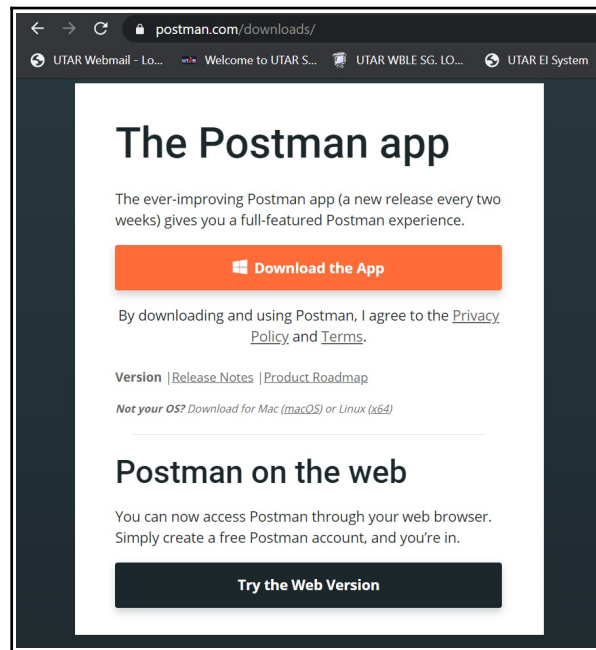


Practical 7 : Laravel ReactJS CRUD with RESTful API (Part I).

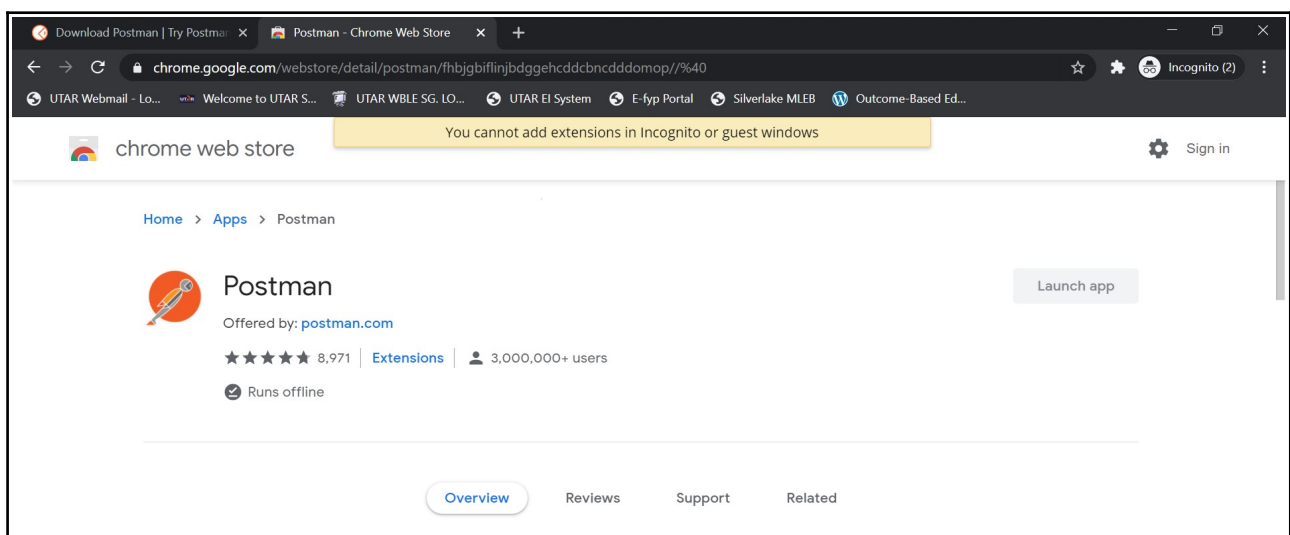
In this lab, we will explore the usage of client-side scripts using a JavaScript library. In this case, we will explore ReactJS library with Laravel with the continuation of previous laravelAuth web application that we have created for Laravel Authentication and Authorization.

Pre-Requisite

Install Postman application/software into your machine OR register to use Postman on the web OR install Postman as an extension/app for your Chrome Browser. This application is used to test API that will be created for the client side access to database.



<https://www.postman.com/downloads/>



<https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdgghehcdcbncdddomop/%40>

1. ReactJS Scaffolding

Previously, we had a VueJS scaffolding in laravelAuth web application project. Execute the following command in order to replace the previous scaffolding with React scaffolding.

```
php artisan ui react --auth
npm install
npm run dev
```

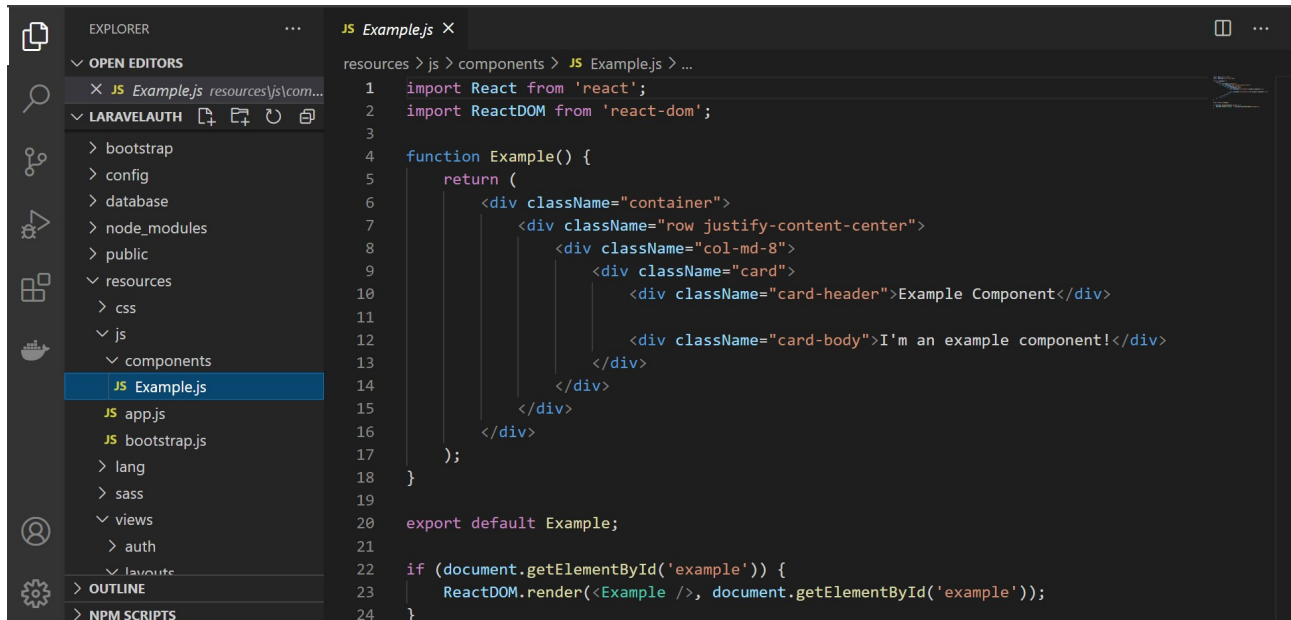


Figure 1: Example React Component.

2. Integrate React Component in Welcome Blade.

Let's start afresh by removing all of the <styles> and <div> (except for the login or register divisions to preserve the login and register links) within Welcome Blade Template as shown in Figure 2.

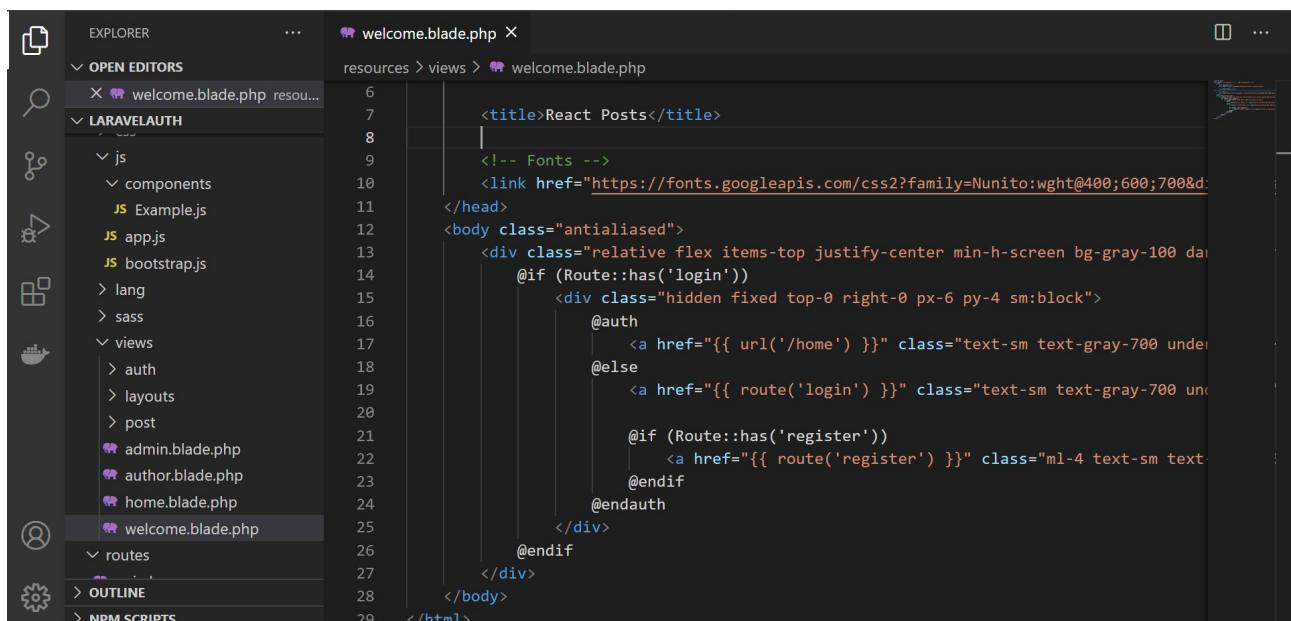
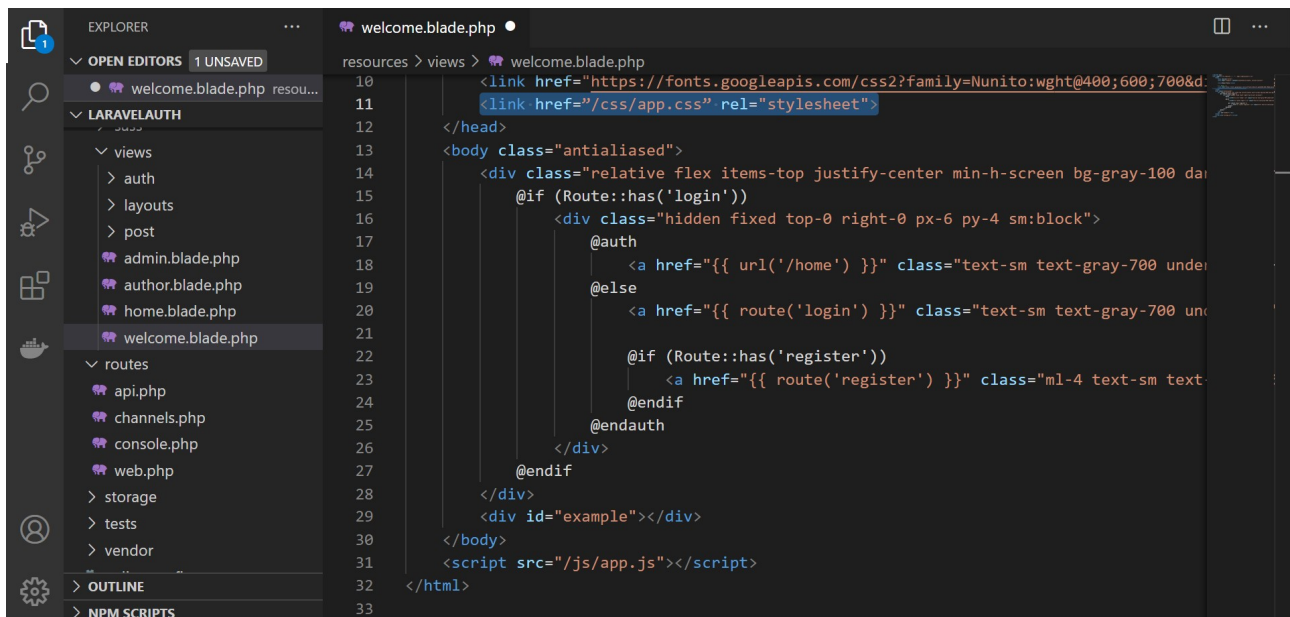


Figure 2: Example React Component.

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

Then, link Welcome Blade to the bootstrap (bootstrap file is in public\css\app.css), create a division to use React Example Component, and link to javascript (javascript file is in resources\js\app.js) as shown in Figure 3.



```
10 <link href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&d=
11 <link href="/css/app.css" rel="stylesheet">
12 </head>
13 <body class="antialiased">
14 <div class="relative flex items-top justify-center min-h-screen bg-gray-100 da
15 @if (Route::has('login'))
16 <div class="hidden fixed top-0 right-0 px-6 py-4 sm:block">
17 @auth
18 <a href="{{ url('/home') }}" class="text-sm text-gray-700 unde
19 @else
20 <a href="{{ route('login') }}" class="text-sm text-gray-700 unc
21
22 @if (Route::has('register'))
23 <a href="{{ route('register') }}" class="ml-4 text-sm text-
24 @endif
25 @endauth
26 </div>
27 @endif
28 </div>
29 <div id="example"></div>
30 </body>
31 <script src="/js/app.js"></script>
32 </html>
33
```

Figure 3: Modify Welcome Blade Template.

Host the web application and see to it that React Example Component is in the output as shown in Figure 4.

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

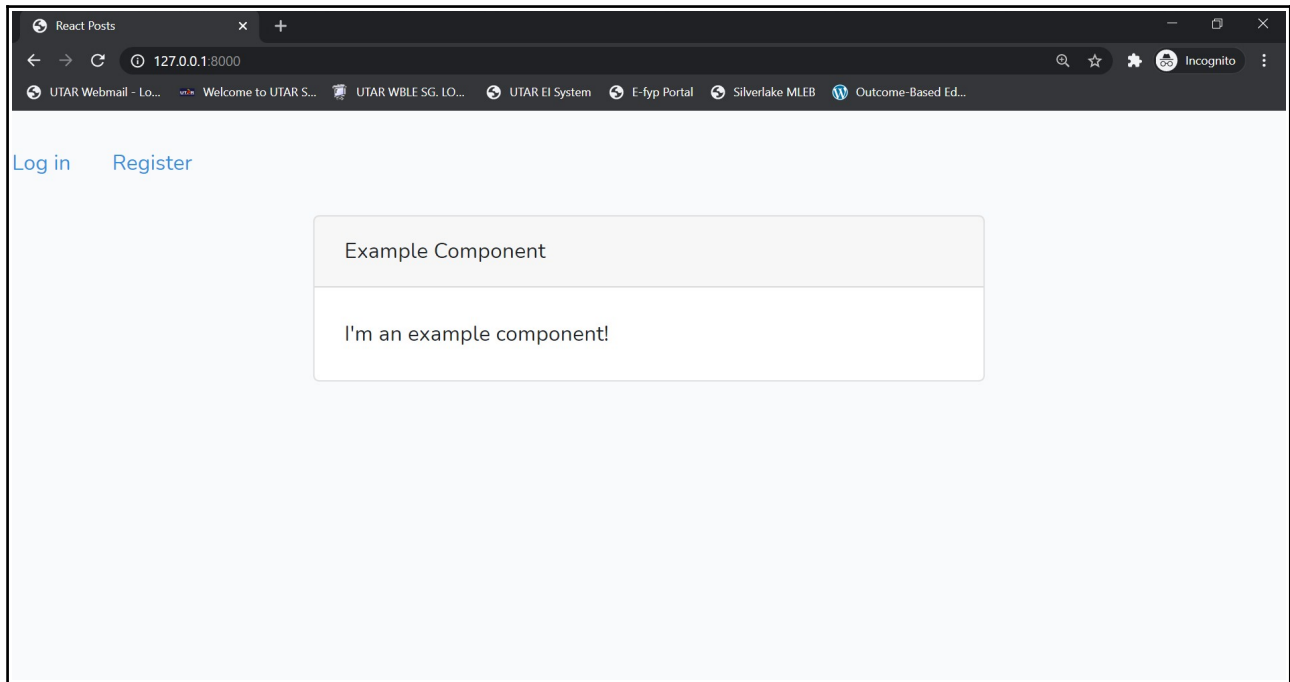
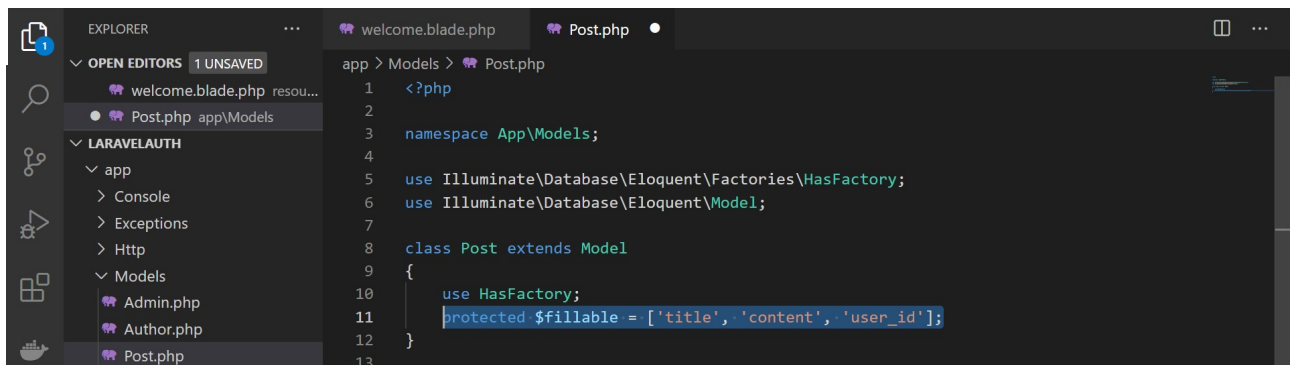


Figure 4: Example Component.

3. Define Post Model.

As a beginning to explore the CRUD concept on posts database table, modify the Post model by creating a mass assignment for title, content and user_id of the model.



4. Use Reactstrap.

In this project, we would like to use another React component called React Bootstrap 4 (<https://reactstrap.github.io/>). Install reactstrap with the following command line:

```
npm install --save reactstrap react react-dom
```

5. Define React Component.

Import “table” and “button” from reactstrap into react component as shown in Figure 5.



Figure 5: Import Table and Button from reactstrap.

Then, create a dummy table as shown in Figure 6, compile the assets and see the change in Welcome Blade Template.

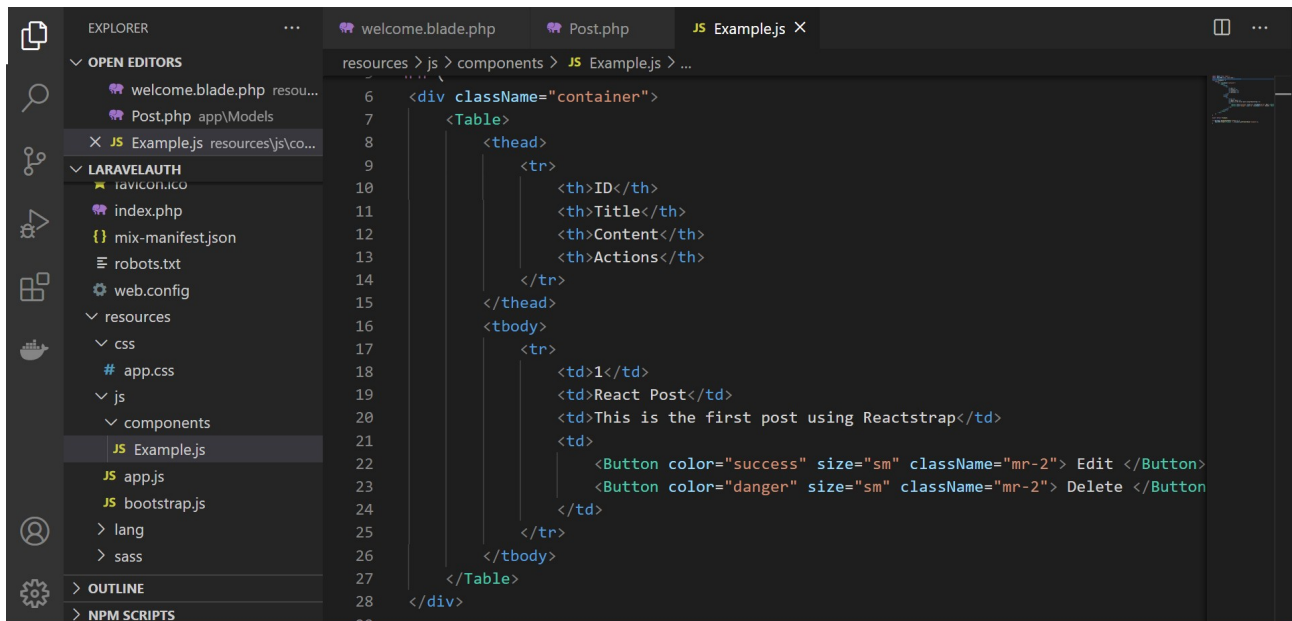


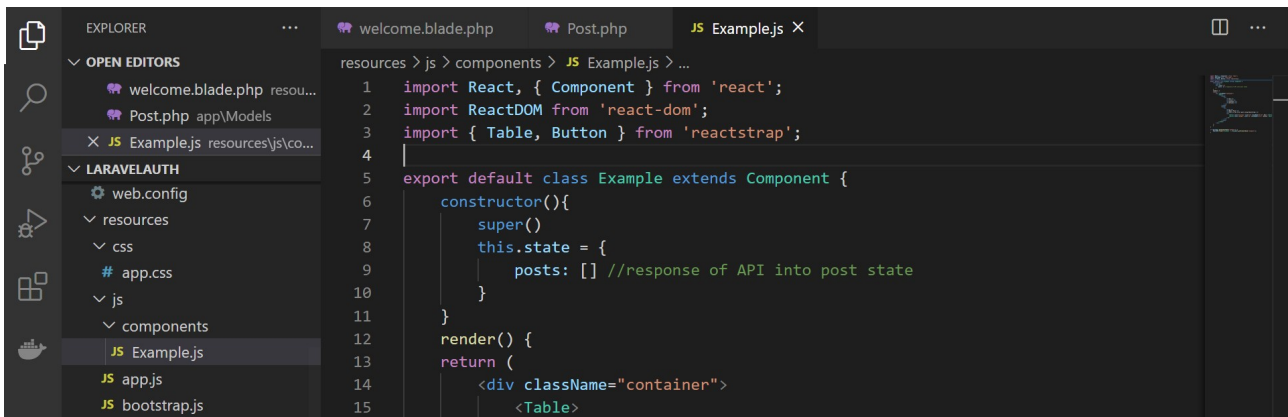
Figure 6: Define Table and Button to form a dummy table.

***Take note that the web application’s frontend will not change even though a modification is done on a React Component and is saved because every changes in a JS/CSS asset is to be compiled before the changes can be seen in front end. Thus, there are two ways of seeing the changes (manual compilation | automated compilation upon file save):*

```
npm run dev
npm run watch
```


6. Create React Component Constructor.

In order to initialize a default state for API response, create a constructor and modify function Example into a class Example as shown in Figure 7.



```

1  import React, { Component } from 'react';
2  import ReactDOM from 'react-dom';
3  import { Table, Button } from 'reactstrap';
4
5  export default class Example extends Component {
6    constructor(){
7      super()
8      this.state = {
9        posts: [] //response of API into post state
10     }
11   }
12   render() {
13     return (
14       <div className="container">
15         <Table>

```

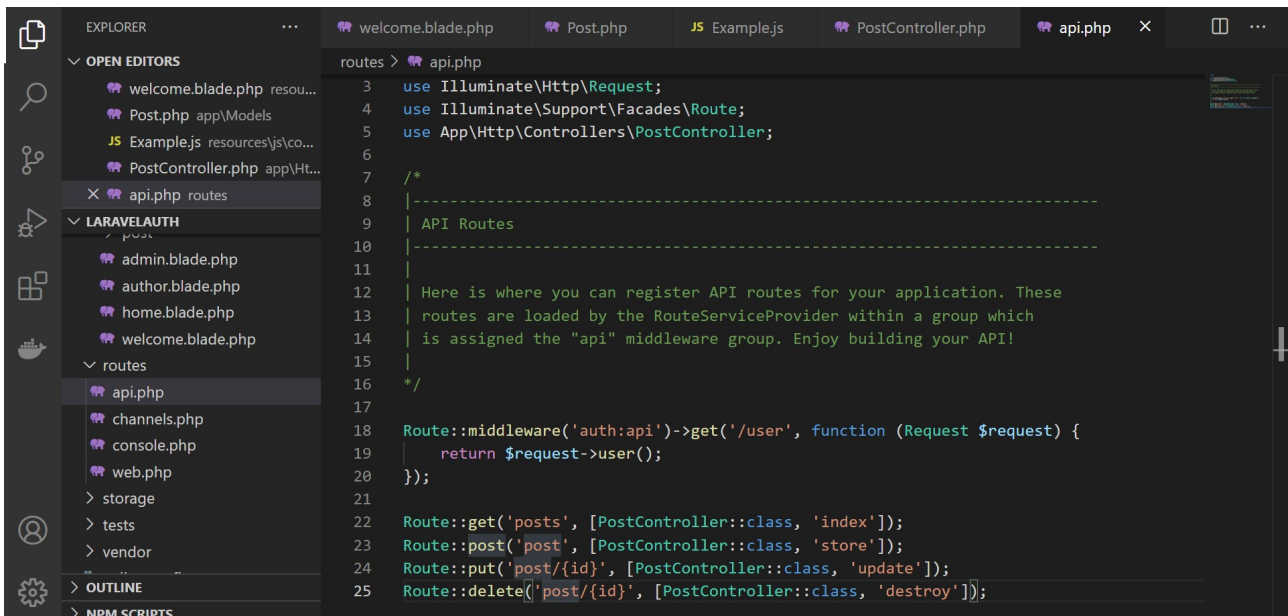
Figure 7: Define default response state and modify function to class.

7. Create Controllers.

In order to explore CRUD in Laravel ReactJS, CRUD controllers are required for the application logics. With the knowledge and understandings thus far, create controller functions in PostController for showing all posts, create new post, edit post and delete post.

8. Declare API endpoints.

API is needed to enable client-side / front end to interact to the database. Let's declare all the API endpoints in routes\api.php as shown in Figure 8.



```

3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5  use App\Http\Controllers\PostController;
6
7  /*
8   |-----
9   | API Routes
10  |-----
11  |
12  | Here is where you can register API routes for your application. These
13  | routes are loaded by the RouteServiceProvider within a group which
14  | is assigned the "api" middleware group. Enjoy building your API!
15  |
16  */
17
18  Route::middleware('auth:api')->get('/user', function (Request $request) {
19    return $request->user();
20  });
21
22  Route::get('posts', [PostController::class, 'index']);
23  Route::post('post', [PostController::class, 'store']);
24  Route::put('post/{id}', [PostController::class, 'update']);
25  Route::delete('post/{id}', [PostController::class, 'destroy']);

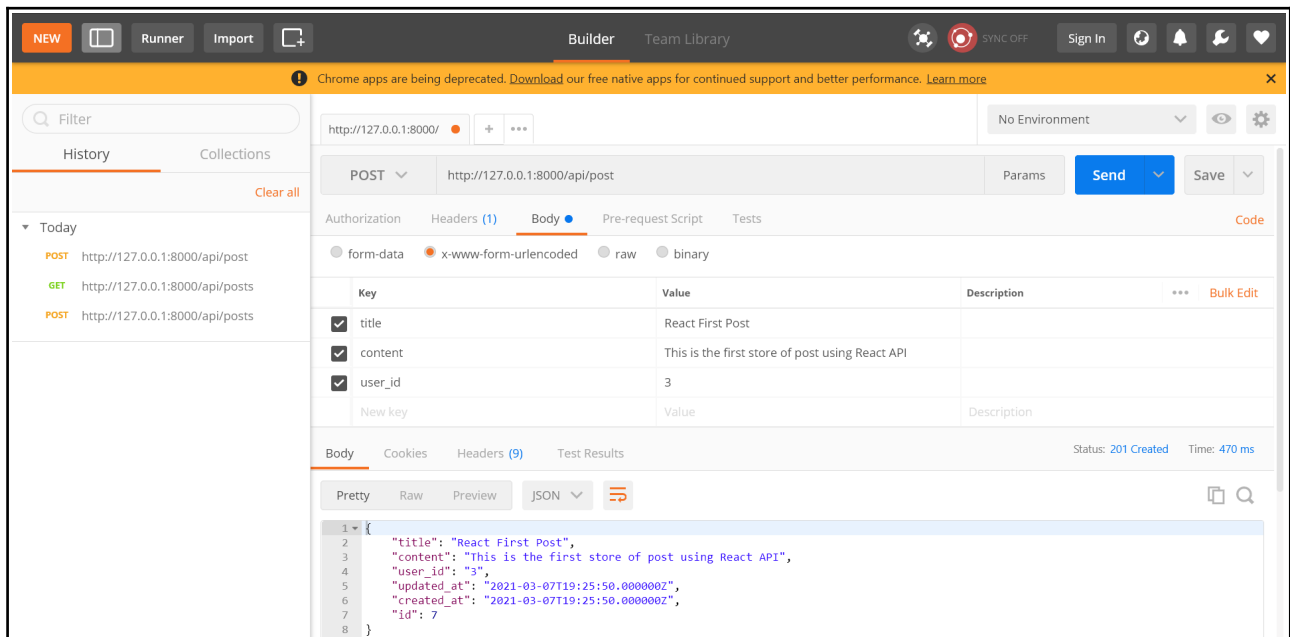
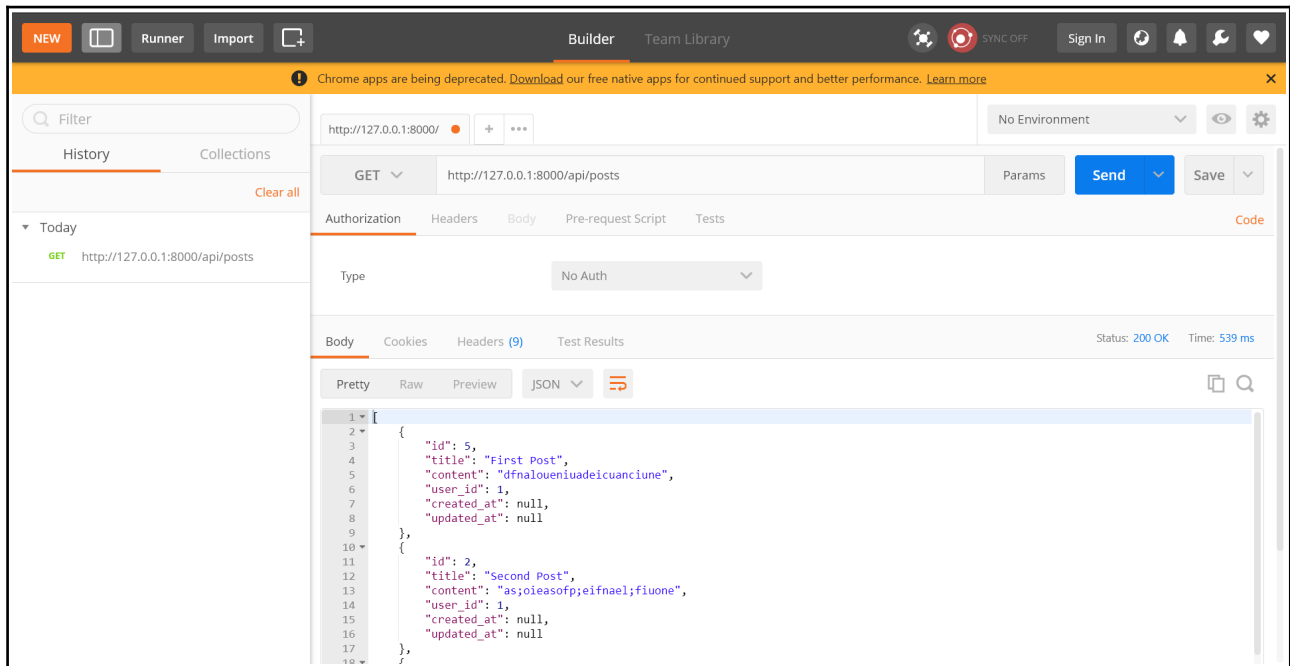
```

Figure 8: API endpoints.

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

9. Validate API.

Test the API endpoints in Postman to validate whether the API works successfully as shown in the following figures in Figure 9.



UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

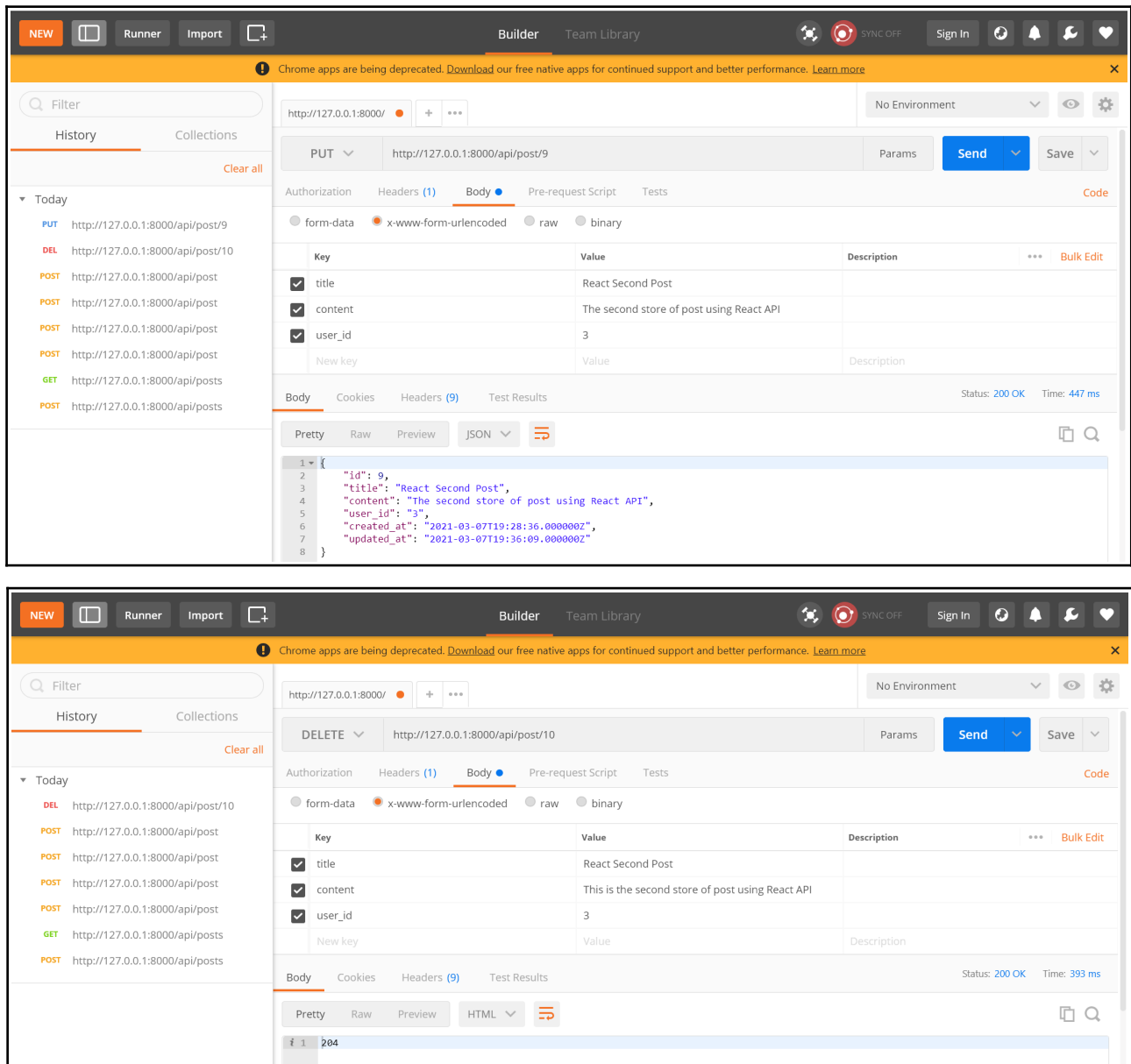


Figure 9 (Compilation): Test API in Postman.

10. Axios to enable API – React Component Communication.

We had previously tested that all the API endpoints are working properly. Now, we would like to install another package / dependency to enable React Component to be able to communicate with API using the following command line:

```
npm install axios --save
```

11. Define API – Component communication and front end to show all posts.

In order to enable the same operations done within Postman to be able to be done in front end, we need to define our React Component as shown below.

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

```
import React, { Component } from 'react';
import ReactDOM from 'react-dom';
import { Table, Button } from 'reactstrap';
import axios from 'axios';

export default class Example extends Component {
  constructor(){
    super()
    this.state = {
      posts: [] //response of API into post state
    }
  }
  loadPost(){
    axios.get('http://127.0.0.1:8000/api/posts').then((response) => {
      this.setState({
        posts:response.data
      })
    })
  }
  componentWillMount(){
    this.loadPost();
  }
  render() {
    let posts = this.state.posts.map((post) => {
      return(
        <tr key={post.id}>
          <td>{post.id}</td>
          <td>{post.title}</td>
          <td>{post.content}</td>
          <td>
            <Button color="success" size="sm" className="mr-2"> Edit </
Button>
            <Button color="danger" size="sm" className="mr-2"> Delete
</Button>
          </td>
        </tr>
      )
    })
    return (
      <div className="container">
        <Table>
          <thead>
            <tr>
              <th>ID</th>
              <th>Title</th>
              <th>Content</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {posts}
          </tbody>
        </Table>
      </div>
    );
  }
}

if (document.getElementById('example')) {
  ReactDOM.render(<Example />, document.getElementById('example'));
}
```

