

CSE 316: Fundamentals of Software Development

Stony Brook University

Programming Assignment #2

Spring 2022

Assignment Due: Mar 15, 2022, 11:59 PM

Learning Outcomes

After completing this homework assignment, you will be able to

1. Learn the principles of reactive applications.
2. Develop dynamic webpages using React.

Introduction

In this assignment, we will re-create a mock stackoverflow.com web application using the React Library. Since we will be using the same style for this application, you can use the same CSS that you made for homework 1.

Getting Started

Install [Node.js](https://nodejs.org/). We will need Node.js to manage the react app and its dependencies. We do not actually need to know node.js yet. When you install Node.js, it will come with the **npm** package manager, which will also get installed. We will use **npm** to install dependencies and start the react application.

Download/clone your personal GitHub repository (**add link**). The repository is structured as a react application. The `public/` directory has metadata about the application such

as images and logos and the main HTML page, `index.html`. You should not change anything in this file.

The `src/` directory currently has `index.js`. React looks for this file when rendering content to the browser. You can add code to this file but do not rename it. The directory also has `index.css`. You can re-use the CSS file you used for HW 1. All components of the application should be defined in the directory `src/components`. The `src/models` directory defines the underlying data structure that the application uses. Currently, it only has `model.js`. You can add your own methods to the class in this file, similar to HW1. You should create an instance of this model when the app starts and share it with all components that need it to render elements. Ideally, you should only create one instance of the model and maintain that throughout the lifetime of your application.

In the repository, you will find two files called `package.json` and `package-lock.json`. **Do not change these files.** If you have added dependencies to these files, then you will need to commit these files to the repository so we can run your application with the same dependencies. However, there should be no reason for you to change these files. These files list the external dependencies which we will need for the react application to run successfully on your local machines. When you clone the repository run the command **npm install**. Running the command will create a directory called `node_modules` which will contain all the required packages. You can then start your application using the command **npm start**. Running this command will open a local host server at port 3000. The application will automatically open in a browser. If it does not then manually open `http://localhost:3000/` in a browser. In the browser, you should see the message **Change Me** displayed. You can now go proceed and add code. If you change code and save it, the server will automatically pick up the changes.

Note when you commit your code to the repository, the `node_modules` directory will not be committed. This is fine. We will run *npm install* on our end and recreate the required packages.

Grading

We will clone your repository and run your code in the Chrome web browser as a react application. You will get points for each functionality implemented. Make sure you test your code in Chrome. The rubric we will use is shown below:

1. Banner: 10 pts.

2. All Questions Page: 10 pts.
3. Post a New question: 15 pts.
4. Searching by text: 10 pts.
5. Searching by tags: 10 pts.
6. Answers Page: 10 pts.
7. Post a new answer: 10 pts.
8. All Tags page: 10 pts.
9. Questions of a tag: 10 pts.
10. Code Modularity: 5 pts.
 - a. Codebase should be composed of components.
 - b. Components should set up and manage their own state as much as possible.
 - c. Minimal code reuse.
 - d. Easy to read and understand.

Total points: 100 pts.

The functionality required for building the React application is the same as that in HW1. For convenience, the requirements have been listed below again. Changes are highlighted in blue.

Note, the application must be made using React. You will not receive credit if you use anything else.

Data Model

The primary data elements we need to store for this application are questions, tags, and answers. To this end, we will use a JavaScript Object to represent the application data in memory. The object has the following attributes:

- *questions*. A list of *Question* objects.
- *tags*. A list of *Tag* object.
- *answers*. A list of *Answer* objects.

The *Question* object has the following attributes:

- *qid*. A unique string used to uniquely identify this question.

- *title*. A string to hold this question's title. The max length of the string is 100.
- *text*. A string to hold this question text.
- *tagIds*. A list of strings to hold the tag ids of tags associated with this question.
- *askedBy*. A string to indicate the username associated with this question.
- *askedOn*. A string to indicate the date of this question.
- *askedAt*. A string to indicate the time of this question.
- *answers*. A list of strings to hold the answers ids of answers associated with this question.
- *views*. A number to indicate the no. of times the question has been viewed.

The *Tag* object has the following attributes:

- *tid*. A unique string used to uniquely identify this tag.
- *name*. A string to hold this tag's name.

The *Answer* object has the following attributes:

- *aid*. A unique string used to uniquely identify this answer.
- *text*. A string to hold this answer's text.
- *ansBy*. A string to indicate the username associated with this answer.
- *ansOn*. A string to indicate the date of this answer.
- *ansAt*. A string to indicate the time of this answer.

See the class definition of this object in `src/model.js` in the code repository.

Since React is still a client-side scripting framework the data will not be persistent. This means that anytime the application is stopped and restarted, all the data created during that session will be lost and the application will begin with the initial state. This is fine for the purposes of this homework assignment. In later homework assignments, we will see how the application state can be persisted in a backend system using server-side scripting.

Application Behavior and Layout

We will mimic the original stackoverflow.com website as much as possible. Although, we won't be implementing all of its features. You can visit stackoverflow.com for inspiration. The layout is quite simple. It has two parts – a **banner** and the **main body**. The *banner* should remain constant, that is, it should have the same UI elements throughout and should be displayed at the top of the page. The *main body* will be displayed below the banner and will render relevant content based on user interactions with the web page.

Banner

The banner should have the following UI elements arranged horizontally:

1. A *link* with the name **Questions**.
2. A *link* with the name **Tags**.
3. The *name* of the application (**Fake Stack Overflow**).
4. A *search box* where users can do a text search.

Below is an example of how the banner should look:



Notice that the banner has a light grey background. The links are styled as blocks with no borders. Hovering over a link with the mouse highlights them. For example, hovering on *Tags* should look as follows:



Similarly, hovering on *Questions* should look as follows:



The font sizes shown in the figures above and in all subsequent images are relative to this document. You should select font sizes so that your content is clearly visible and scales accurately when rendered in a full-sized browser window.

All Questions Page

When a user loads the application in the browser for the first time, the banner should be displayed at the top of the page. The *Questions* Link should be highlighted with the color **#0281E8** to indicate that the user is currently viewing all questions. The *main body* of the page should render all the questions that have been asked in a tabular format as follows:

- A header row with 3 columns. The first column should display the text **N Questions**, where N is the total no. of questions that have been asked. The second column should display a title with the text **All Questions** to indicate that all questions that have been asked are being displayed. The third column should display a *button* with the label **Ask A Question**. The button should have the color code **#165A92**. The contents in the columns have no other style constraints. However, make sure that they are clearly visible.
- The second column must occupy the maximum width. The first and the third columns should be of the same width but lesser than the second column.
- A row for each question. A row should have 3 columns.
 - The first column should display the text **N1 views** first and then **N2 answers** in separate lines, N1 is the no. of views for the question and N2 is the no. of answers.
 - The second column should display the question's title as a link followed by a list of tags in a new line. Hovering over the title's link should highlight the link with a deeper shade of the existing text color. The list of tags should have rounded borders and should be displayed beside each other. There should be 4 tags per line. For example, if the question has 7 tags, they should be displayed in two lines with the first line showing 4 tags and the next 3 tags.
 - The third column should display three lines. The first line should display the text **asked by <user>**, where <user> is the username of the person that asked the question. The second line should display the text **on <Month Day, Year>** and the third line should display the time as **at <hh: min>**.
- The questions should be displayed in ascending order of the day and time they were asked. In other words, the most recently asked questions should be displayed first.
- Make sure that all fonts and content are clearly legible.
- After each question row, a horizontal divider should be drawn. You can choose the style of this divider.
- There should be no other borders surrounding the questions being displayed.
- If the total no. of questions does not fit on the page, then the *main body* must be made scrollable. Note the banner must remain fixed to the top of the page.

Below is an example of how the page should look.

Questions

Tags

Fake Stack Overflow

2 Questions

All Questions

Ask A Question

10 Views

2 Answers

Programmatically navigate using React router

react

javascript

Asked By JoJi John

On Jan 19, 2022

At 21:25

121 Views

3 Answers

android studio save string shared preference, start activity and load the saved string

android-studio

javascript

shared-preferences

Asked By saltyPeter

On Jan 01, 2022

At 01:15

New Question Page

When a user clicks on the **Ask A Question** button, the *main body* section of the page should display a form with the following inputs fields:

- A text box for question title. The title should not be more than 100 characters and should not be empty.
- A text box for question text. Should not be empty.
- A text box for a list of tags that should be associated with the question. This is a space-separated list. A user is allowed to enter duplicate tag names but only one of those should be saved in the underlying model. Tag names are case insensitive. This means that "JavaScript" and "javascript" should count as one tag.
- A text box for the username of the user asking the question. The username should not be more than 15 characters.

There should be a button with the label *Post Question* and hex code #165A92 as the background color. Each field in the form should have an appropriate hint to help the user enter the appropriate data. An example of such a page is shown below:

[Questions](#) [Tags](#) **Fake Stack Overflow**

Question Title

Title should not be more than 100 characters.

Question Text

Add details.

How can I fix the following syntax error in my code:
`driver.get(" testing")`
The syntax error seems under com. It's very confusing.

Tags

Add Keywords separated by whitespace.

Username

Should not be more than 15 characters.

When the *Post Question* button is pressed, the question should be added to the *data object* in *model.js*. If the question is added successfully then the *main body* section of the page should display all the questions including the question currently added in sorted order of the time they were posted. Further, the page should also display the total no. of questions, which should have incremented by 1. For example, in the page shown below, the first question displayed was most recently asked by the user *jumanji* using the inputs on the previous page.

Questions

Tags

Fake Stack Overflow

Search ...

3 Questions

All Questions

Ask A Question

0 Views

0 Answers

Web scripting invalid syntax URL

web-scripting HTML

Asked By jumanji

On Jan 25, 2022

At 12:19

10 Views

2 Answers

Programmatically navigate using React router

react javascript

Asked By JoJi John

On Jan 19, 2022

At 21:25

121 Views

3 Answers

android studio save string shared preference, start activity and load the saved string

android-studio javascript shared-preferences

Asked By saltyPeter

On Jan 01, 2022

At 01:15

An error will occur if unexpected data is entered in any of the input fields or a runtime error occurs when updating the *data object*. If an error occurs, the *main body* of the page should still display the form with appropriate error messages at the top of the form. The error messages' font color should be red. For example, in the figure shown below, the user did not add any tags to the question and entered a username that is more than 15 characters.

QuestionsTagsFake Stack OverflowSearch ...

Tags cannot be empty!
Username cannot be more than 15 characters!

Question Title

Title should not be more than 100 characters.

Web scripting invalid syntax URL

Question Text

Add details.

How can I fix the following syntax error in my code:
`driver.get(" testing")`
The syntax error seems under com. It's very confusing.

Tags

Add Keywords separated by whitespace.

Username

Should not be more than 15 characters.

jumanjimookherjee

Post Question

Searching

A user can search for certain questions based on words occurring in the question text or title. On pressing the ENTER key, The search should return *all questions for which their title or text contains at least one word in the search string*. For example, in the example shown below, there is only one question in our data model that matches the search string.

Questions
Tags
Fake Stack Overflow

1 Question
Search Results
Ask A Question

121 Views
3 Answers

[android studio save string shared preference, start activity and load the saved string](#)

android-studio
javascript
shared-preferences

Asked By saltyPeter
On Jan 01, 2022
At 01:15

Further, if a user surrounds individual words with `[]` then all questions corresponding to each tagname in `[]` should be displayed. *The search results should be displayed when the user presses the **ENTER** key.* See the example in the figure below.

Questions
Tags
Fake Stack Overflow

2 Questions
Search Results
Ask A Question

10 Views
2 Answers

[Programmatically navigate using React router](#)

react
javascript

Asked By JoJi John
On Jan 19, 2022
At 21:25

121 Views
3 Answers

[android studio save string shared preference, start activity and load the saved string](#)

android-studio
javascript
shared-preferences

Asked By saltyPeter
On Jan 01, 2022
At 01:15

Note the searching is **case-insensitive**. A search string can contain a combination of [tagnames] and non-tag words. When this happens, all questions tagged with each [tagname] and all questions with text or title containing at least one of the non-tag words should be displayed.

If the search string does not match any question or tag names then display the **No Questions Found**. The total number of questions displayed should be 0 and the page title and the button should remain.

Answers Page

Clicking on a question link should increment by 1 the no. of views associated with the question and load the answers for that question in the *main body* of the page. Note the banner should still remain at the top of the page. The answers should be displayed in a tabular format as follows:

- A header row with 3 columns. The first column should display the text **N answers**, where N is the total no. of answers given for the question. The second column should display the title of the question in bold. The third column should display a *button* with the label **Ask A Question**. You are free to add other style constraints to the elements other than what has already been mentioned. However, make sure that they are clearly visible.
- The second column must occupy the maximum width followed by the first column and then the third column.
- The second row should display 3 columns.
 - The first column should display the text **N views** indicating the no. of times the question has been viewed (including this one).
 - The second column should display the question text.
 - The third column should display three lines. The first line should display the text **asked by <user>**, where <user> is the username of the person that asked the question. The second line should display the text **on <Month Day, Year>** and the third line should display the time as **at <hh: min>**.
- The answers to the questions should be displayed in subsequent rows. An answer row should have 2 columns.
 - The first column should display the answer.
 - The second column should display three lines. The first line should display the text **Ans by <user>**, where <user> is the username of the person that asked the question. The second line should display the text **on <Month Day, Year>** and the third line should display the time as **at <hh: min>**.
- If no. of answers do not fit on the page, then add a scroll bar.
- The answers should be displayed in ascending order of the day and time they were posted. In other words, display the answers that were posted most recently first.
- At the end of all the answers, you should display a *button* with the label **Answer Question**. This button should be centered relative to the page. It should have rounded borders on all four sides. It should have hex code **#165A92** as the background color. Make sure that the button and the label are clearly visible.

- After each answer row, a horizontal divider should be drawn. You can choose the style of this divider.
- There should be no other borders surrounding the answers being displayed.

Shown below is an example of the 'answers' page for a question.

Questions
Tags
Fake Stack Overflow

2 Answers
Programmatically navigate using React router
Ask A Question

11 Views

the alert shows the proper index for the li clicked, and when I alert the variable within the last function I'm calling, moveToNextImage(stepClicked), the same value shows but the animation isn't happening. This works many other ways, but I'm trying to pass the index value of the list item clicked to use for the math to calculate.

Asked By [JoJi John](#)
On Jan 19, 2022
At 21:25

React Router is mostly a wrapper around the [history](#) library. history handles interaction with the browser's [window.history](#) for you with its browser and hash histories. It also provides a memory history which is useful for environments that don't have a global history. This is particularly useful in mobile app development (react-native) and unit testing with Node.

Ans By [hamkalo](#)
On Jan 25, 2022
At 12:19

On my end, I like to have a single history object that I can carry even outside components. I like to have a single history.js file that I import on demand, and just manipulate it.
You just have to change BrowserRouter to Router, and specify the history prop. This doesn't change anything for you, except that you have your own history object that you can manipulate as you want.
You need to install [history](#), the library used by react-router.

Ans By [azad](#)
On Feb 01, 2022
At 12:19

Answer Question

Pressing the *Ask A Question* button on this page will render elements as described in the **New Question Page** section.

New Answer Page

Pressing the *Answer Question* button will display a page with input elements to enter the new answer text and username. Note the banner should remain at the top of the page. The figure below shows an example of the same.

QuestionsTagsFake Stack OverflowSearch ...

Answer Text

You can use the new `useNavigate` hook. `useNavigate` hook returns a function which can be used for programmatic navigation. See example from the react router [documentaion](#)

Username

Should not be more than 15 characters.

jumanji

Post Answer

Pressing the *Post Answer* button, should capture the answer text and the username and update the data model. If the data entered satisfies all the necessary constraints and no runtime error occurs, then all the answers are displayed as shown on the **Answers Page**. As an example see the figure below.

Questions
Tags
Fake Stack Overflow

3 Answers
Programmatically navigate using React router
Ask A Question

11 Views

the alert shows the proper index for the li clicked, and when I alert the variable within the last function I'm calling, moveToNextImage(stepClicked), the same value shows but the animation isn't happening. This works many other ways, but I'm trying to pass the index value of the list item clicked to use for the math to calculate.

Asked By JoJi John
On Dec 12, 2021
At 21:25

You can use the new useNavigate hook. useNavigate hook returns a function which can be used for programmatic navigation. See example from the react router [documentaion](#)

Ans By jumani
On Feb 01, 2022
At 12:19

React Router is mostly a wrapper around the [history](#) library. history handles interaction with the browser's [window.history](#) for you with its browser and hash histories. It also provides a memory history which is useful for environments that don't have a global history. This is particularly useful in mobile app development (react-native) and unit testing with Node.

Ans By hamkalo
On Jan 25, 2022
At 12:19

On my end, I like to have a single history object that I can carry even outside components. I like to have a single history.js file that I import on demand, and just manipulate it.
You just have to change BrowserRouter to Router, and specify the history prop. This doesn't change anything for you, except that you have your own history object that you can manipulate as you want.
You need to install [history](#), the library used by react-router.

Ans By azad
On Mar 25, 2021
At 12:19

Answer Question

Note how there are now 3 answers for this question since a new answer was posted.

If the answer text is empty or the username is more than 15 characters (username may be empty) or a runtime error occurs, then display error messages in red on the same page similar to the *new question* page.

Tags Page

Clicking on the *Tags* link in the banner should display the list of all tags in the system. Additionally, the *Tags* link in the banner should be highlighted with hex code **#0281E8**. The page should render the following elements in a tabular format:

1. A header row with 3 columns. The first column should display the text **N Tags** in the first column, **N** is the no. of tags in the system. The second column should display the text **All Tags**. The third column should display the button with the label **Ask A Question**. This is the same button that was described in the *Questions* and *Answers* pages.
2. The following rows should display the tags in groups of 3, that is, each row should have at most 3 tags. Each tag should be displayed in a block with black-colored borders. The block should display the tag name as a link and the no. of questions associated with the tag in a new line in the same block.

The figure below shows an example of the page.

[Questions](#)

Tags

Fake Stack Overflow

Search ...

6 Tags

All Tags

Ask A Question

[react](#)
1 question

[javascript](#)
2 questions

[android-studio](#)
1 question

[shared-preferences](#)
1 question

[web-scripting](#)
1 question

[HTML](#)
1 question

Upon clicking a tag link, the questions associated with the tag should be displayed. For example, if the *javascript* tag is clicked then the page shown below is displayed. All questions associated with the *javascript* tag are rendered.

Questions
Tags
Fake Stack Overflow

2 Questions
Questions tagged [javascript]

Ask A Question

11 Views
2 Answers

Programmatically navigate using React router

react
javascript

Asked By JoJi John
On Jan 19, 2022
At 21:25

121 Views
3 Answers

android studio save string shared preference, start activity and load the saved string

android-studio
javascript
shared-preferences

Asked By saltyPeter
On Jan 01, 2022
At 01:15

Submitting Code

You can submit code to your GitHub repository as many times as you want till the deadline. After the deadline, any code you submit will be rejected. To submit a file to the remote repository, you first need to add it to the local git repository in your system, that is, the directory where you cloned the remote repository initially. Use the following commands from your terminal:

```
$ cd /path/to/cse316-hw2-<username> (skip if you are already in this directory)
```

```
$ git add <filename-you-want-to-add>
```

To submit your work to the remote GitHub repository, you will need to commit the file (with a message) and push the file to the repository. Use the following commands:

```
$ git commit -m "<your-custom-message>"
```

```
$ git push
```

Finally, submit your GitHub username to the listing created in Blackboard for this homework. This will help us locate your repository easily.

