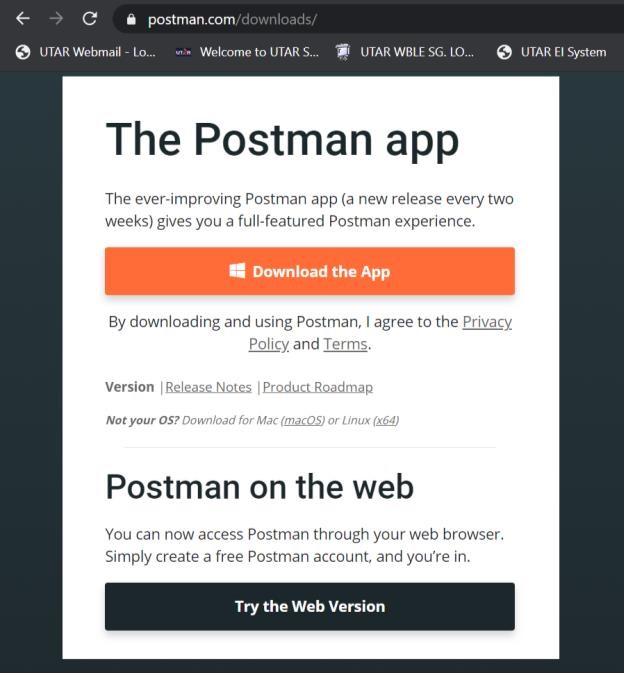
**Practical 7 : Laravel ReactJS CRUD with RESTful API (Part I).**

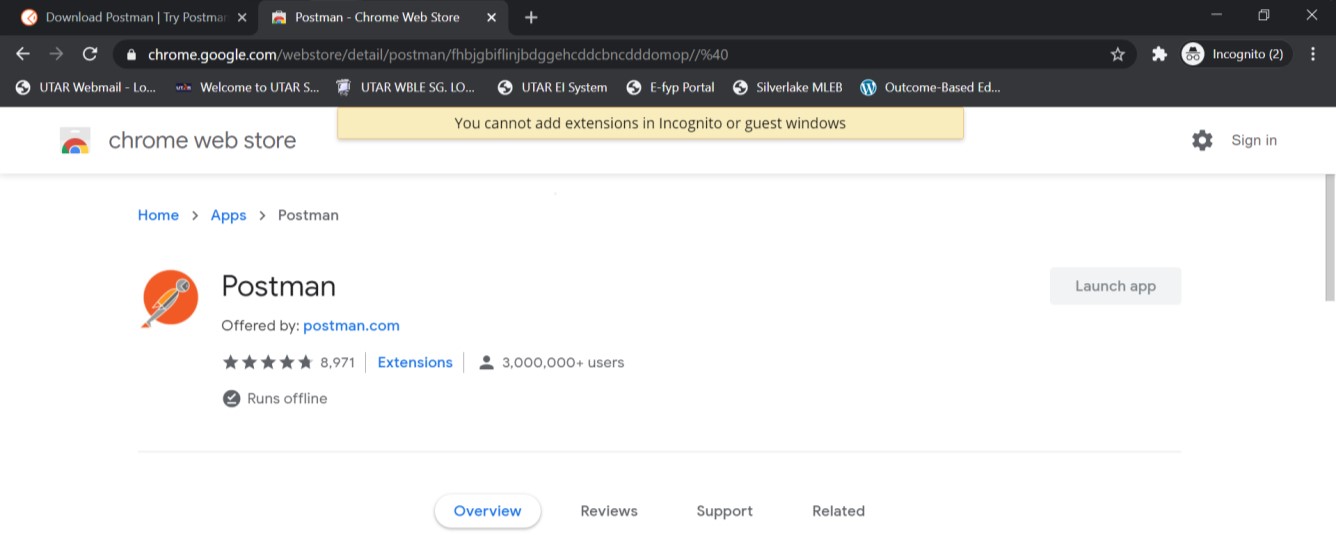
In this lab, we will explore the usage of client-side scripts using a JavaScript library. In this case, we will explore ReactJS library with Laravel with the continuation of previous laravelAuth web application that we have created for Laravel Authentication and Authorization.

# 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/fhbjgbiflinjbdggehcddcbncdddomop//%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** |  |

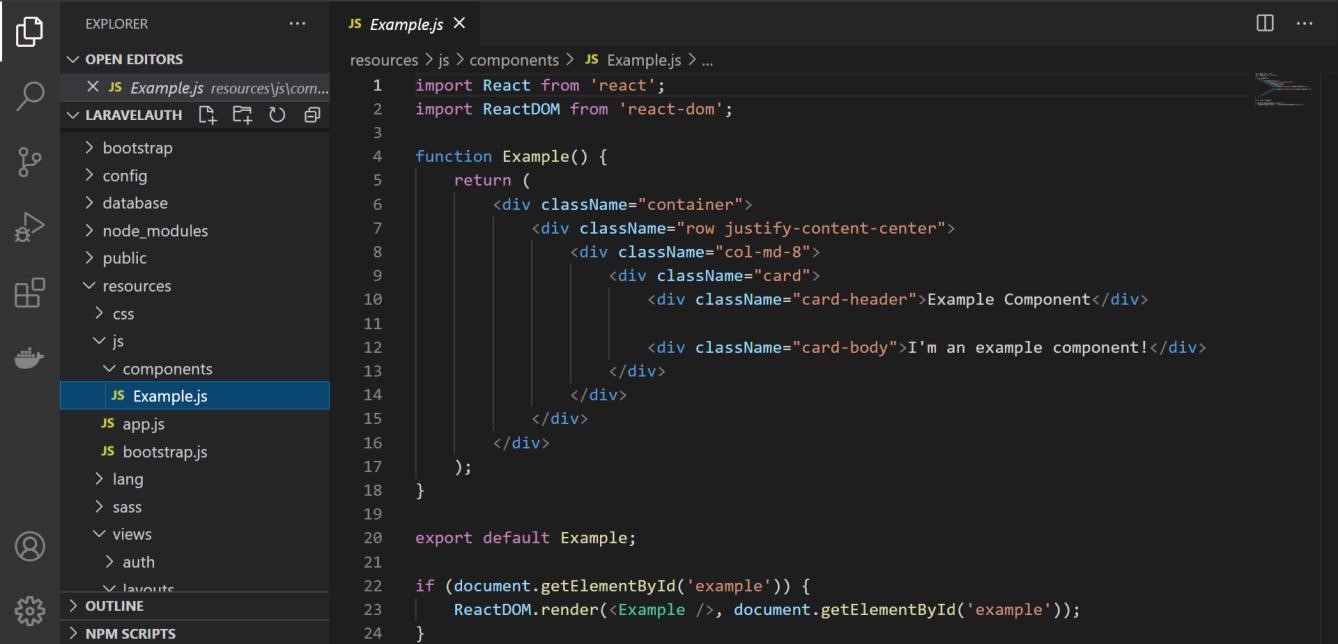


Figure 1: Example React Component.

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  function Example() {      return (          <div className="container">              <div className="row justify-content-center">                  <div className="col-md-8">                      <div className="card">                          <div className="card-header">Example Component</div>                          <div className="card-body">I'm an example component!</div>                      </div>                  </div>              </div>          </div>      );  }  export default Example;  if (document.getElementById('example')) {      ReactDOM.render(<Example />, document.getElementById('example'));  } |

1. **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.

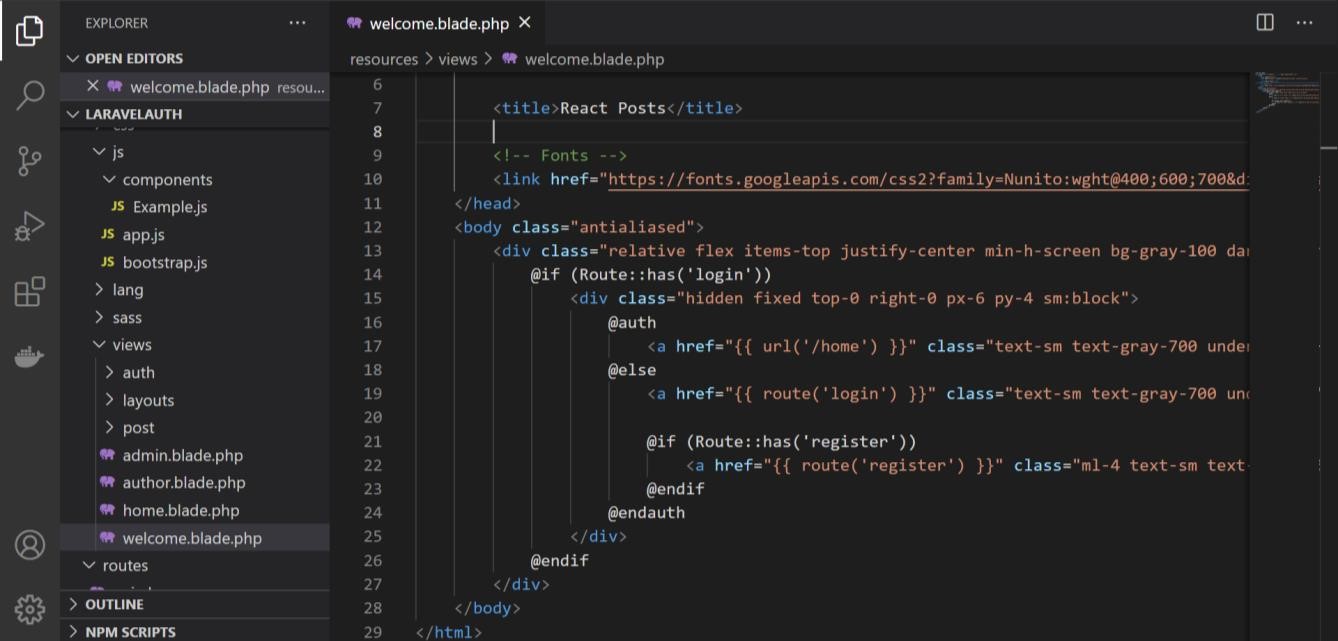


Figure 2: Example React Component.

|  |
| --- |
| <!DOCTYPE html>  <html lang="{{ str\_replace('\_', '-', app()->getLocale()) }}">      <head>          <meta charset="utf-8">          <meta name="viewport" content="width=device-width, initial-scale=1">            <title>Laravel</title>            <!-- Fonts -->          <link href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swap" rel="stylesheet">          <link href="/css/app.css" rel="stylesheet">          <!-- Styles -->        </head>      <body class="antialiased">          <div class="relative flex items-top justify-center min-h-screen bg-gray-100 dark:bg-gray-900 sm:items-center py-4 sm:pt-0">              @if (Route::has('login'))                  <div class="hidden fixed top-0 right-0 px-6 py-4 sm:block">                      @auth                            <a href="{{ url('/home') }}" class="text-sm text-gray-700 dark:text-gray-500 underline">Home</a>                      @else                          <a href="{{ route('login') }}" class="text-sm text-gray-700 dark:text-gray-500 underline">Log in</a>                            @if (Route::has('register'))                              <a href="{{ route('register') }}" class="ml-4 text-sm text-gray-700 dark:text-gray-500 underline">Register</a>                          @endif                      @endauth                  </div>              @endif          </div>          <div id="example"></div>      </body>      <script src="/js/app.js"></script>  </html> |

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.

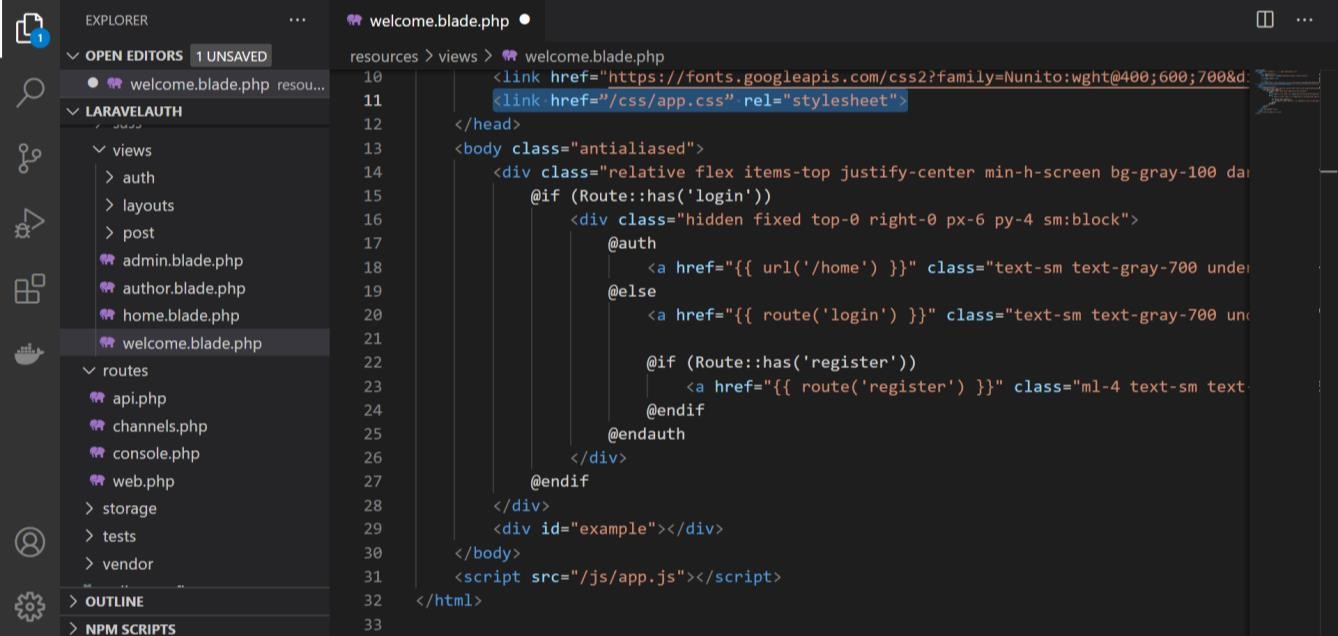
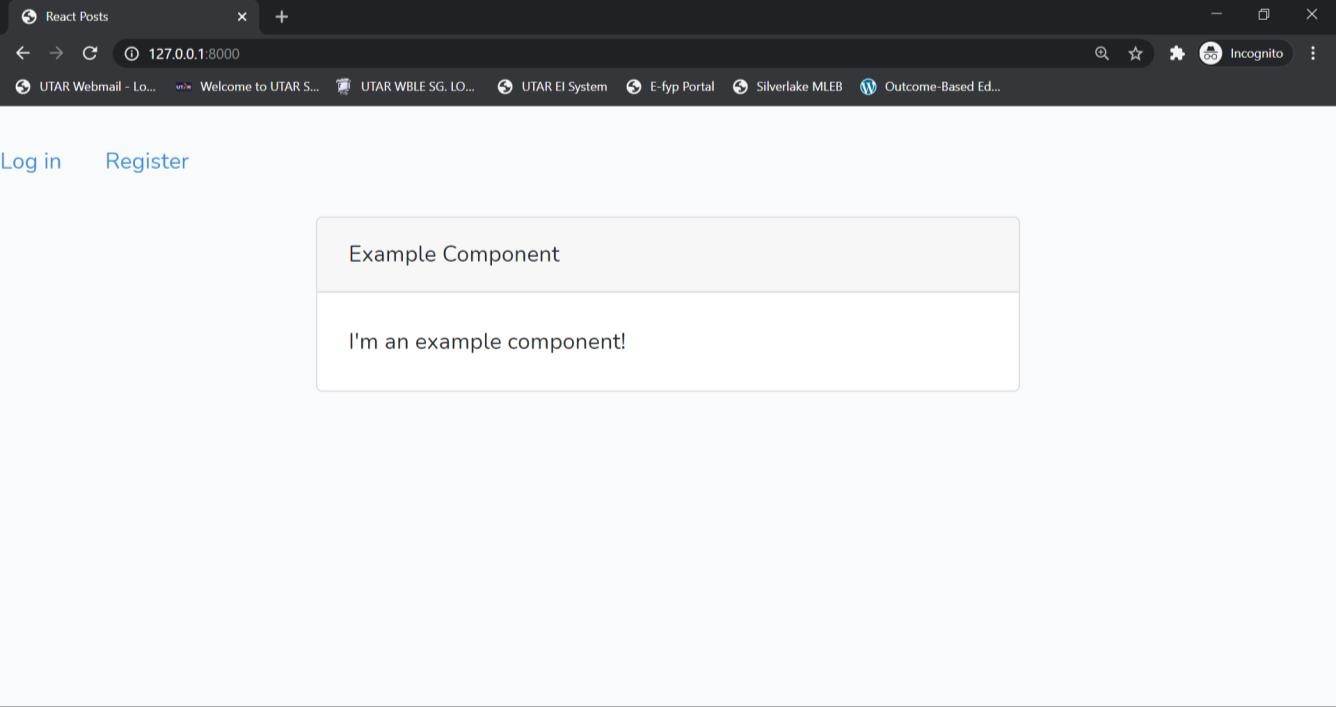


Figure 3: Modify Welcome Blade Template.

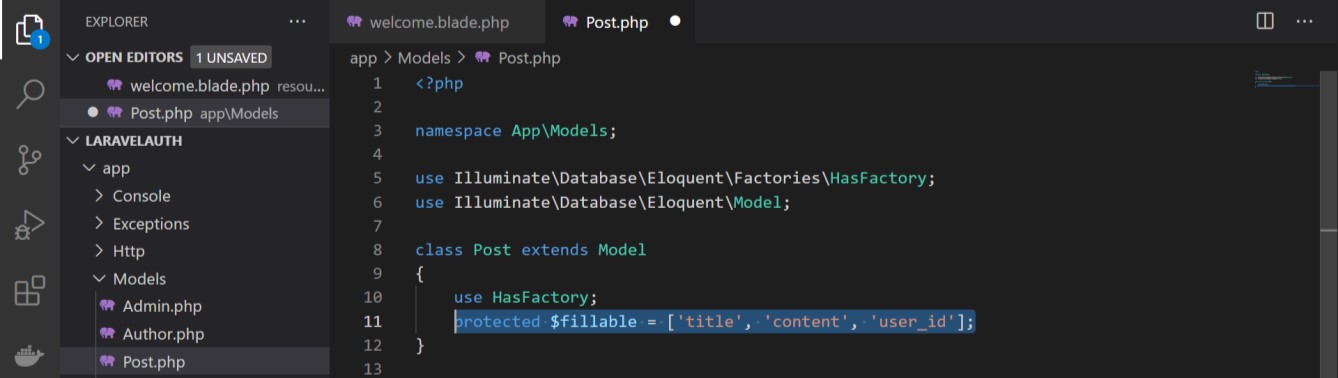
Host the web application and see to it that React Example Component is in the output as shown in Figure 4.

Figure 4: Example Component.



1. **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.



|  |
| --- |
| <?php    namespace App\Models;    use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;    class Post extends Model  {      use HasFactory;      protected $fillable=['title','content','user\_id'];        //public $table="post";        public $timestamps=false;  } |

1. **Use Reactstrap.**

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

**npm install reactstrap react react-dom**

## npm install --save bootstrap

\*\*import bootstrap in application code:

**import 'bootstrap/dist/css/bootstrap.min.css';**

1. **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.

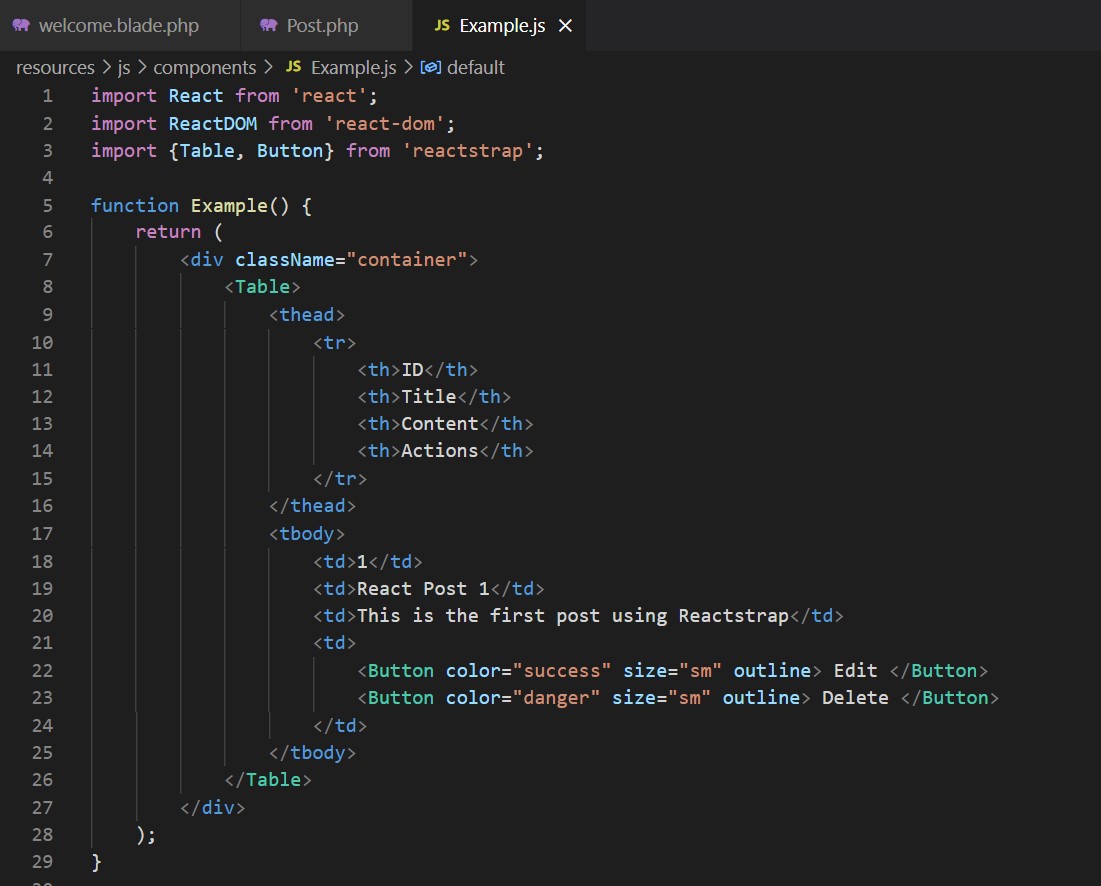


Figure 6: Define Table and Button to form a dummy table.

|  |
| --- |
| Example1.js  import React from 'react';  import ReactDOM from 'react-dom';  import { Table, Button } from 'reactstrap';  function Example() {      return (          <div className="container">              <Table>                  <thead>                      <tr>                          <th>ID</th>                          <th>Title</th>                          <th>Content</th>                          <th>Action</th>                      </tr>                  </thead>                  <tbody>                      <tr>                          <td>1</td>                          <td>React Post 1</td>                          <td>This is the first reactstrap Post </td>                          <td>                              <Button color="success" size="sm">Edit</Button>                              <Button color="danger" size="sm">Delete</Button>                          </td>                      </tr>                  </tbody>              </Table>          </div>      );  }  export default Example;  if (document.getElementById('example')) {      ReactDOM.render(<Example />, document.getElementById('example'));  } |

|  |  |  |
| --- | --- | --- |
| *\*\*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** | |  |

1. **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.

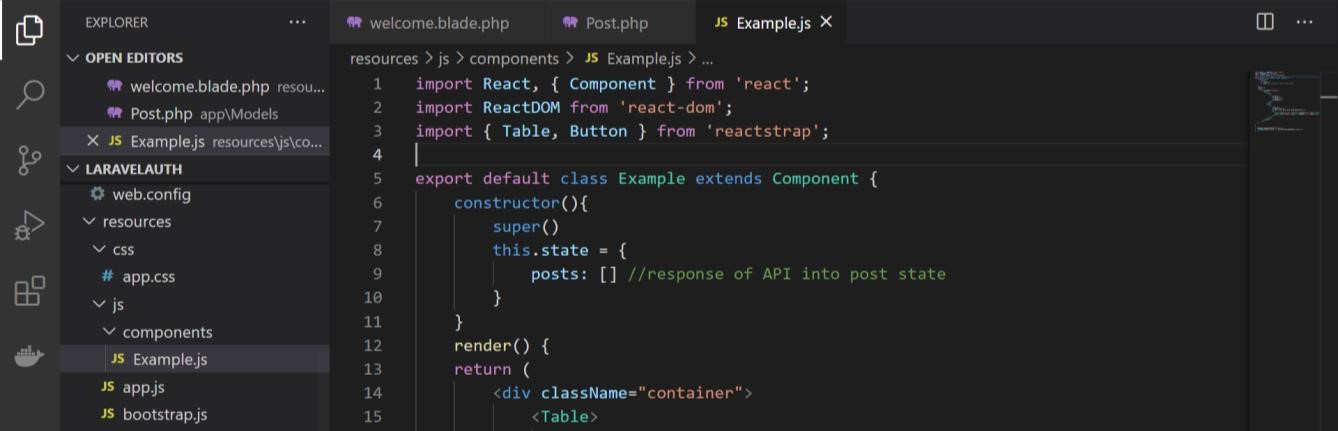


Figure 7: Define default response state and modify function to class.

1. **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.

|  |
| --- |
| <?php  namespace App\Http\Controllers;  use Illuminate\Support\Facades\Gate;  use Illuminate\Http\Request;  use App\Models\Post;  use App\Models\User;  class PostController extends Controller    {      public function index()      {          //$posts = Post::all();          //return view("/posts/index", ["posts"=> $posts]);          return Post::all();      }      public function show()      {          $posts = Post::all();          return view("/posts/show", ["posts"=> $posts]);        }        //public function destroy(Post $post)      public function destroy($id)      {         //$post->delete();         //return redirect("/posts/index");         $post=Post::findOrFail($id);         $post->delete();         return 204;      }      public function create()      {          if (Gate::allows('isAuthor')) {              dd('Author allowed');          } else              dd('You are not an Author');      }      public function edit()      {          if (Gate::allows('isAuthor')) {              dd('Author allowed');          } else              dd('You are not an Author');      }      public function delete()      {          if (Gate::allows('isAdmin')) {              dd('Admin allowed');          } else              dd('You are not Admin');      }      public function store(Request $request)      {          return Post::create($request->all());      }      public function update(Request $request, $id)      {          $post=Post::findOrFail($id);          $post->update($request->all());          return $post;      }  } |

1. **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.

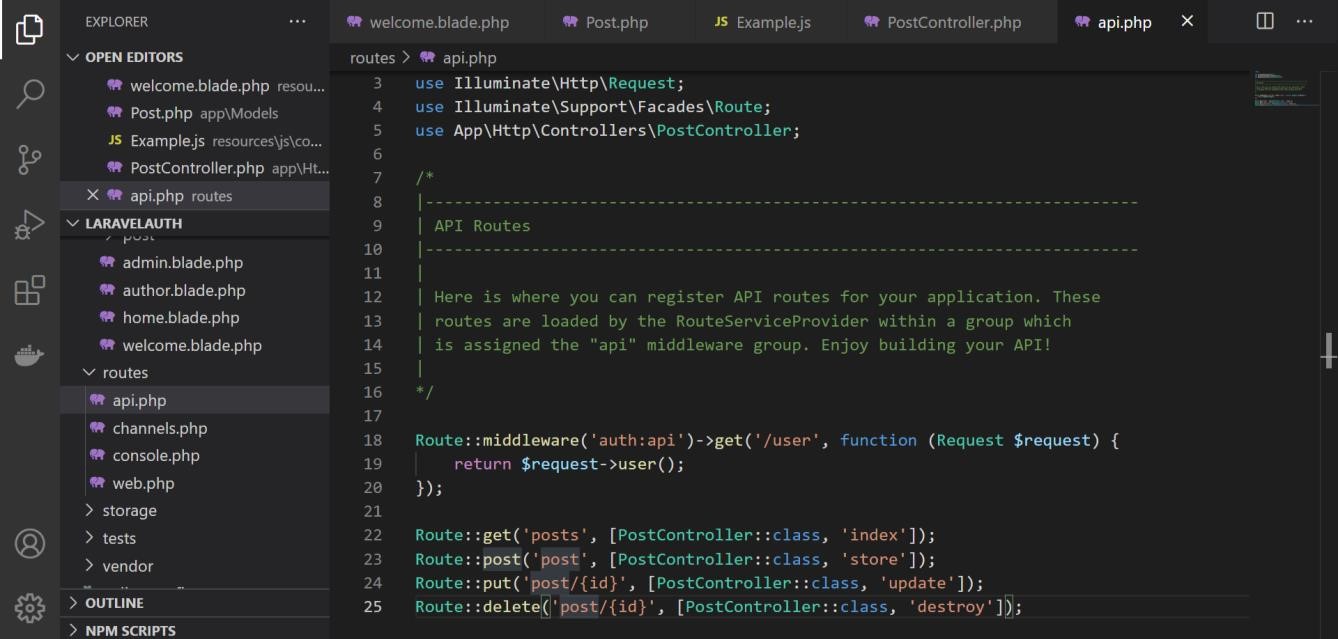
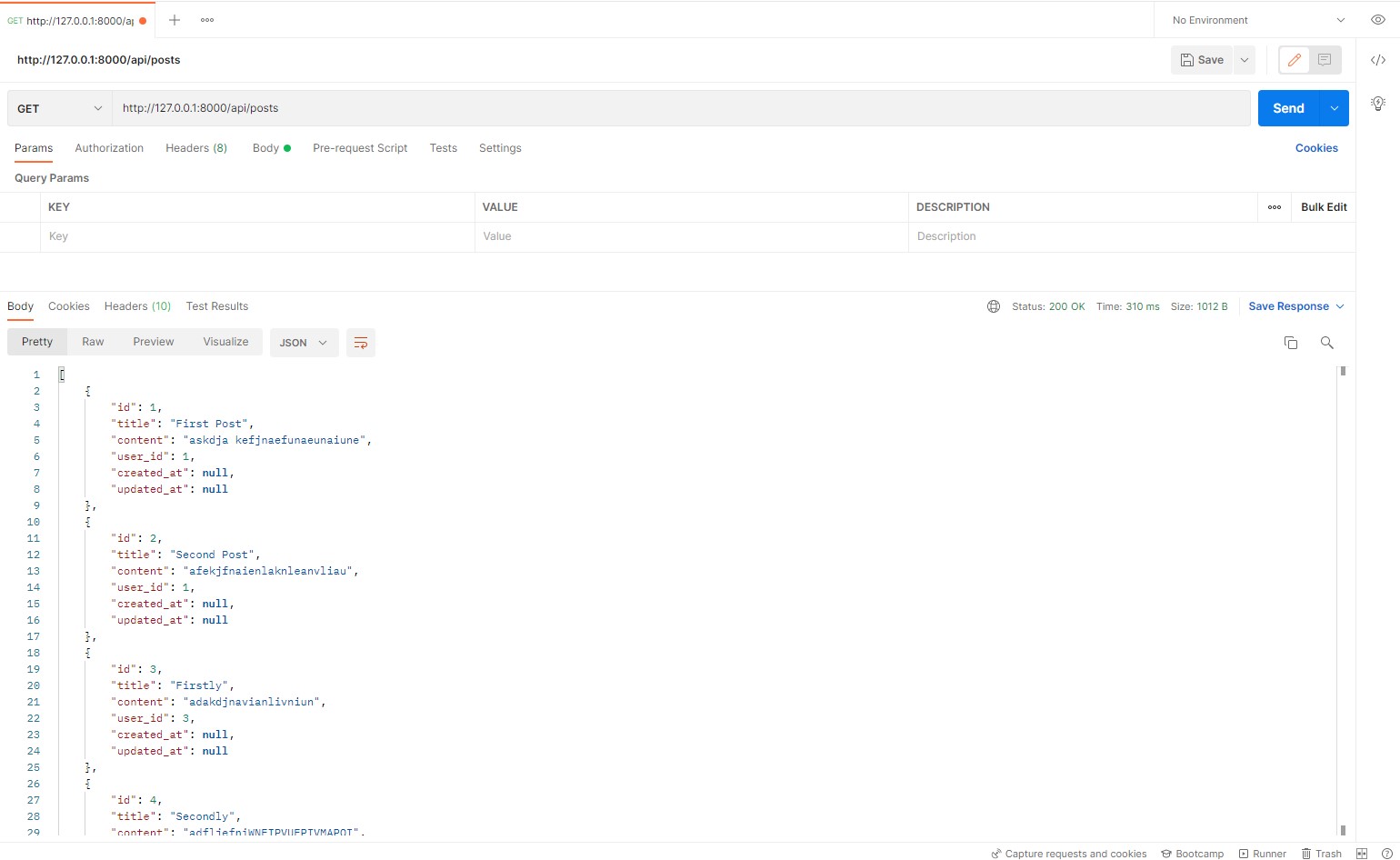


Figure 8: API endpoints.

|  |
| --- |
| <?php    use Illuminate\Http\Request;  use Illuminate\Support\Facades\Route;  use App\Http\Controllers\PostController;  use App\Http\Controllers\Controller;  /\*  |--------------------------------------------------------------------------  | API Routes  |--------------------------------------------------------------------------  |  | Here is where you can register API routes for your application. These  | routes are loaded by the RouteServiceProvider within a group which  | is assigned the "api" middleware group. Enjoy building your API!  |  \*/    Route::middleware('auth:sanctum')->get('/user', function (Request $request) {        return $request->user();  });    Route::get('/posts', [PostController::class, 'index']);//http://localhost:8000/api/posts  Route::post('/post', [PostController::class, 'store']); //http://localhost:8000/api/post  Route::put('/post/{id}', [PostController::class, 'update']);//http://localhost:8000/api/post  Route::delete('/post/{id}', [PostController::class, 'destroy']);//http://localhost:8000/api/post |

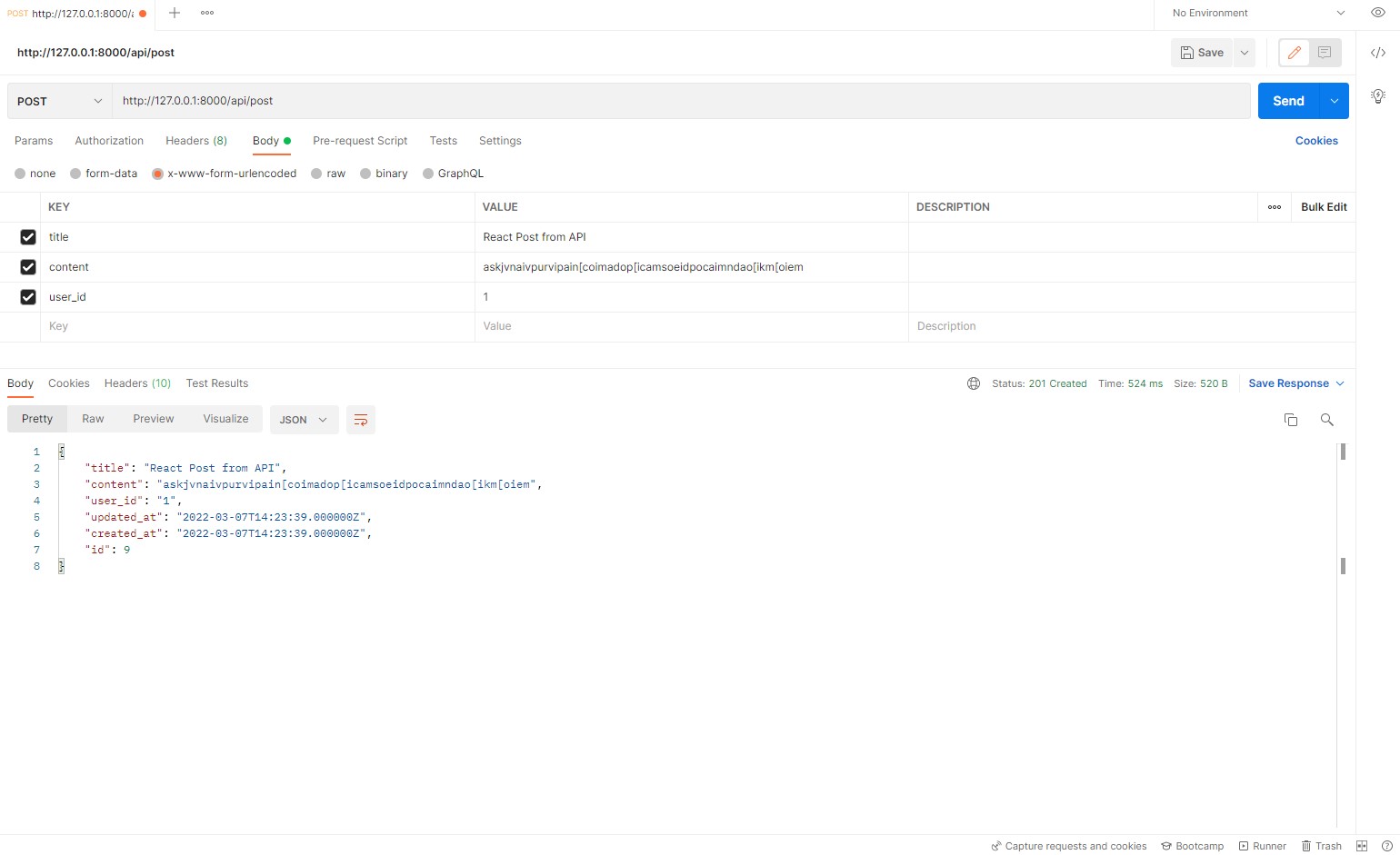
1. **Validate API.**

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



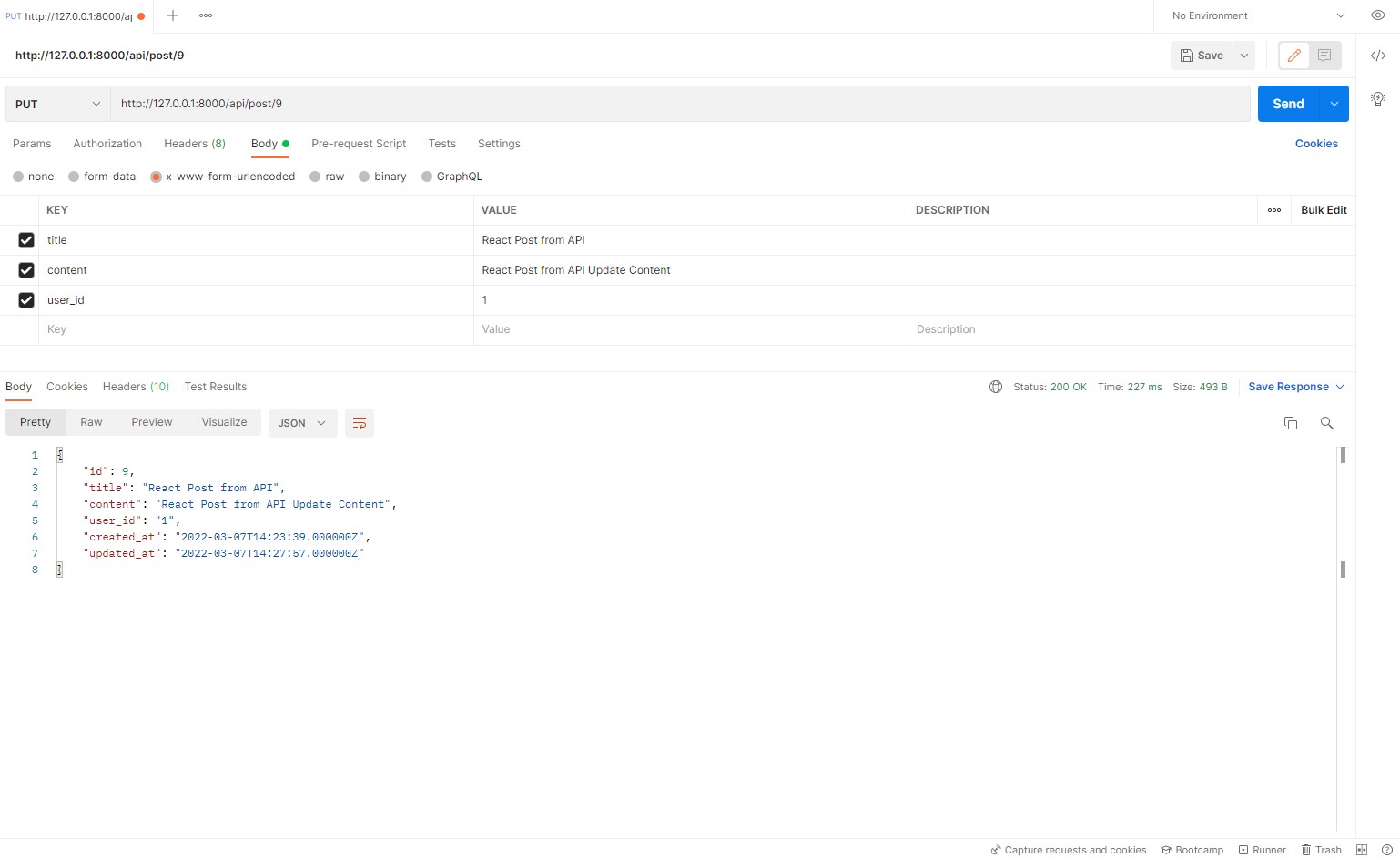
A screenshot of a computer

Description automatically generated



A screenshot of a computer

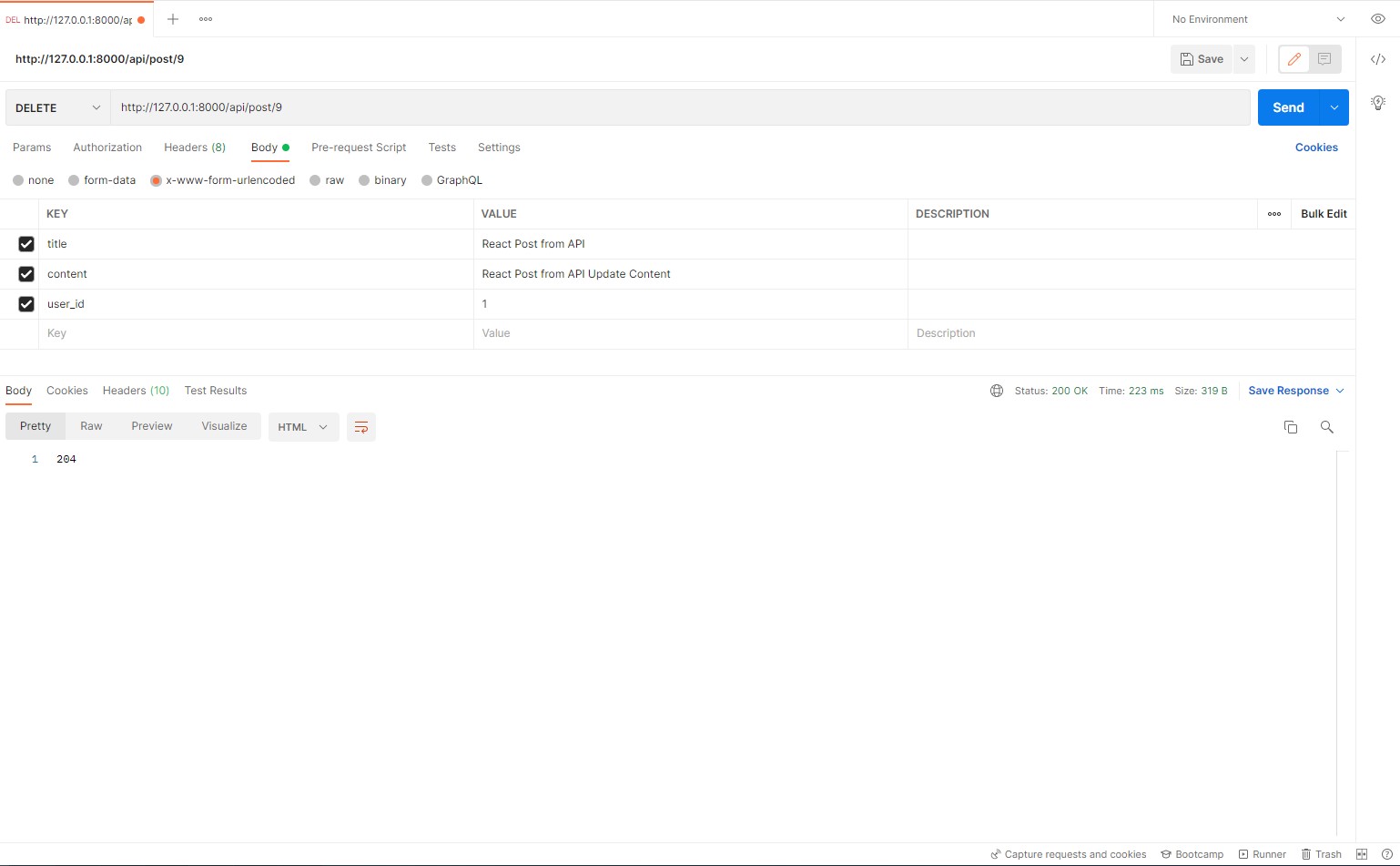
Description automatically generated



A screenshot of a computer

Description automatically generated

Figure 9 (Compilation): Test API in Postman.



A screenshot of a computer

Description automatically generated

1. **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

**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.

|  |
| --- |
| 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'));  } |