

Practical 8 : Laravel ReactJS CRUD with RESTful API (Part II).

In this lab, we will continue to explore the usage of client-side scripts using a JavaScript library. In this case, we will explore ReactJS library with Laravel with the continuation of previous laravelAuth web application that we have created for Laravel Authentication and Authorization.

1. Define Front End for Adding, Updating and Deleting a Post.

We can choose to use Modals in reactstrap to pop up pages so that user may interact the different CRUD through pop up forms (modals). Import components of Modal and declare default state of Modal as shown in Figure 1.

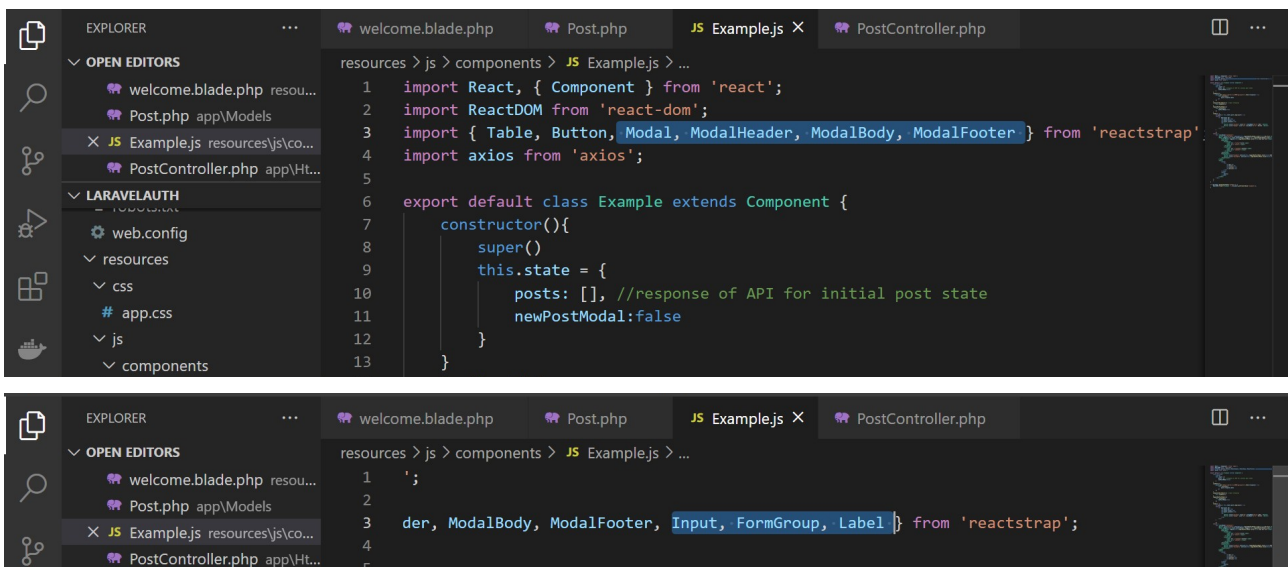


Figure 1: Initialize Modal and Form libraries and Modal's default state.

Define a state change for modal when a trigger (button onClick) is executed as shown in Figure 2.

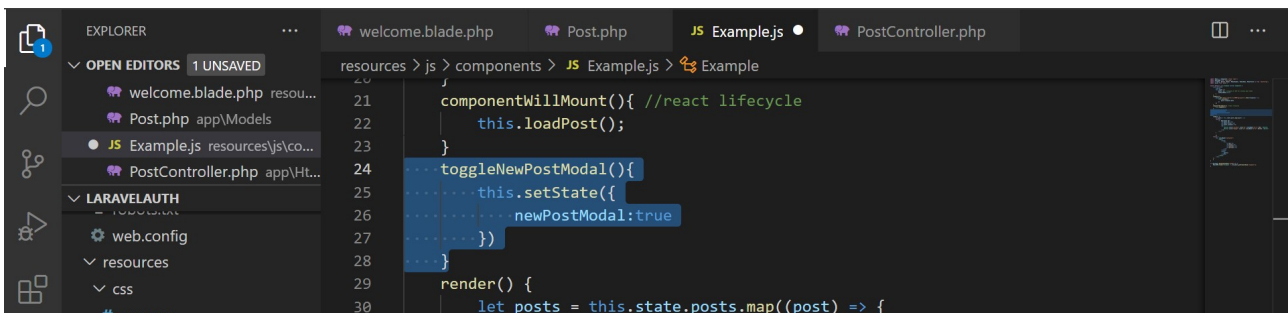


Figure 2: Change of Modal state.

Define the Modal through example codes provided in React Bootstrap 4 (<https://reactstrap.github.io/>) documentation as shown in Figure 3.

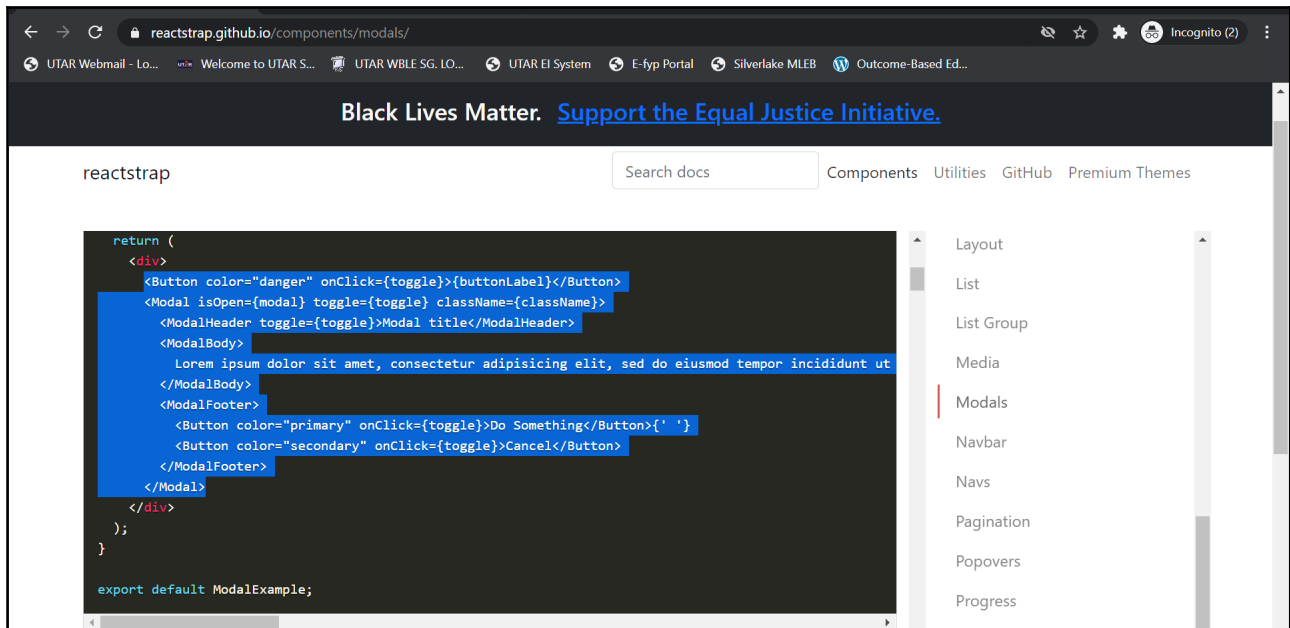


Figure 3: Modal example code.

Modify the code as shown in Figure 4 to create binding actions; bind the 'onClick' action of the button to set the Modal's state. This will complete the modal for adding a new post.

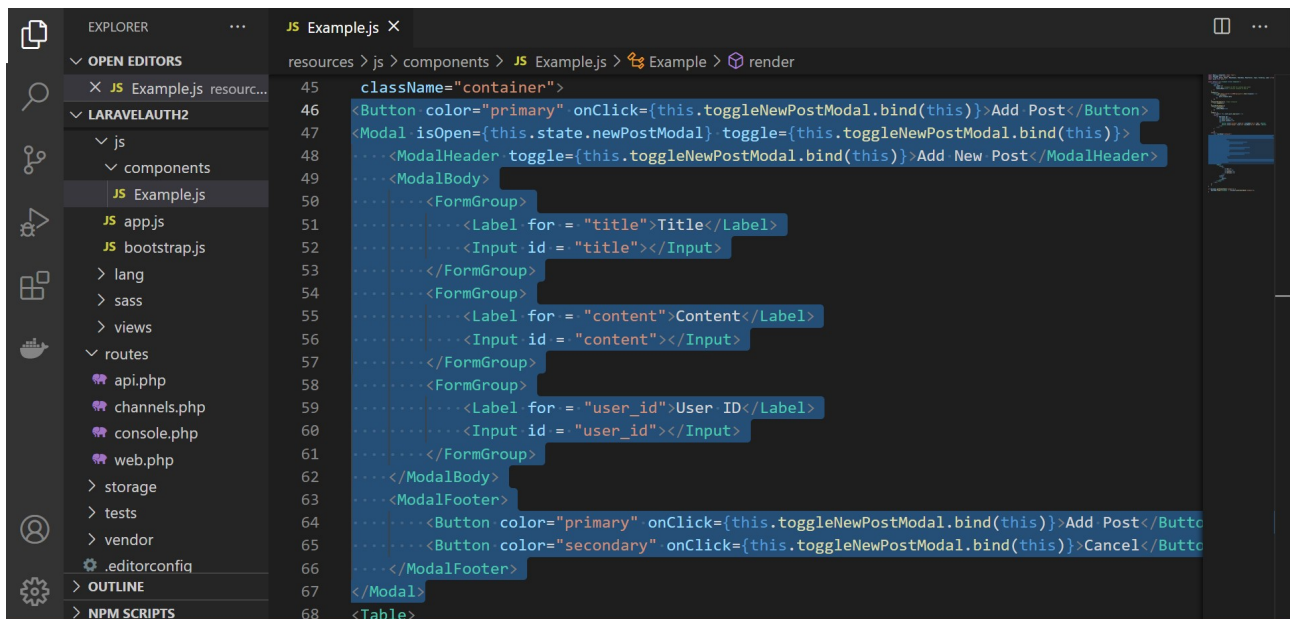


Figure 4: Define Modal for Add Post.

```

<Button color="primary" onClick={this.toggleNewPostModal.bind(this)}>Add Post</
Button>
<Modal isOpen={this.state.newPostModal}
  toggle={this.toggleNewPostModal.bind(this)}>
  <ModalHeader toggle={this.toggleNewPostModal.bind(this)}> Add New Post
  </ModalHeader>
  <ModalBody>
    <FormGroup>
      <Label for = "title">Title</Label>
      <Input id = "title"></Input>
    </FormGroup>
    <FormGroup>
      <Label for = "content">Content</Label>
      <Input id = "content"></Input>
    </FormGroup>
    <FormGroup>
      <Label for = "user_id">User ID</Label>
      <Input id = "user_id"></Input>
    </FormGroup>
  </ModalBody>
  <ModalFooter>
    <Button color="primary" onClick={this.toggleNewPostModal.bind(this)}>
      Add Post </Button>{' '}
    <Button color="secondary"
      onClick={this.toggleNewPostModal.bind(this)}> Cancel </Button>
  </ModalFooter>
</Modal>

```

Compile JS and CSS assets and host the web application to see the Modal pop up prior to onClick event of “Add Post” button as shown in Figure 5.

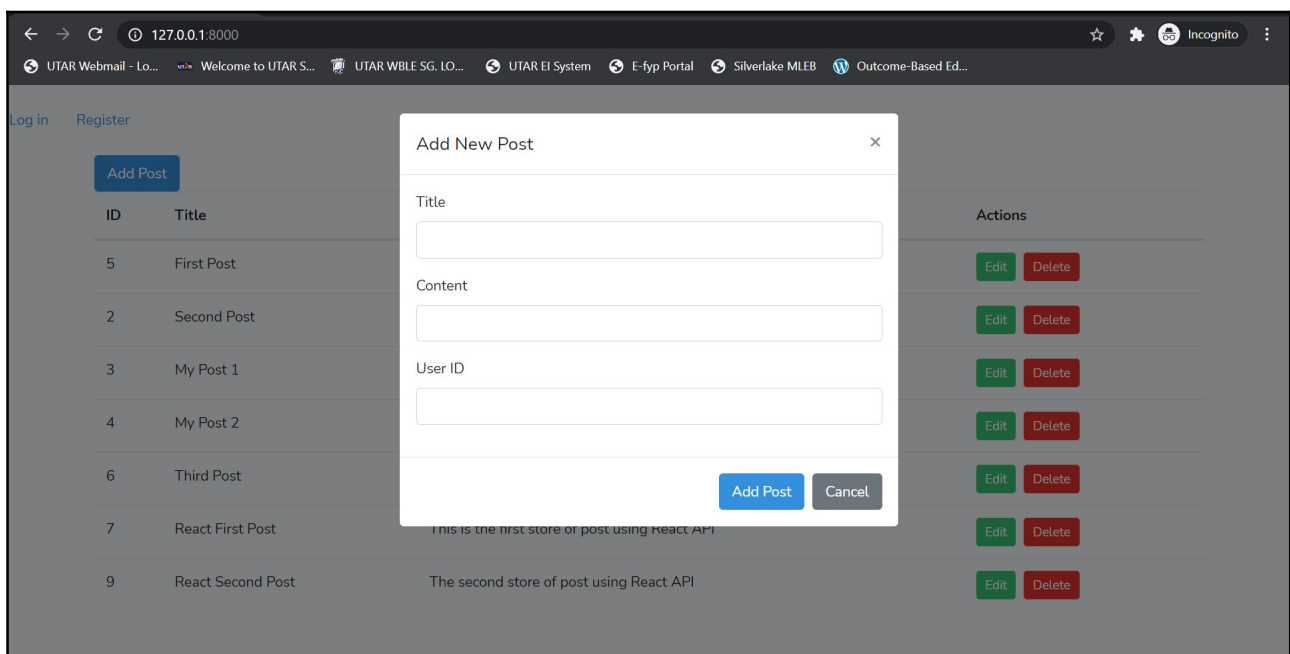
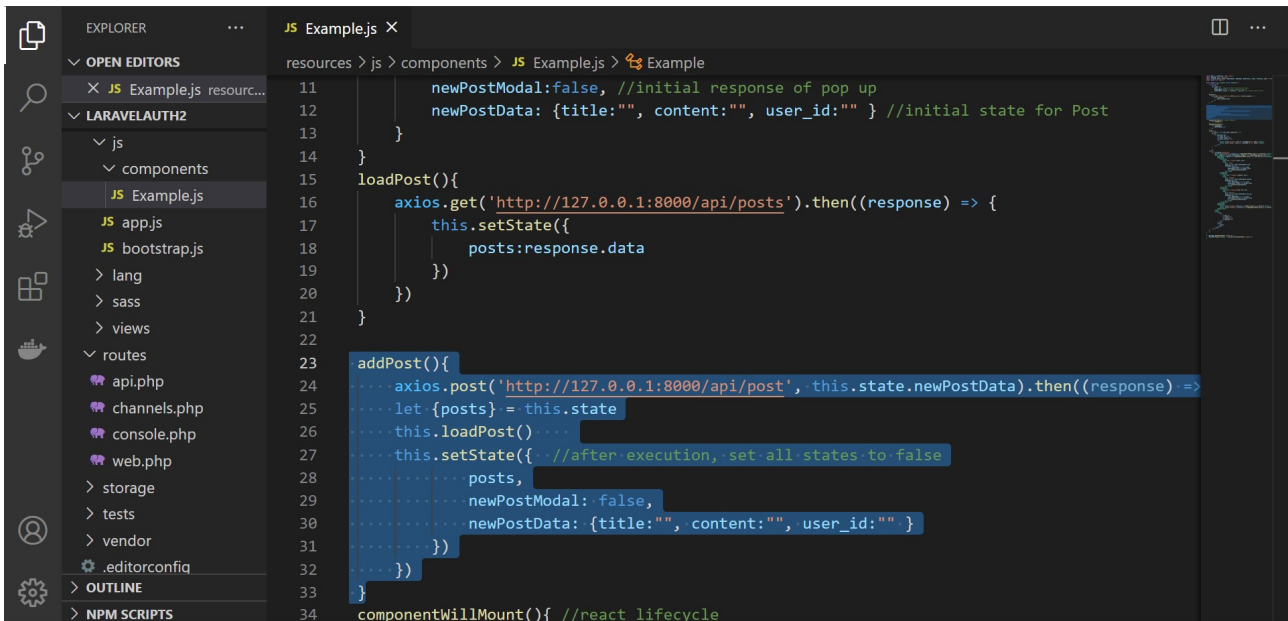


Figure 5: **newPostModal: true.**

The Modal has no further actions for all of the buttons on the Modal, for there are no definitions of setting back the state to be FALSE after the state was set to be TRUE prior to the onClick of “Add Post” button.

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

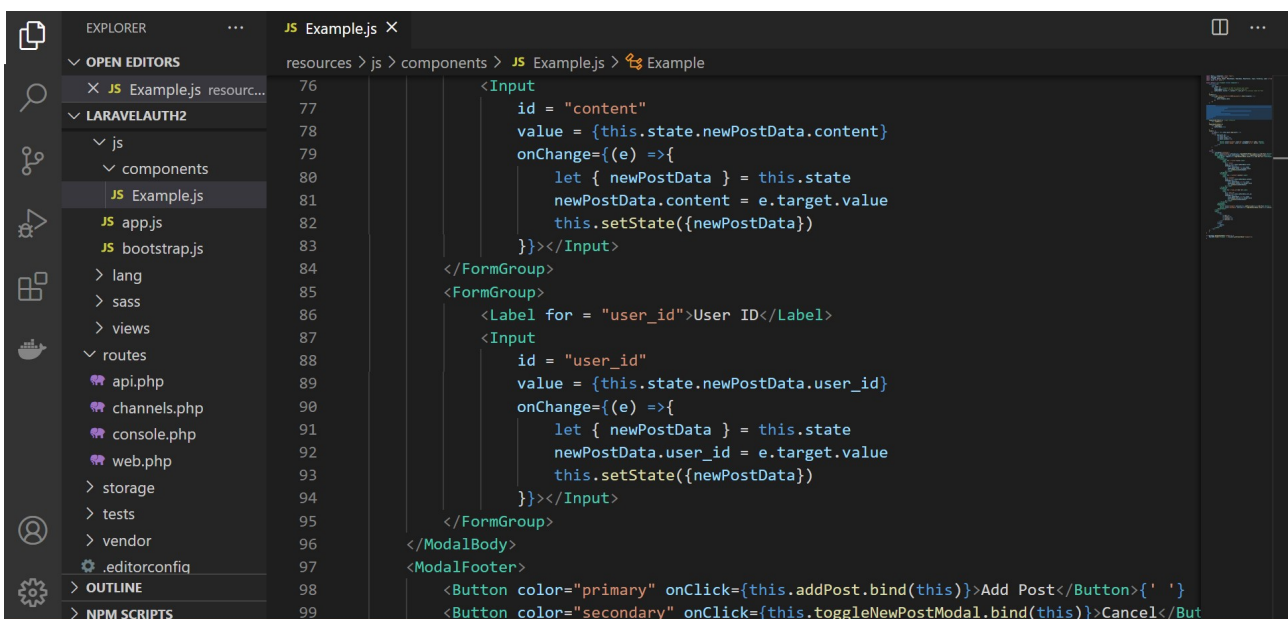
Since this modal is for the purpose of adding a new post, let's create a method/function for "post" API as shown in Figure 6.



```
resources > js > components > JS Example.js > Example
11      newPostModal:false, //initial response of pop up
12      newPostData: {title:"", content:"", user_id:""} //initial state for Post
13    }
14  }
15  loadPost(){
16    axios.get('http://127.0.0.1:8000/api/posts').then((response) => {
17      this.setState({
18        posts:response.data
19      })
20    })
21  }
22
23  addPost(){
24    axios.post('http://127.0.0.1:8000/api/post', this.state.newPostData).then((response) => {
25      let {posts} = this.state
26      this.loadPost()
27      this.setState({ //after execution, set all states to false
28        posts,
29        newPostModal: false,
30        newPostData: {title:"", content:"", user_id:""}
31      })
32    })
33  }
34  componentWillMount(){ //react lifecycle
```

Figure 6: addPost method/function to add a new post.

Prior to creation of addPost method/function, modify the Add Post modal's form components in order to send the request from user to API for processing of adding new post as shown in Figure 7.



```
resources > js > components > JS Example.js > Example
76      <Input
77        id = "content"
78        value = {this.state.newPostData.content}
79        onChange={ (e) =>{
80          let { newPostData } = this.state
81          newPostData.content = e.target.value
82          this.setState({newPostData})
83        }}></Input>
84    </FormGroup>
85    <FormGroup>
86      <Label for = "user_id">User ID</Label>
87      <Input
88        id = "user_id"
89        value = {this.state.newPostData.user_id}
90        onChange={ (e) =>{
91          let { newPostData } = this.state
92          newPostData.user_id = e.target.value
93          this.setState({newPostData})
94        }}></Input>
95    </FormGroup>
96  </ModalBody>
97  <ModalFooter>
98    <Button color="primary" onClick={this.addPost.bind(this)}>Add Post</Button>{' '}
99    <Button color="secondary" onClick={this.toggleNewPostModal.bind(this)}>Cancel</But
```

Figure 7: setStates to fetch the value from HTTP request (user inputs).

Then, compile JS and CSS assets and refresh the web application to create a new post as shown in Figure 8.

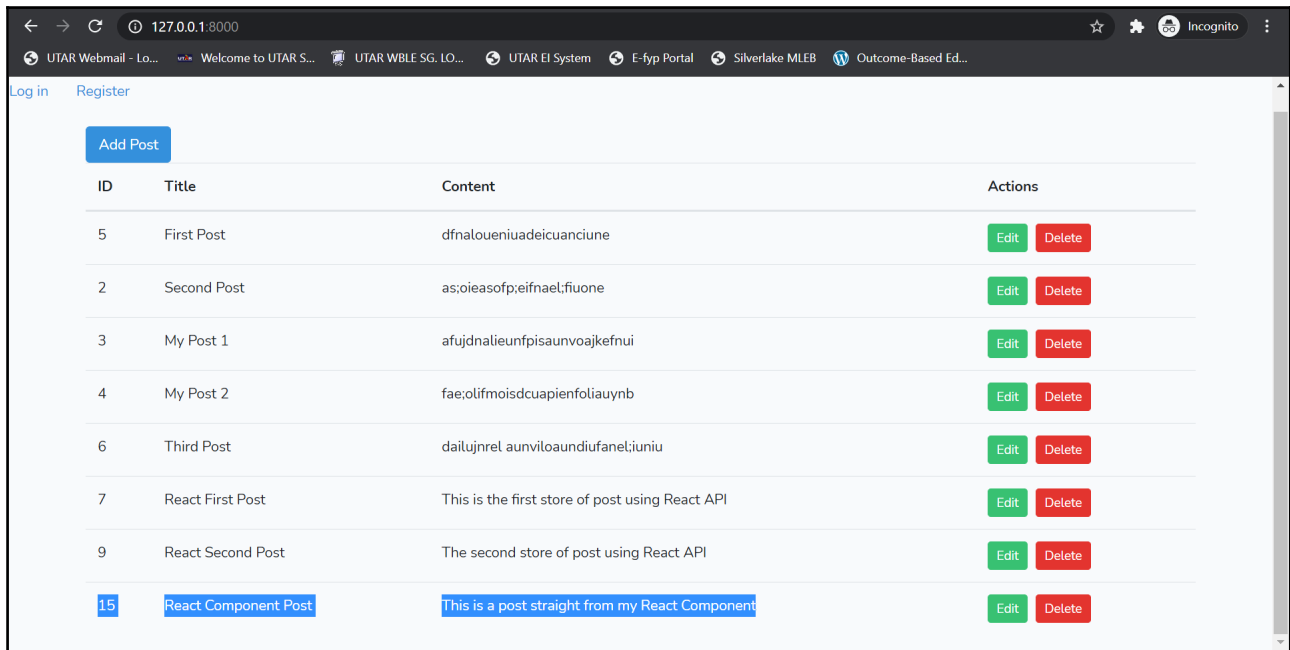


Figure 8: Add new post using React Component.

Now, let's proceed to create edit/update method/function to fetch the data from the table and another method/function for the "put" API in React Component to update Post as shown in Figure 9.

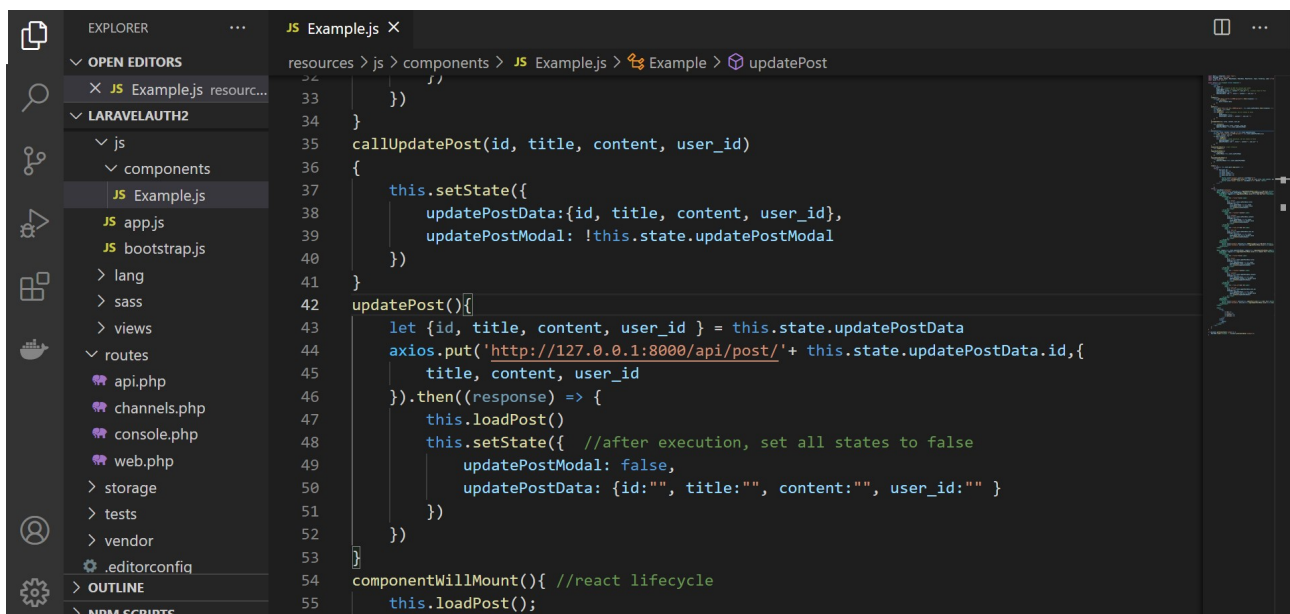


Figure 9: A method for fetching and changing modal state and a method for "put" API.

```

callUpdatePost(id, title, content, user_id)
{
  this.setState({
    updatePostData:{id, title, content, user_id},
    updatePostModal: !this.state.updatePostModal
  })
}
updatePost(){
  let {id, title, content, user_id } = this.state.updatePostData
  axios.put('http://127.0.0.1:8000/api/post/' + this.state.updatePostData.id, {
    title, content, user_id
  }).then((response) => {
    this.loadPost()
    this.setState({ //after execution, set all states to false
      updatePostModal: false,
      updatePostData: {id:"", title:"", content:"", user_id:"" }
    })
  })
}

toggleNewPostModal() {
  this.setState({
    newPostModal: !this.state.newPostModal
  })
}
toggleUpdatePostModal() {
  this.setState({
    updatePostModal: !this.state.updatePostModal
  })
}

```

Then, bind the “Edit” buttons in Actions to callUpdatePost method/function as shown in Figure 10.

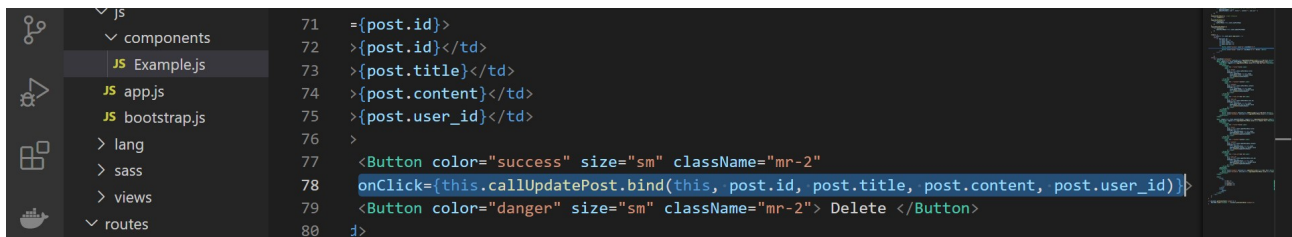


Figure 10: Bind callUpdatePost() to Edit button to invoke modal and API.

Exercise.

Create updatePostModal in order to be pop up, once onClick of Edit Button is triggered and bind the modal to toggleUpdatePostModal (*Hint: similar to what was done in newPostModal*).

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

Subsequent to creation of `updatePostModal`, proceed to test updating a post using React Component as shown in Figure 11.

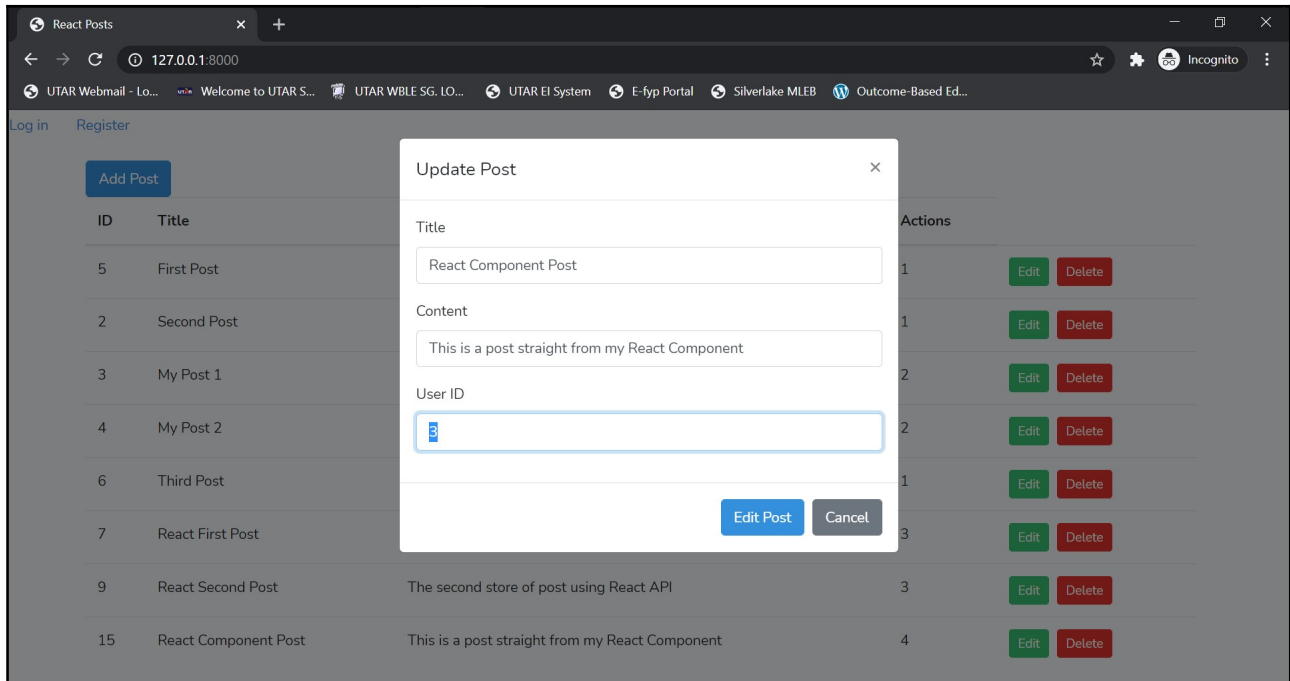


Figure 11: Update post using React Component.

Exercise:

Create a delete method/function for Delete button to function. Use the knowledge and understandings that you have for add post and edit post using React Component for this exercise. The output is as illustrated in Figure 12 (newly created Post id 15 is deleted subsequent to onClick of Delete button).

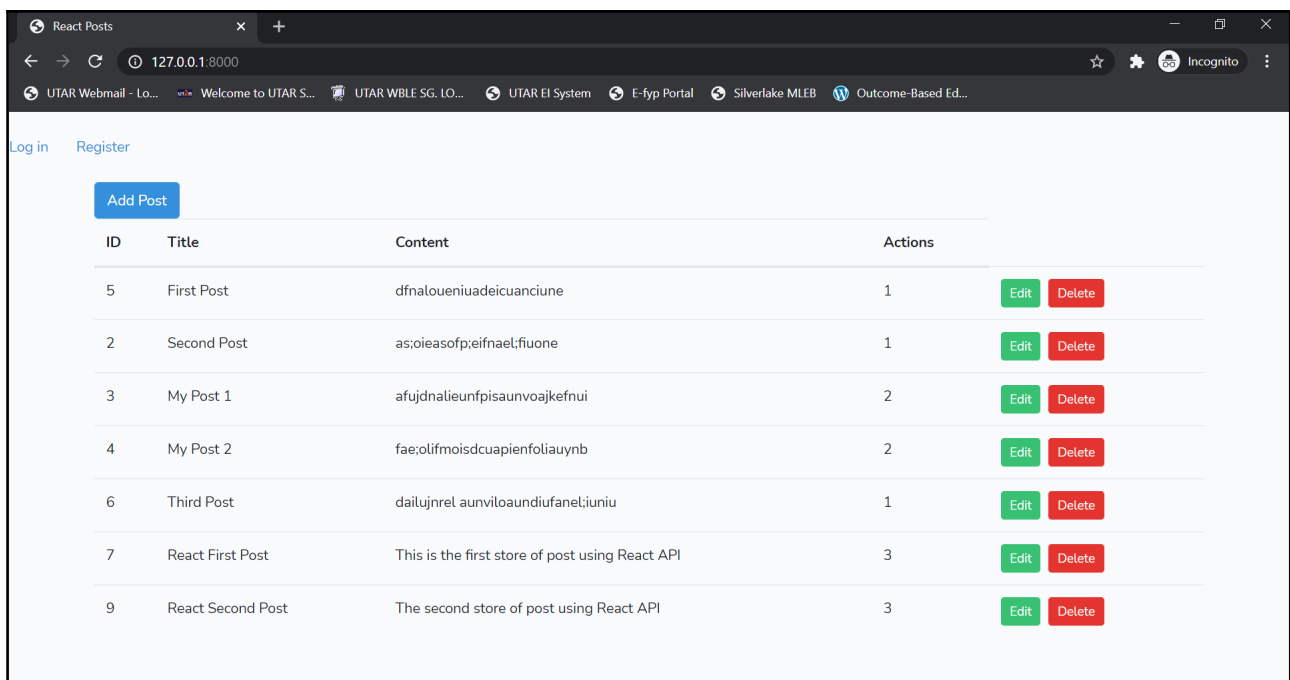


Figure 12: Delete post using React Component.

2. Using React Component within React Component.

We can modularize the larger component into smaller components. Let's modularize the “addPost” into another component as shown in Figure 13.

```

1  import React, {Component} from 'react';
2  import ReactDOM from 'react-dom';
3  import { Table, Button, Modal, ModalHeader, ModalBody, ModalFooter, Input, FormGroup, Label
4
5  export default class NewModal extends Component {
6
7      constructor(){
8          super()
9          this.state = {
10             posts: [], //response of API for initial post state
11             newPostModal:false, //initial response of pop up
12             newPostData: {title:"", content:"", user_id:"" }, //initial state for Post
13         }
14     }
15
16     loadPost(){
17         axios.get('http://127.0.0.1:8000/api/posts').then((response) => {
18             this.setState({
19                 posts:response.data
20             })
21         })
22     }
23
24     componentWillMount(){ //react lifecycle
25         this.loadPost();
26     }
27
28     addPost(){
29
30         axios.post('http://127.0.0.1:8000/api/post', this.state.newPostData).then((response)
31         let {posts} = this.state
32         this.loadPost()
33         this.setState({ //after execution, set all states to false
34             posts,
35             newPostModal: false,
36             newPostData: {title:"", content:"", user_id:"" }
37         })
38     })
39
40     toggleNewPostModal(){
41         this.setState({
42             newPostModal:!this.state.newPostModal
43         })
44     }
45
46     render(){
47         return(
48             <div>
49                 <Button color="primary" onClick={this.toggleNewPostModal.bind(this)}>Add Post</B
50                 <Modal isOpen={this.state.newPostModal} toggle={this.toggleNewPostModal.bind(thi
51                 <ModalHeader toggle={this.toggleNewPostModal.bind(this)}>Add New Post</Modal
52                 <ModalBody>
53                     <FormGroup>
54                         <FormGroup>
55                             this.setState({newPostData})
56                         </Input>
57                     </FormGroup>
58                     <FormGroup>
59                         <Label for= "user_id">User ID</Label>
60                         <Input
61                             id = "user_id"
62                             value = {this.state.newPostData.user_id}
63                             onChange={(e) =>{
64                                 let { newPostData } = this.state
65                                 newPostData.user_id = e.target.value
66                                 this.setState({newPostData})
67                             }}></Input>
68                     </FormGroup>
69                 </ModalBody>
70                 <ModalFooter>
71                     <Button color="primary" onClick={this.addPost.bind(this)}>Add Post</Butt
72                     <Button color="secondary" onClick={this.toggleNewPostModal.bind(this)}>C
73                 </ModalFooter>
74             </Modal>
75         </div>
76         );
77     }
78 }
79

```

Figure 13: “addPost” React Component.

If there are troubles loading popper js or SourceMaps (no modal after clicking “Add Post” button), follow the steps below:

Within webpack.mix.js

```

1  window._ = require('lodash');
2  import Popper from 'popper.js/dist/umd/popper.js'; //add this line for Popper js
3
4  /**
5   * We'll load jQuery and the Bootstrap jQuery plugin which provides support
6   * for JavaScript based Bootstrap features such as modals and tabs. This
7   * code may be modified to fit the specific needs of your application.
8   */
9
10 try {
11     //window.Popper = require('popper.js').default;
12     window.$ = window.jQuery = require('jquery');
13     window.Popper = Popper; //add this line for Popper js
14     require('bootstrap');
15 } catch (e) {}

```

Within resources\js\bootstrap.js

```

1  const mix = require('laravel-mix');
2
3  /*
4   | -----
5   | Mix Asset Management
6   | -----
7   |
8   | Mix provides a clean, fluent API for defining some Webpack build steps
9   | for your Laravel application. By default, we are compiling the Sass
10  | file for the application as well as bundling up all the JS files.
11  |
12  | */
13
14  mix.js('resources/js/app.js', 'public/js')
15      .sourceMaps() //add this for popper js
16      .react()
17      .sass('resources/sass/app.scss', 'public/css');

```

This works for enabling the popper js and SourceMaps.

