Single Page Applications with React

Rendering dynamic Webpages with React

Introduction

A previous assignment described using HTML and CSS to build a static web site as a prototype of a course manager application called WhiteBoard. This assignment expands on that by making the application a dynamic Single Page Application (SPA). Feel free to reuse the HTML and CSS used in previous assignments. The SPA will use the JavaScript library React to dynamically render the user interface in terms of reusable React components.

Learning objectives

The learning objectives of this assignment are

- Implementing single page applications
- Implementing component based web applications
- Using the react JavaScript library to build single page web applications
- Using the react routing library to implement web application navigation

Install Node.js

Before you get started in this assignment, you'll need to install Node.js. Head over to https://nodejs.org and downloadand install Node.js for your operating system. When done, restart your machine. To test the installation create a JavaScript file in your file system, e.g., hello.js might contain the following:

```
console.log("Hello World")
```

Run your new JavaScript script from the command line as following

```
$ node hello.js
```

Confirm that the script prints "Hello World" at the commandline.

Create a react project

Use the ***create-react-app*** command line tool to create a React application called whiteboard. Append modifiers to the name of the application based on the term, section and your name. For instance, consider the following React command line:

```
$ npx   create-react-app  wbdv-fa45-charlie-brown-client-react
$ cd    wbdv-fa45-charlie-brown-client-react
$ npm   start
```

The above project might be for someone called Charlie Brown enrolled in a fall semester in the year 2345.

Install Bootstrap, Fontawesome and React Router

Install bootstrap and fontawesome from the root of the application as follows. All commands are from root or project

```
$ npm install bootstrap --save
$ npm install @fortawesome/fontawesome-free --save
$ npm install react-router-dom --save
```

Import bootstrap and fontawesome into **src/index.js** right before **index.css** as follows

```
import React from 'react';
import ReactDOM from 'react-dom';
import 'bootstrap/dist/css/bootstrap.min.css'
import '@fortawesome/fontawesome-free/css/all.min.css'
import './index.css';
```

Implement a course service

In a file called ***course-service.js***, in directory called ***src/services*** create a service called ***CourseService***. The service should provide an API to create, retrieve, update and delete course instances as shown below

Function	Description
createCourse(course)	creates a new course instance and adds it to the collection of courses
findAllCourses()	retrieves all course instances as an array of courses
findCourseById(id)	retrieves a course instance that matches the id parameter. Implement this function even if you don't need/use it
updateCourse(id, course)	updates the course instance whose id matches the id parameter. Updates the instance with values in course parameter
deleteCourse(id)	deletes course instance whose id matches the id parameter

To implement the CRUD operations, the service must use the generic server introduced in class as used in the previous assignment. Use the URL pattern
https://wbdv-generic-server.herokuapp.com/api/YOURNEUID/courses

Display a set of courses

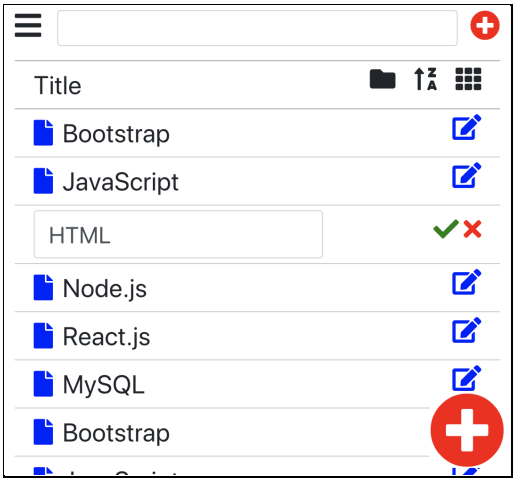
A prototype built in an earlier assignment displayed courses as a table of courses with one course per row. In this assignment you will create a grid view that displays the samce courses but laid out in a grid. A button at the top of the screen will toggle between the two views. Create React components CourseTable and CourseGrid that render the courses as tables and as grids. Implement the toggle between the two components. Feel free to reuse HTML templates and CSS from previous assignments.

Implement the CourseManager component

Create a react component called ***CourseManager*** that implements a course manager that allows managing a collection of courses including adding new courses, deleting courses and editing existing courses. The course manager should maintain the current list of courses and implement event handlers. The course manager should also be used to manage navigation between the courses rendered as a table or rendered as a grid.

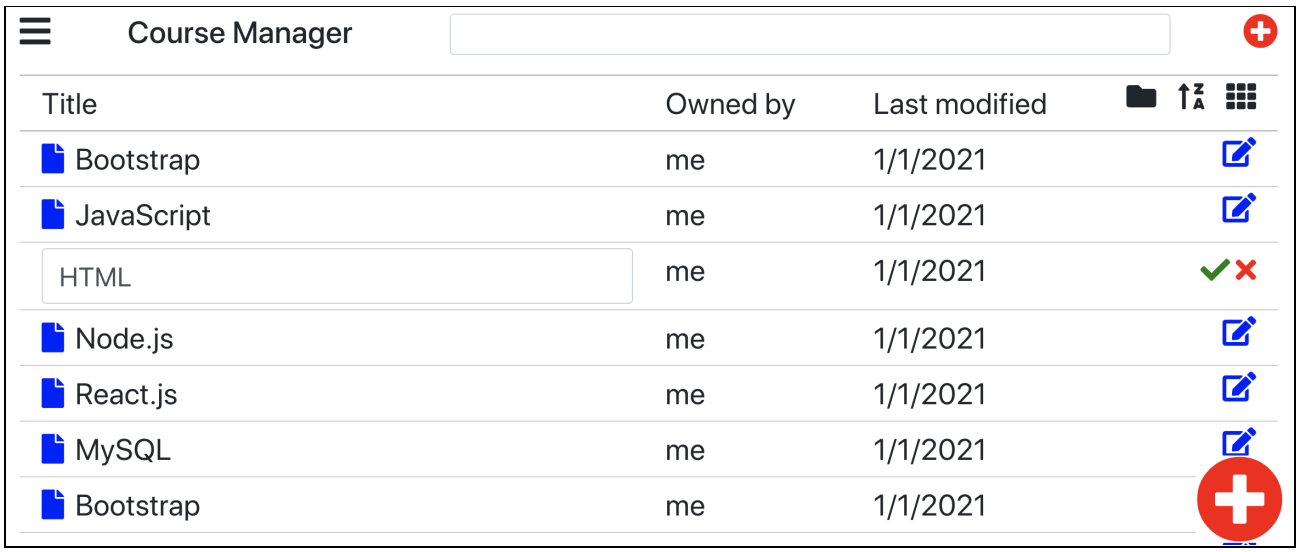
Implement the CourseTable component

In a file called ***course-table.js*** in directory called ***components/course-table/course-table***, create a React component called ***CourseTable*** that renders an array of courses as a table where each row represents a course as shown below. Each row that renders a course must be implemented in a separate React component ***CourseRow*** implemented in a file called ***course-row.js*** in the same directory. The ***CourseTable*** component should be as responsive as described in earlier assignments and illustrated below. Implement the ***CourseTable*** as a stateful class. If necessary, use Postman to create some initial courses. Clicking on the course layout button toggles between the table layout shown below and the grid layout described later in this document. The bars on the left, the folder and sort icons don't do anything yet. Clicking on the red plus icons adds a new course with the title in the input box at the top of the component.



Extra Small Screens ("xs")

Medium Screens ("md")

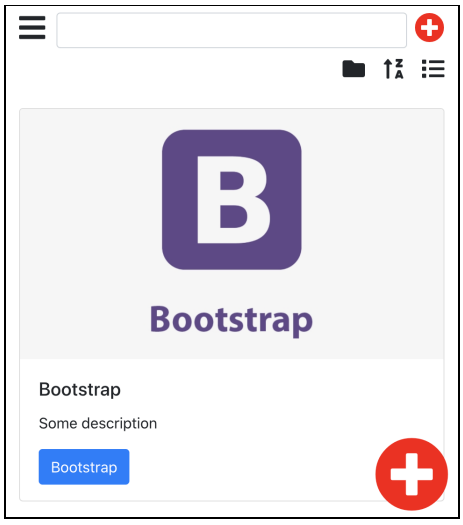


Large Screens ("lg")

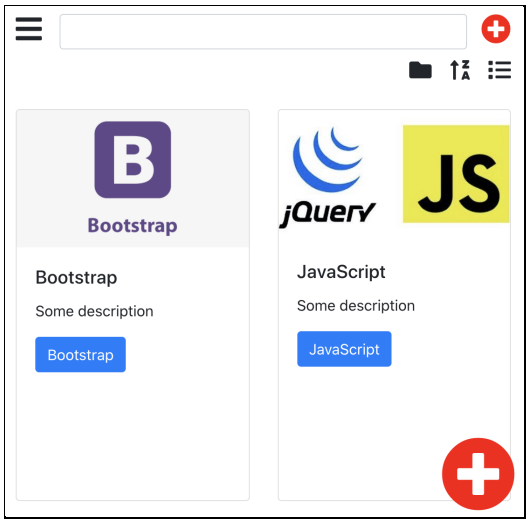
Implement the CourseGrid component

In a file called ***course-grid.js*** in directory called ***components/course-grid/course-grid***, create a React JS component called ***CourseGrid*** that renders an array of courses as a grid. The component should render an

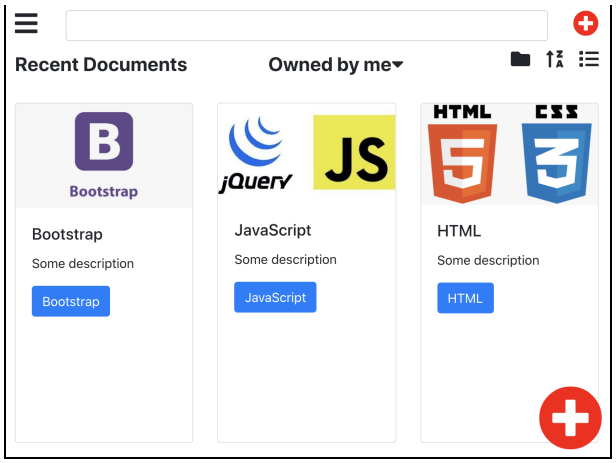
array of courses as a grid of cards where each card represents a course as shown below. Each card that renders a course must be implemented in a separate React component called ***CourseCard***. The ***CourseGrid*** component should be responsive so that the number of grid columns should respond to the size of the screen as shown below. Clicking on the course layout button toggles between the grid layout shown below and the table layout described earlier. Implement the ***CourseGrid*** and ***CourseCard*** as functions, not as classes.



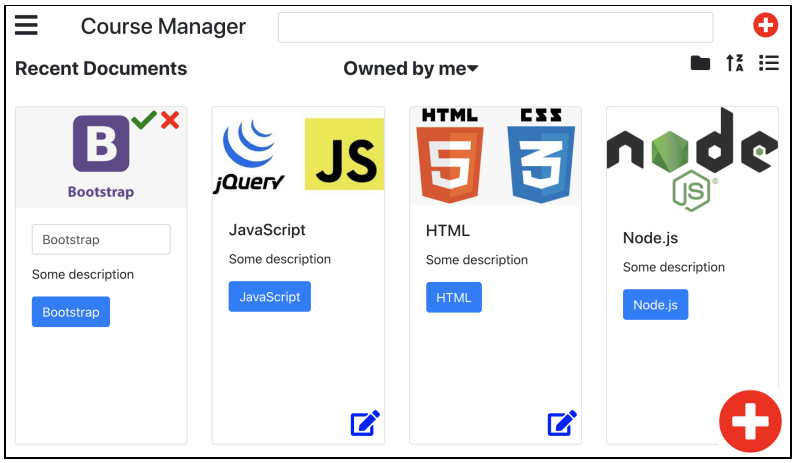
Extra Small Screens ("xs")



Small Screens ("sm")

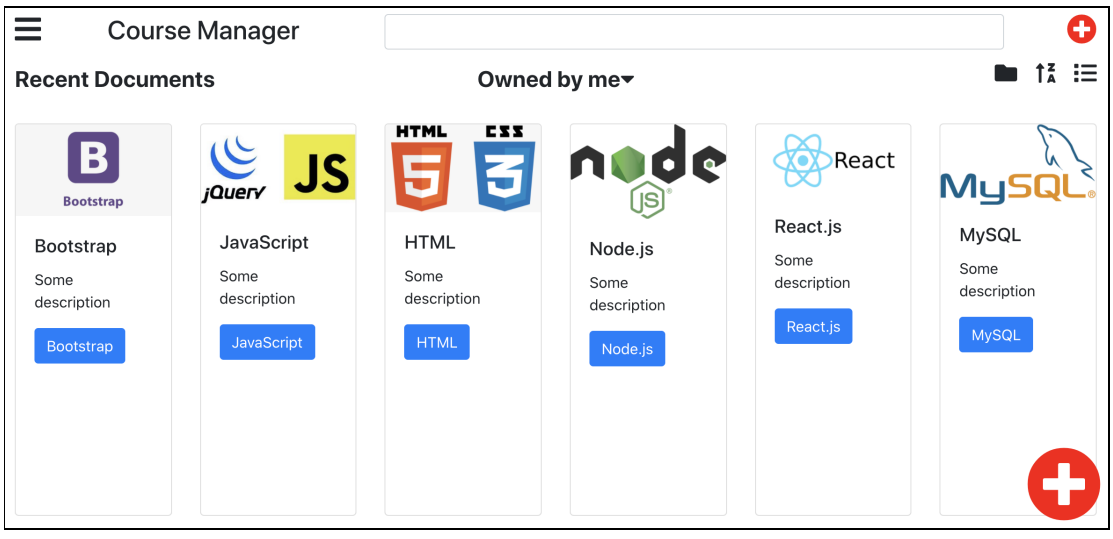


Medium Screens ("md")



Large Screens ("lg")

Note: that the pencil, checkmark and X icons illustrated in the grid in the ***Large Screens*** layout applies to all other layouts as well, e.g., ***Extra Small Screens***, ***Small Screens***, ***Medium Screens***, and ***Extra Large Screens***



Extra Large Screens ("xl")

Course row and card

Both the CourseRow and CourseCard components should have the following buttons and behaviors

- Clicking on the delete course icon (red x) deletes the corresponding course
- Clicking on the edit course icon (pencil) changes the row into edit mode replacing the course title with an input field, hides the pencil and showsthe delete and save icons
- Clicking on the save course icon (checkmark) saves the course title permanently and exits the edit mode replacing the input field with the title and replaces the delete and save icons with the edit icon
- Clicking on the course title or the course button navigates to the course editor implemented in earlier assignments

Course editor

In a file called ***course-editor.js*** in a directory called ***components/course-editor/course-editor***, create a React component called ***CourseEditor*** as a stateless function. Use the content of the body of the course editor HTML document created in earlier assignments to implement the return or render methods of the ***CourseEditor*** component. The CourseEditor can be static for now and will be made dynamic in a later assignemnt. Add an icon near the top left of the component that allows navigating back to the course manager.

Deliverables

As a deliverable, check in all your work into a github source control repository. Use a brand new github repository different from the one you have been using in previous assignments. Deploy your project to a remote public server on Heroku or AWS. Deploy to a brand new remote server different from the previous server you've used from previous assignments. Submit the URL of the repository and the remote server in blackboard. Tag the commit you want graded as assignment3. TAs will clone your repository and grade the tagged commit.

To deploy to Heroku will require adding an additional buildpack:
https://buildpack-registry.s3.amazonaws.com/buildpacks/mars/create-react-app.tgz