05. React 3

Due Oct 30, 2019 by 11:59pm **Points** 6 **Submitting** a text entry box
Available Oct 10, 2019 at 3pm - Nov 9, 2019 at 11:59pm about 1 month

This assignment was locked Nov 9, 2019 at 11:59pm.

Homework: React 3 (6 Points Total)

This assignment is meant to facilitate the design of of React, e.g. states, props, more modularized components etc. This project is the final part of the React project. We recommend starting this assignment as soon as possible, and not waiting until the deadline!

Unlike the previous assignment, you can make a choice in which way you want to add features to the application. There are two main branches that you can proceed on, the **Recommender** and **Planner**.

Also unlike the previous assignment, you will be building off your code for React 2. The Assignment Repository where you will submit your code is empty with the exception of a README. You should copy the contents of your React 2 repository (with the exception of the `.git`) into the React3 repository.

Submission Details:

To submit the assignment, you will need to do two things.

1. Make sure that you commit your final version to the personal repository that gets created for you in GitHub classrooms. The last commit you make before the deadline is the one that will be graded.
2. In the homework submission, type the ***NAME OF YOUR REPOSITORY*** in the text field. Also, indicate the version of the assignment you are doing, either **Planner** or **Recommender**.
3. If you want to use a late day, ***indicate this in the text field as well***. Otherwise, the last commit before the deadline in the GitHub repository is the one that will be graded.

Course Data:

The course data is being fetched from `https://mysqlcs639.cs.wisc.edu/classes/` and is formatted as follows:

```
{
  <alpha-numeric key of length 3>: {
    "credits": <number of credits for the course>,
    "description": <course description>,
    "keywords": <1D list of string keywords>,
    "name": <course name>,
    "number": <course number>,
    "requisites": <2D list of course requisites>,
    "sections": {
      <section number>: {
        "instructor": <instructor name>,
        "location": <section location>,
        "subsections": {
          <subsection number>: {
            "location": <subsection location>,
            "time": {
              <weekday>: <time range>, ...
            }
          }, ...
        },
        "time": {
          <weekday>: <time range>, ...
        }
      }, ...
    },
    "subject": <course subject>
  }
}
```

- The list of course requisites consists of 1D lists with AND operations between them. Each 1D list has OR operations between elements. For example: `[[A, B], [C, D, E], [F]]` means that the requisites are `(A OR B) AND (C OR D OR E) AND (F)`. The requisites will be represented as the course's alpha-numeric key used in the outermost object.
- A course can have any number of sections, and each section can have any number of subsections. If the course contains subsections, the user must schedule exactly one subsection with its parent section.
- Sections and subsections can have any number of times. Each time's key is a weekday in all lowercase ("monday", "tuesday", "wednesday", ...). Each time's value is a string with the following format: `"<12 hour time><am or pm> - <12 hour time><am or pm>"`. An example of this would be `"11:45am - 12:35pm"`.
- Each course has exactly one subject

NOTE:

> Your project must be able to accept any data with the same format as above and the data located at https://mysqlcs639.cs.wisc.edu/classes/. Additionally, you should be able to handle edge cases, such as missing fields (for example, if a class does not have subsections, etc).
> Do not assume that the exact data will be used to evaluate your code, however, any testing code will adhere to these specifications.

Recommender

Problem 1 (1.25 points)

1. Fetch data from server: https://mysqlcs639.cs.wisc.edu/students/5022025924/classes/completed
2. Create a new component to represent previously taken courses. This component will look somewhat like the Course component, but it will

Submission

✓ **Submitted!**
Oct 30, 2019 at 11:59pm
Submission Details

Grade: 5.75 (6 pts possible)
Graded Anonymously: no
View Rubric Evaluation

Comments:
No Comments

be simpler and won't have options to add the course to the cart.
3. Create a new component to hold the previously taken courses. Make this component accessible from the app.

Problem 2 (1.25 points)

1. Create a component for rating a specific course.
2. Allow the user to rate courses they have already taken.

Problem 3 (1.25 points)

1. Generate a list of interest areas based on the course data (maybe look at subjects and keywords?)
2. Create a component for the user to select interest areas as defined in step 1.
3. Make this component available to the user.

Problem 4 (1.25 points)

1. Create the recommender algorithm that takes in the rated courses and interest areas. Search through subjects and keywords to find the courses most similar to the highly rated courses and the courses that match the most interest areas
2. Display the recommended courses to the user. (Maybe click a button in search to pop up a modal with a few recommended courses or sort all courses by their recommendation score)

General (1 point)

Projects will be graded on the general usability and design of the system. You should consider the way the users navigate around the application, and other concepts covered in design lecture.

Planner

Problem 1 (2 points)

1. Create a new planner view
2. Create a sidebar component for this planner view that displays all of the courses, sections, and subsections in the cart with check boxes next to each one
3. Store the data of what is checked to make the check boxes related (if a course is checked, all of its sections and subsections will also be checked)

Problem 2 (1 point)

1. Create a function to generate all of the possible schedules based on the data from planner's sidebar

Problem 3 (2 points)

1. With the provided component, create a schedule component that displays a generated schedule
2. Create two buttons to switch between schedules by changing the data that is sent to the Schedule component

Pseudocode for schedule generation

```
courses = a list of all of the selected courses in the same form
          as the server data, with only the checked sections
          and subsections included.

schedules = the list of generated schedules = []

timeBlocks = []

for(course in courses):
    courseTimeBlocks = []
    for(section in course):
        if(section has no subsections):
            courseTimeBlocks.push([section])
        else:
            for(subsection in section):
                courseTimeBlocks.push([section, subsection])
    timeBlocks.push(courseTimeBlocks)
```

Now, timeBlocks contains a list for each course in which one of the section-subsection pairs must be selected to schedule. Iterate through all of these possibilities and add to schedules if there are no conflicts

General (1 point)

Projects will be graded on the general usability and design of the system. You should consider the way the users navigate around the application, and other concepts covered in design lecture.

Recommender/Planner Rubric (1)				
Criteria	**Ratings**			**Pts**
R: Problem 1: Fetch Data from Server Fetch data from server	**0.25 pts** **Full Marks** Able to fetch data from server		**0.0 pts** **No Marks**	0.25 pts
R: Problem 1: Previous Courses and Previous Courses Area components Create a new component to represent previously taken courses. This component will look	**1.0 pts** **Full Marks**	**0.5 pts** **Half Marks**	**0.0 pts** **No**	

somewhat like the Course component, but it will be simpler and won't have options to add the course to the cart. Create a new component to hold the previously taken courses. Make this component accessible from the app.	Both components created		one component created/correct	**Marks**	1.0 pts
R: Problem 2: Component for Rating Create a component for rating a specific course.	**0.25 pts** **Full Marks**		**0.0 pts** **No Marks**		0.25 pts
R: Problem 2: Rate Classes Allow the user to rate courses they have already taken.	**1.0 pts** **Full Marks** Able to rate classes	**0.75 pts** **3/4 Marks**	**0.5 pts** **Half Marks**	**0.25 pts** **1/4 Marks**	**0.0 pts** **No Marks**	1.0 pts
R: Problem 3: List of Interest Areas Generate a list of interest areas based on the course data	**0.5 pts** **Full Marks**		**0.0 pts** **No Marks**		0.5 pts
R: Problem 3: Interest Component Create a component for the user to select interest areas as defined in step 1. Make this component available to the user.	**0.75 pts** **Full Marks**	**0.5 pts** **2/3 Marks**	**0.25 pts** **1/3 Marks**	**0.0 pts** **No Marks**	0.75 pts
R:Problem 4: Recommender Algorithm Create the recommender algorithm that takes in the rated courses and interest areas. Search through subjects and keywords to find the courses most similar to the highly rated courses and the courses that match the most interest areas	**0.75 pts** **Full Marks**	**0.5 pts** **2/3 Marks**	**0.25 pts** **1/3 Marks**	**0.0 pts** **No Marks**	0.75 pts
R:Problem 4: Display Recommended Courses Display the recommended courses to the user.	**0.5 pts** **Full Marks**		**0.25 pts** **Half Marks**	**0.0 pts** **No Marks**	0.5 pts
R: General Projects will be graded on the general usability and design of the system.	**1.0 pts** **Full Marks**	**0.75 pts** **3/4 Marks**	**0.5 pts** **Half Marks**	**0.25 pts** **1/4 Marks**	**0.0 pts** **No Marks**	1.0 pts
P: Problem 1: Planner View Create a new planner view	**1.0 pts** **Full Marks**	**0.75 pts** **3/4 Marks**	**0.5 pts** **Half Marks**	**0.25 pts** **1/4 Marks**	**0.0 pts** **No Marks**	1.0 pts
P: Problem 1: Sidebar Create a sidebar component for this planner view that displays all of the courses, sections, and subsections in the cart with check boxes next to each one	**0.5 pts** **Full Marks**		**0.25 pts** **Half marks**	**0.0 pts** **No Marks**	0.5 pts
P: Problem 1: Data Storage Store the data of what is checked to make the check boxes related (if a course is checked, all of its sections and subsections will also be checked)	**0.5 pts** **Full Marks**		**0.25 pts** **Half Marks**	**0.0 pts** **No Marks**	0.5 pts
P: Problem 2: Generate Schedules Create a function to generate all of the possible schedules based on the data from planner's sidebar	**1.0 pts** **Full Marks**	**0.75 pts** **3/4 Marks**	**0.5 pts** **Half Marks**	**0.25 pts** **1/4 Marks**	**0.0 pts** **No Marks**	1.0 pts
P: Problem 3: Display Schedule With the provided component, create a schedule component that displays a generated schedule	**1.0 pts** **Full Marks**	**0.75 pts** **3/4 Marks**	**0.5 pts** **Half Marks**	**0.25 pts** **1/4 Marks**	**0.0 pts** **No Marks**	1.0 pts
P: Problem 3: Switch between Schedules Create two buttons to switch between schedules by changing the data that is sent to the Schedule component	**1.0 pts** **Full Marks**	**0.75 pts** **3/4 Marks**	**0.5 pts** **Half Marks**	**0.25 pts** **1/4 Marks**	**0.0 pts** **No Marks**	1.0 pts
P: General Projects will be graded on the general usability and design of the system.	**1.0 pts** **Full Marks**	**0.75 pts** **3/4 Marks**	**0.5 pts** **Half Marks**	**0.25 pts** **1/4 Marks**	**0.0 pts** **No Marks**	1.0 pts
					Total Points: 12.0	