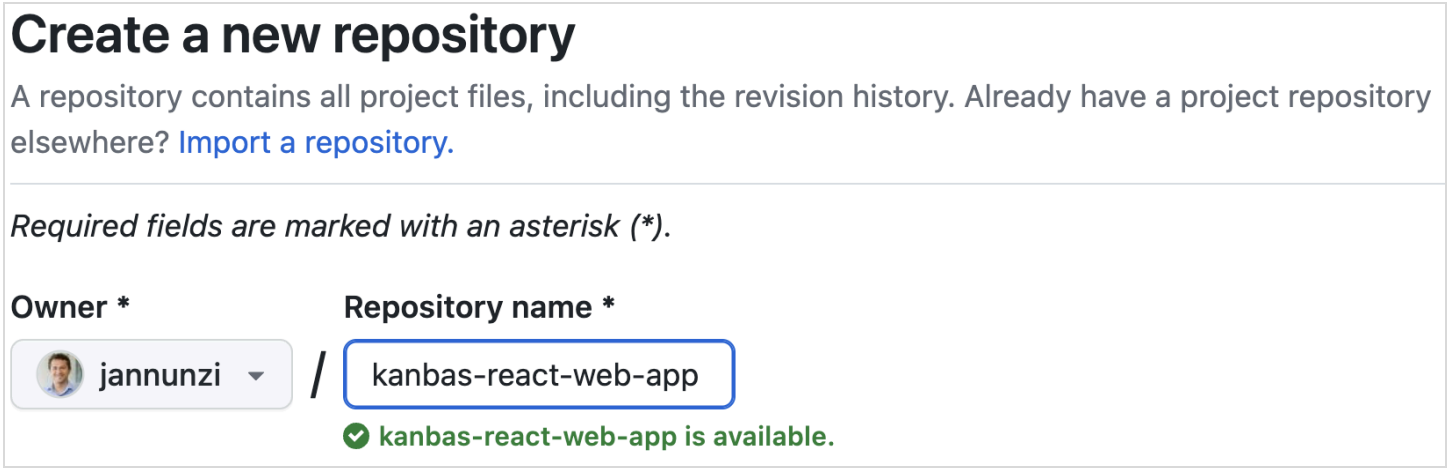
.gitignore

```
.idea
node_modules
```

This tells git that the ***.idea*** folder should be ignored because it is a directory specific to IntelliJ and not relevant to React.js project itself. If you are using other IDEs, then you might want to add other files or directories here relevant to your specific IDE. The ***node_modules*** folder should also be ignored since it contains library files that should not be added to source control. ***Note***: make sure that IDE specific folders and the ***node_modules*** folders don't make it into the repository!!

5.3 Create a remote source repository

If you don't already have an account on ***github.com***, create a new account or use an account you already have at ***github.com***. Do not use the university's github or your work's source control if you already have one. Login to your ***github.com*** account and click the ***New*** button to create a new ***public repository*** called ***kanbas-react-web-app***, just like the name of the React.js project you created earlier. Here's an example on how it looks on my laptop. Your username "***jannunzi***" will obviously be different.



Once you create the remote repository on github, it will display commands on how to commit and push your code from your computer up to the remote repository. The commands will be similar to the ones shown below. The username "***jannunzi***" will obviously be different.

5.4 Adding and committing your code

Now that we have a remote source repository, let's add, commit and push our code to the repository. From the terminal or console, make sure you are in the ***kanbas-react-web-app*** directory and then type the following commands which are based on the commands git suggested. Ignore the commentary on the right. Also, your actual URL below will differ for you, so ***don't blindly copy the example below***. Use the commands git suggested for you.

```
$ git  init                                                          # initializes local repository
$ git  add  .                                                        # adds all files to repository
$ git  commit  -m   "first commit"                                   # commits files with message
$ git  remote  add  origin https://github.com/jannunzi/kanbas-react-web-app.git       # adds remote repository
$ git  push    -u   origin main                                      # copies local files to remote
```

Refresh the remote repository on github.com and confirm that the files are now available there

5.5 Using personal access tokens

While pushing to GitHub you might encounter an error stating that ***password authentication was removed on August 13, 2021***. One way to fix this is to generate a ***personal access token*** and use that instead of your password. To generate a personal access token go to your GitHub ***Settings***, and then ***Developer Settings***. Click on ***Personal access tokens*** and then on ***Generate new token***. Enter a short description in the ***Note*** field, select ***No expiration*** for ***Expiration***, and grant all access privileges by selecting all the checkboxes under ***Select scopes***. You are welcome to be more restrictive it you want. Click on ***Generate token*** and copy the long unique access token to your clipboard. Note that this will be the only opportunity to copy the token and you fail to do so you'll have to delete this token, create a brand new one, and try again. With the token copied to the clipboard, try pushing again to GitHub, but this time paste the token when asked for a password.

If you are not being asked for a password then GitHub might be using a cached authentication. To clear cached GitHub authentications on Windows go to the ***Credential Manager***, click on ***Windows Credentials***, click the GitHub credentials, and click ***Remove***. This time github should ask for your username and password again when trying to push. Paste the access token when asked for the password. To clear cached authentications on macOS, go to your ***Key Chain*** and search for github. Remove the GitHub key chain.

6 Deploying to a remote server

Create an account at https://www.netlify.com/ if you don't already have one. Follow the instructions to create a team or organization and then navigate to the ***Team overview*** screen. On the ***Team overview*** or ***Sites*** page, click on ***Create/Add new site***. Select **"Import an existing project".**

- Connect to **Git provider**, select **Github.** Give **Netlify** all the authorizations it asks for. **Pick a repository**, in the **Search repos** field. **Lookup and pick the kanbas-react-web-app** repository you created earlier
- In the **Site settings, and deploy** tab, select the branch to deploy from, e.g., **master or main**, and click **Deploy site.** The deployment process might take a few minutes. Netlify will pick a random silly name for your application, e.g., **loving-torvalds-effde8.** That's fine for now, we can setup custom domain names later. While your project deploys it will show the main steps it's going through and will eventually say **Published** in green half way down the screen. **You can then click on the URL towards the top, e.g., https://loving-torvalds-effde8.netlify.app and see your project deployed.**

In the ***Deployment settings***, disable ***Continuous Integration*** so your deployment is not rejected by trivial warnings. From your Netlify dashboard, click on ***Deployment settings***, then ***Build & deploy***, and then ***Continuous deployment***. Scroll down to ***Build configuration*** and click on ***Configure***. Change the ***Build command*** from `npm run build` to `CI=false npm run build`.

Also enable all branches to deploy in separate URLs so that TAs can grade each assignment individually while allowing you to continue working on the next assignment. Scroll further down to ***Branches and deploy contexts*** and click ***Configure***. In the section ***Branch deploys*** select the radio button ***All***. Now all the branches pushed to the repository will be available in separate URLs.

7 Deliverables

1. Download and install the latest ***Node.js*** as described in section 2.1
2. Create a React.js application called ***kanbas-react-web-app*** as described in section 2.2
3. Push the source code of the React.js application ***kanbas-react-web-app*** to a remote source repository in ***GitHub.com*** as described in section 4
4. Deploy the ***kanbas-react-web-app*** React.js application to ***Netlify*** as described in section 5

5. As a deliverable in ***Canvas***, submit the following URLs
    a. ***(50%)*** The source repository in ***GitHub.com***. It should look something like **`https://github.com/jannunzi/kanbas-react-web-app-1234-df25`**
    b. ***(50%)*** The React.js application running on ***Netlify***. It should look something like **`https://roaring-hummingbird-1d48d4.netlify.app`**