CIS 550: Database and Information Systems

Homework 2 MS 2: Developing a React Client

(60 points)

Please read this handout from start to finish before proceeding to work on the assignment

Introduction

In MS1, you implemented an API according to a specification using Node.js. Now that you are familiar with the two 'lowest' tiers of the architecture (the API server and database), you will use that API to create a frontend application using React.js. Specifically, you will develop an interactive multi-page client using UI component libraries like Ant Design, Shards React, and React-Viz.

This assignment primarily serves as a refresher on front-end web development and by the end of the assignment, you can hope to have a great template that you can refer to, modify, and use for the project.

This milestone is developed to be similar to a 'follow-along' exercise on React and its component libraries. In fact, we provide you with most of the code and many examples, and the short tasks ask you to fill in or correct some of this implementation. In doing so, you will have the benefit of learning these (much needed) web development skills while avoiding a steeper learning curve.

Advice to Students

In many ways, this milestone will guide you through the implementation of a React client using UI libraries in a 'bubble'. This is because we want you to take advantage of a highly-guided and gentle introduction before you eventually are able to fully implement such software independently. For example, we already made the application structure, selected and imported the libraries, will point you to the exact page of the documentation that you need to follow, and made examples that you can use to understand the usage of these in a very specific context. However, if this were your project, you would have to do a lot of these tasks on your own (even if you use the code developed for this assignment as a template)! You should not underestimate the time you might need to spend on choosing the right libraries, finding the correct portion of the documentation, debugging, and going through the many other steps that you need to complete even before you are ready to code your planned implementation!

While this may sound daunting, we are confident that a thoughtful attempt at this milestone will prepare you well for the project! Another (important) skill that you will also develop from this assignment is that of code collaboration. When working on a large project, you will inevitably have to read, understand, and possibly modify or add to code written by your peers. This also includes debugging and correcting that code. You'll find this analogous to the short tasks we lay out for you in that you will need to understand first our implementation and the requirements

first, and then edit or extend the code. Of course, you will need to do this much more independently for the project, but working through the milestone with this in mind will help you build the right mindset.

Please start early, be patient, and avoid last minute Piazza and OH traffic!

Setup

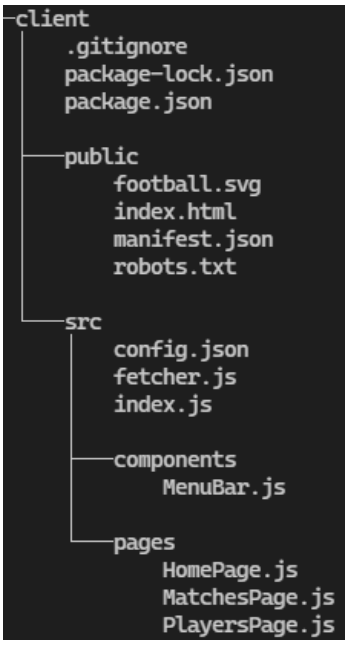
Required & Recommended Programs

Requirements and recommendations from MS1 hold. You will also need to install the XCode command-line tools (**if you are using a Mac**). To do this, run:
`xcode-select --install`

Application Structure

You will need to use the server application from your MS1 solution in conjunction with the React client described below. Documents in the `./docs` folder from MS1 will also come in handy for reference. Feel free to revisit the application structure from the MS1 handout if needed.

Extract the .zip file for MS2 from the assignment page. You will see the `./client` folder at the root under which the following folders and files can be found:



Here is an explanation of these folders and (selected) files:\

/ (root)

- .gitignore: A gitignore file for the client application. Read more on .gitignore files here
- package.json: maintains the project dependency tree; defines project properties, scripts, etc
- package-lock.json: saves the exact version of each package in the application dependency tree for installs and maintenance

/public

This folder contains static files like index.html file and assets like robots.txt for specifying web page titles, crawlability, et cetera (more info here)

/src
This folder contains the main source code for the React application. Specifically:

- config.json: Holds server connection information (like port and host). Could be replaced by a *.env* file, but students find this easier to manage
- fetcher.js: Contains helper functions that wrap calls to API routes. improved testability, reusability, and usability
- index.js: This the main JavaScript entry point to the application and stores the main DOM render call in React. For this application, page routing via components and imports for stylesheets are also embedded in this file.
- **/pages** This folder contains files for React components corresponding to the three pages in the application (see the sections below for more details). These are:
    - HomePage.js: The landing page, provides a brief overview of players and matches in the form of two paginated tables
    - MatchesPage.js: A page specifically for matches: allows users to search for a specific match and view specific details for a selected match
    - PlayersPage.js: A page specifically for players: allows users to search and filter for players and provides a detailed view of the player with visualizations for selected statistics
- **/components** Similar to the /pages folder, but this folder contains files for React components corresponding to smaller, reusable components, especially those used by pages. In this application, this is only the top navigation bar (described in MenuBar.js) used by all three pages. This is a good structure to follow for larger applications (such as the project)

Getting Started

Make sure that you have the required software installed and have a good understanding of the lecture and recitation materials before proceeding.

Open the (unzipped) folder using your code editor. If you are using VS Code, you should be able to do this by clicking *File -> Open Folder* from the top menu. You could also just use the 'code' command on the terminal or right click on the folder and select 'open with code' if you have added VS code to the terminal or to the options menu for your system respectively.

Open a new terminal (on VS code) and cd into the client folder, then run npm install:

```
cd client
npm install
```

This will download and save the required dependencies into the *node_modules* folder within the client directory.

Note: You might encounter some warnings about deprecated dependencies and vulnerabilities, but you can safely ignore them for this assignment

CIS 550 Fall 2021

Understanding React

Recitation 3, the lecture materials, and the official React docs provide some great guidance on this, but here are a few essentials we think you might find helpful. First, you should start the server application that you developed for MS1 (feel free to refer to the MS1 handout if need be).

After starting the server application, which should run on port 8080, you should start the React application by running the command npm start within the /client directory in a terminal window and follow along as needed. **It is imperative that you run the server application before starting the client** since the client assumes that the server is able to communicate and return necessary data. If there is such a communication issue, you will most likely get an error like '*Unhandled Rejection (TypeError): Failed to fetch*'.

You can safely ignore any warnings (especially about unused components or deprecations). Once the build is running, the last few lines of the terminal output (for the starter code) should look like this:

```
added 2173 packages, and audited 2174 packages in 51s
Compiled with warnings.

src\pages\MatchesPage.js
  Line 7:5:    'Pagination' is defined but never used           no-unused-vars
  Line 19:9:   'Column' is assigned a value but never used      no-unused-vars
  Line 19:17:  'ColumnGroup' is assigned a value but never used  no-unused-vars

src\pages\PlayersPage.js
  Line 2:62:    'CardTitle' is defined but never used                        no-unused-vars
  Line 5:5:     'Table' is defined but never used                            no-unused-vars
  Line 6:5:     'Pagination' is defined but never used                       no-unused-vars
  Line 7:5:     'Select' is defined but never used                           no-unused-vars
  Line 21:27:   'getPlayer' is defined but never used                        no-unused-vars
  Line 24:7:    'playerColumns' is assigned a value but never used           no-unused-vars
  Line 166:29:  Invalid alt value for img. Use alt="" for presentational images  jsx-a11y/alt-text
  Line 190:37:  Invalid alt value for img. Use alt="" for presentational images  jsx-a11y/alt-text

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

This application, by default, runs on localhost - port 3000. Once you run the above command, your default browser should open up a window to localhost:3000 (you might need to wait a few seconds for it to load).

The client application uses a router to accept a request to the port on a host. For example, this application runs on the host 'localhost' and port 3000. Upon receiving a request from a client (your browser in this case), the application uses a routing library (react-router) that parses the URL string to map it to a registered route (see *index.js*), which, in turn, renders the React page component corresponding to that route. For example, the path http://localhost:3000/players will render the page component **`PlayersPage`**.

Note that we are using React components instead of hooks since the former is closer to classes in Java, and students seem to be more comfortable understanding these. Please refer to the recitation or read more about React components and props here, but here is a short refresher on some essentials:

- Components use props (think of these as arguments passed to a constructor) and maintain a JavaScript object, **`state`**, that holds the necessary variables to represent the current situation of the component

- Components have a **`render`** method that describes how the view is rendered in a browser window. This is also the only required method in a component
- The **`componentDidMount()`** method is called at the first stage of the component life cycle (mounting) and should be used to initialize the state as needed, for example, by making API calls and setting the data displayed by the view
- For functions to be accessible as methods of the component, they must be 'bound' using the **`bind`** method in the component's constructor

See **`PlayersPage`** for an example of how these are used. Additionally, you should refer to these docs to understand how controlled components work in react, especially in the context of forms.

One reason why React is widely preferred by many developers is the vast collection of UI component libraries that make it easy for developers to quickly and easily create beautiful interfaces. In this assignment, you will be working with three such libraries - Ant Design, Shards React, and React-Vis. A careful look at the starter code (especially the files for pages and components) will give you a good idea of how these components are imported and used, but we strongly recommend looking into the documentation for a better understanding. You will find the task-specific documentation links we provide in the following section very helpful as well.

An Overview of Tasks

(60 Points)

In this section, we categorically summarize the tasks that you will complete. The specific instructions and hints for these tasks are included in the code, as inline comments, in the form "***TASK x:*** **…",** for example: `//TASK 11: call getMatchSearch and update matchesResults in state. See componentDidMount() for a hint`. Each of the thirty-two (32) tasks is very short (roughly 1 - 3 lines on average) and is weighted at 2 points. This excludes tasks 12 and 24, weighted at 0 points, which require you to copy over your implementation from other tasks.

You should look at the following files in **src/pages** to complete the tasks in numerical order, but feel free to deviate from this slightly (especially if you would like to implement tasks by categories - remember that tasks may build on top of each other, though!):

- HomePage.js (Tasks 1 through 9)
- MatchesPage.js (Tasks 10 through 18)
- PlayersPage.js (Tasks 19 through 32)

Category 1: Fetching / Updating Component State

Description: These tasks will involve updating the component state to display data, potentially using functions from *fetcher.js* to query the API and parse in the response to get relevant data
Relevant Tasks: 1, 2, 10, 11, 20, 21, 22, 23, 25
Suggested files/documentation:

- fetcher.js
- Using fetch (Mozilla docs)
- Async (Mozilla docs)
- State and Lifecycle (React docs)

Category 2: Select (Ant Design)

Description: Using the selector component and tracking it using the onChange method and state variables
Relevant Tasks: 3
Suggested files/documentation:

- Select (Ant Design docs)
- Forms (React docs)

Category 3: Tables, Columns, Column Groups, Sorters (Ant Design)

Description: Creating columns by mapping response keys to columns, potentially with custom sorters for values for a column to display state data
Relevant Tasks: 4, 5, 6, 7, 8, 9, 12, 13, 14, 19, 24, 28, 29
Suggested files/documentation:

- Table (Ant Design docs)

Category 4: Tables, Rows (Ant Design)

Description: Creating and styling rows with embedded elements to display state data
Relevant Tasks: 15, 16, 17
Suggested files/documentation:

- Table (Ant Design docs)

Category 5: Progress Bars (Shards React)

Description: Creating and styling progress bars to display state data
Relevant Tasks: 18, 31
Suggested files/documentation:

- Progress (Shards React docs)

Category 6: Forms, FormGroups (Shards React)

Description: Creating, styling, and managing state with input forms
Relevant Tasks: 26, 27
Suggested files/documentation:

- Form (Shards React docs)
- FormInput (Shards React docs)
- FormGroup (Shards React docs)

Category 7: Rate (Ant Design)

Description: Creating and styling a rating component to display state data
Relevant Tasks: 30
Suggested files/documentation:

- Rate (Ant Design docs)

Category 8: Radar Charts (React Vis)

Description: Creating and styling progress bars to display state data
Relevant Tasks: 32
Suggested files/documentation:

- Radar Chart (React-Vis examples)

Submission and Grading

Once you are convinced that you have implemented the functionality required, cd into the client directory and DELETE the node_modules folder by running the command:

```
rm -r ./node_modules/
```

Then, zip the contents of the .client folder and submit the zip file to the Homework 2-MS2 submission item on Gradescope. Once the submission deadline has passed, we will grade your submissions manually assessing the correctness of your implementation.