ReactJS+Laravel Restfull API

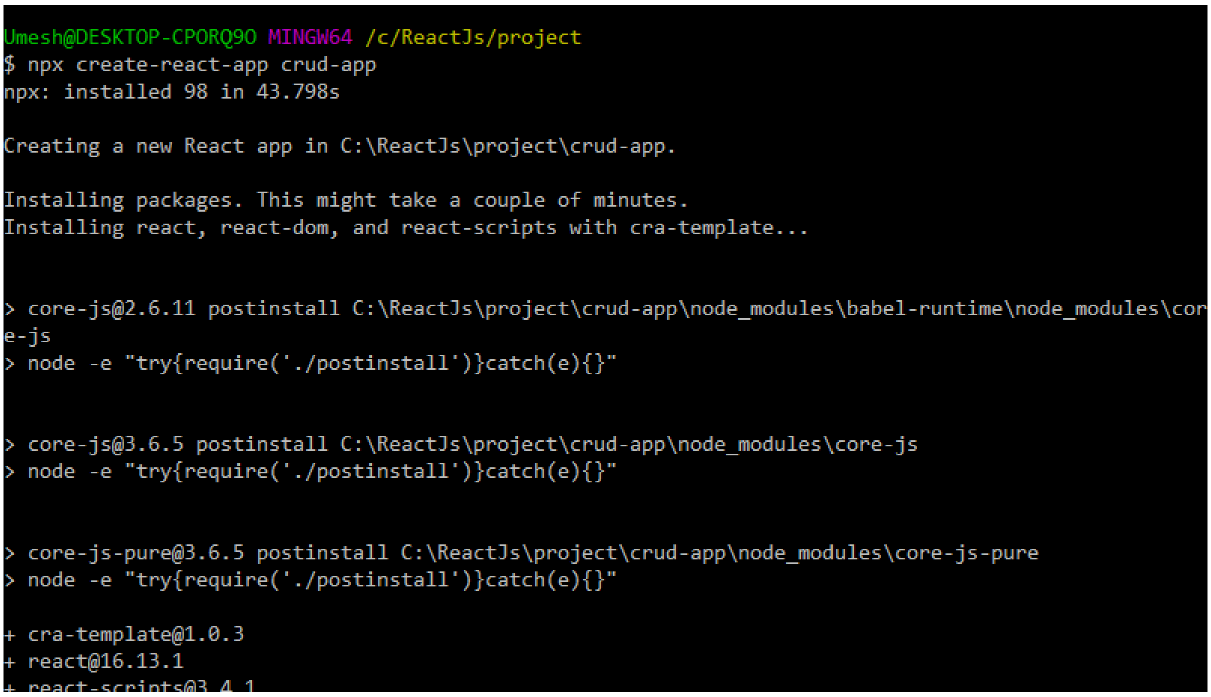
In React.js you can create an interactive UI by combining the different components together. In this post, we are going to create a Reactjs CRUD application. For managing the data and database operations, we will be using the RESTful APIs. You are already familiar with the term RESTful APIs. For the back-end, we will be using the Laravel 7 with MySQL Database. The APIs makes any application to optimized and a light-weight. Basically, in an API call, a request is handled in a different way. There are different types of request types. So, in this post, we will be covering both like the front-end app using the React.js and also will create the RESTful APIs in the Laravel 7. So, let's start with Reactjs crud example.

Create a Reactjs CRUD App

Open the terminal or command prompt and hit the below command to create a new react app.

```
npx create-react-app crud-app
```

The above command will initiate to create a react app inside the crud-app folder.



```
Umesh@DESKTOP-CPORQ90 MINGW64 /c/ReactJs/project
$ npx create-react-app crud-app
npx: installed 98 in 43.798s

Creating a new React app in C:\ReactJs\project\crud-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


> core-js@2.6.11 postinstall C:\ReactJs\project\crud-app\node_modules\babel-runtime\node_modules\cor
e-js
> node -e "try{require('./postinstall')}catch(e){}"


> core-js@3.6.5 postinstall C:\ReactJs\project\crud-app\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"


> core-js-pure@3.6.5 postinstall C:\ReactJs\project\crud-app\node_modules\core-js-pure
> node -e "try{require('./postinstall')}catch(e){}"

+ cra-template@1.0.3
+ react@16.13.1
+ react-scripts@3.4.1
```

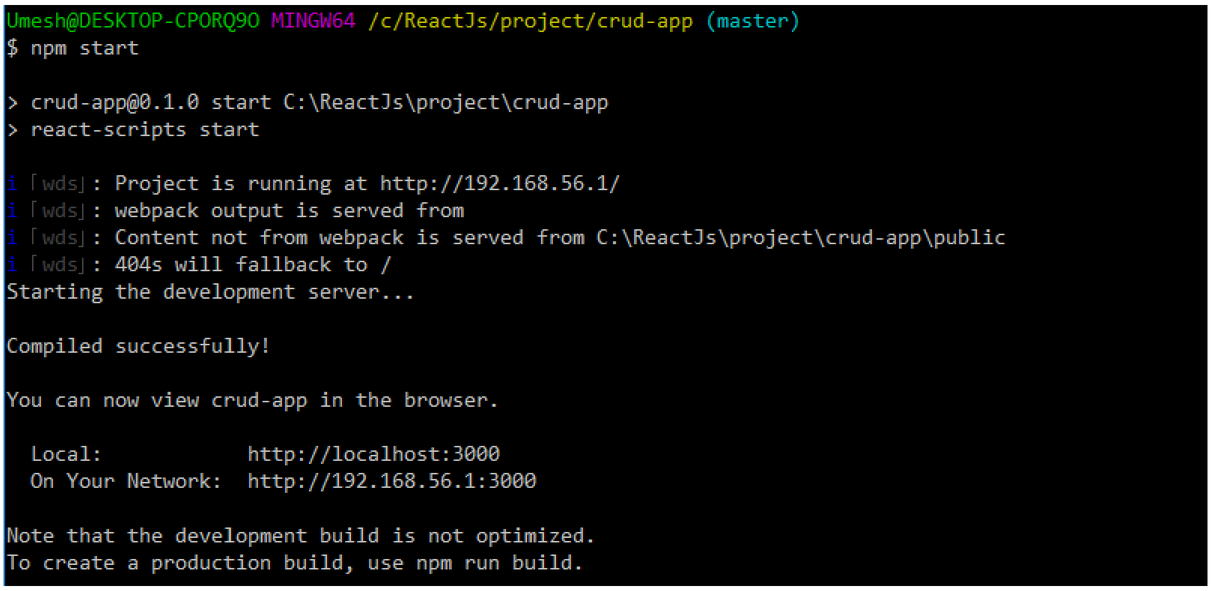
Create React CRUD App

It will take couple of minutes to finish the installation of necessary libraries. Once, app has been created, navigate inside the project folder.

```
cd crud-app
```

Now, run the application by running the npm command.

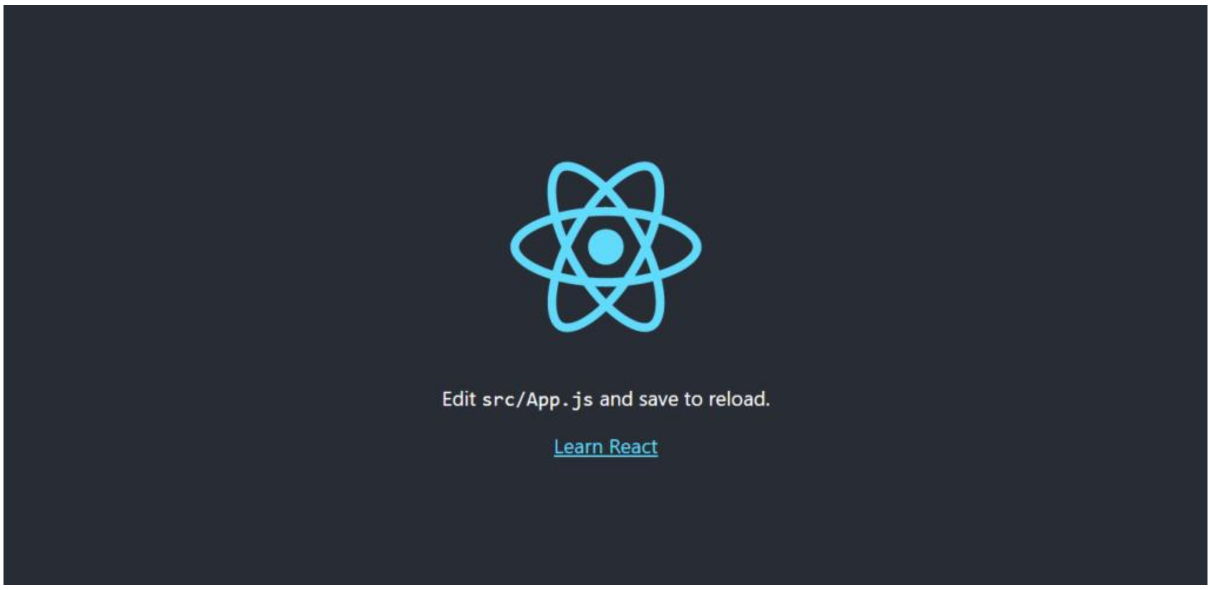
```
npm start
```



```
Umesh@DESKTOP-CPORQ90 MINGW64 /c/ReactJs/project/crud-app (master)
$ npm start

> crud-app@0.1.0 start C:\ReactJs\project\crud-app
> react-scripts start

i ｢wds｣: Project is running at http://192.168.56.1/
i ｢wds｣: webpack output is served from
i ｢wds｣: Content not from webpack is served from C:\ReactJs\project\crud-app\public
i ｢wds｣: 404s will fallback to /
Starting the development server...

Compiled successfully!

You can now view crud-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Run React App

The Reactjs crud application will be running on the default port 3000. The default page is showing below.



Edit `src/App.js` and save to reload.

Learn React

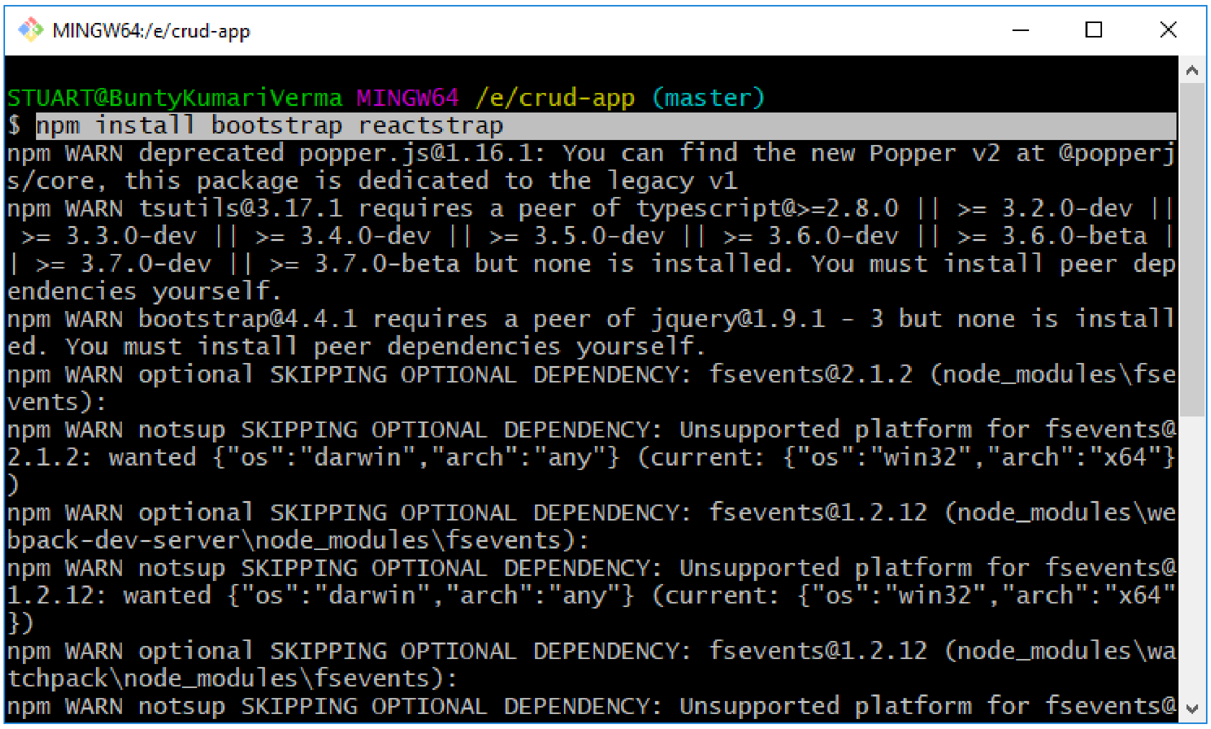
Running the default page in React app

Install Bootstrap and Reactstrap in Reactjs CRUD App

In this project, I am going to use Bootstrap and Reactstrap. For installing the Bootstrap and Reactstrap, we will be using the npm command.

```
npm install bootstrap reactstrap
```

The above command will add the Bootstrap CSS and JS file globally inside the **Reactjs CRUD** application.



```
STUART@BuntyKumariVerma MINGW64 /e/crud-app (master)
$ npm install bootstrap reactstrap
npm WARN deprecated popper.js@1.16.1: You can find the new Popper v2 at @popperj
s/core, this package is dedicated to the legacy v1
npm WARN tsutils@3.17.1 requires a peer of typescript@>=2.8.0 || >= 3.2.0-dev ||
 >= 3.3.0-dev || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta |
| >= 3.7.0-dev || >= 3.7.0-beta but none is installed. You must install peer dep
endencies yourself.
npm WARN bootstrap@4.4.1 requires a peer of jquery@1.9.1 - 3 but none is install
ed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\fse
vents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@
2.1.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"}
)
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.12 (node_modules\we
bpack-dev-server\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@
1.2.12: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"
})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.12 (node_modules\wa
tchpack\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@
```

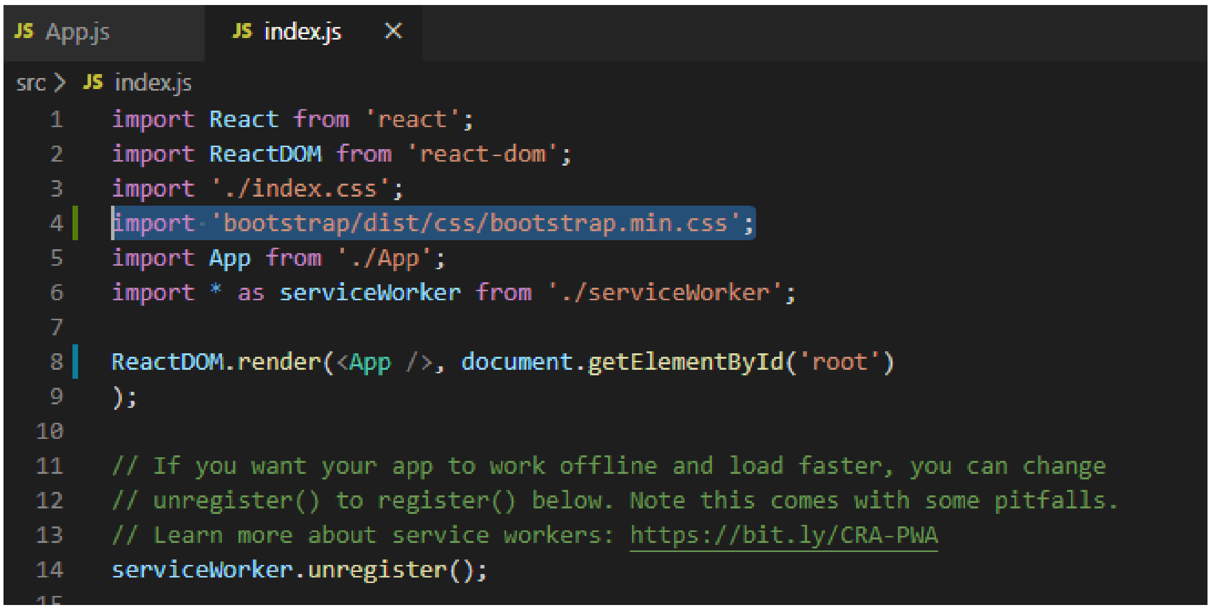
Install Bootstrap in React.js

[Create a CRUD Application in Laravel 7]

After completing the installation, let’s import the **Bootstrap CSS** in our application. Basically, we will be including the globally that’s why we will import it in the **src/index.js** file of our application.

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

Here, I have imported the Bootstrap CSS file in the **index.js** file.



```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root')
);

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

Import Bootstrap in React.js

So, for the CRUD application, we will have to create the RESTful APIs. So, first of all, we will create the APIs in Laravel 7. Then we'll come back to the React app.

Create RESTful APIs For CRUD App in Laravel 7

For creating the Laravel application, I am not going to explain in the brief. Because I have already shared too many posts in the Laravel 6 and Laravel 7 category. So, you can go through it for more details. This is a React app tutorial, so, I will focus on React. But, I will cover the necessary steps to create the APIs in Laravel 7.

I assume, you already familiar with how to start with Laravel and what tools you required to create a project.

So, I will be focusing on the RESTful APIs for the CRUD application.

Create a New Laravel Project For CRUD API

```
composer create-project --prefer-dist laravel/laravel laravel-crud-api
```

The above command will start the installation of necessary libraries using the composer. It will take couple of minutes to add the required files. So, just wait to finish it.

Create and Configure Database

For the database, I will be using MySQL. So, just open the MySQL command prompt and create a new database there.

```
CREATE DATABASE laravel7_crud_api;
```

So, the database has been created. Now, let's configure the details inside the project to create the connection.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel7_crud_api
DB_USERNAME=root
DB_PASSWORD=root
```

Create Model and Migration

I am going to work on the student module so for that I will be creating the model and migration for the students. So, let's do that.

```
php artisan make:model Student --migration
```

Here, model and migration have been created. Now, move to the **create-student.php** migration file that will be located inside the **database/migrations** file. After that, paste the below schema for the student table.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateStudentsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('students', function (Blueprint $table) {
            $table->id();
            $table->string("first_name")->nullable();
            $table->string("last_name")->nullable();
            $table->string("full_name")->nullable();
            $table->string("email")->nullable();
            $table->string("phone")->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('students');
    }
}
```

In the next step, we will migrate the above schema into the database. After migrating the schema the table will be created inside the database. So, let’s migrate the table by artisan command.

```
php artisan migrate
```

Once, the tables are migrated, we will add the fillable data in the Student Model. This fillable data will be managed through the controller into the tables.

How to Implement Google Charts in Laravel 7

Add Fillable Data in Student Model

We will have to specify the fields which will gonna insert through the form. Later, we’ll be creating the form inside the Reactjs CRUD app.

```
<?php

namespace App;
```

```
use Illuminate\Database\Eloquent\Model;

class Student extends Model
{
    protected $fillable = [
        "first_name", "last_name", "full_name", "email", "phone"
    ];
}
```

So, that’s it for the model and the migration. Now, let’s create the functionality for the CRUD operation.

Create Controller

Open the terminal or command prompt and hit the below command. It will create a controller file inside the **app/Http/Controllers**.

```
php artisan make:controller StudentController
```

Now, put the below snippet for the CRUD operations.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use App\Student;

class StudentController extends Controller
{
    private $status     =   200;
    // --------------- [ Save Student function ] -------------
    public function createStudent(Request $request) {

        // validate inputs
        $validator          =       Validator::make($request->all(),
            [
                "first_name"        =>      "required",
                "last_name"         =>      "required",
                "email"             =>      "required|email",
                "phone"             =>      "required|numeric"
            ]
        );

        // if validation fails
        if($validator->fails()) {
            return response()->json(["status" => "failed",
"validation_errors" => $validator->errors()]);
        }

        $student_id             =       $request->id;
         $studentArray           =       array(
            "first_name"            =>      $request->first_name,
            "last_name"             =>      $request->last_name,
            "full_name"             =>      $request->first_name . " " .
$request->last_name,
            "email"                 =>      $request->email,
            "phone"                 =>      $request->phone
        );

        if($student_id !="") {
            $student              =         Student::find($student_id);
            if(!is_null($student)){
                $updated_status     =       Student::where("id",
$student_id)->update($studentArray);
                if($updated_status == 1) {
                    return response()->json(["status" => $this->status,
"success" => true, "message" => "student detail updated successfully"]);
                }
                else {
                    return response()->json(["status" => "failed",
"message" => "Whoops! failed to update, try again."]);
                }
            }
        }

        else {
            $student        =       Student::create($studentArray);
            if(!is_null($student)) {
                return response()->json(["status" => $this->status,
"success" => true, "message" => "student record created successfully",
"data" => $student]);
            }
            else {
                return response()->json(["status" => "failed", "success" =>
false, "message" => "Whoops! failed to create."]);
            }
        }
    }

    // --------------- [ Students Listing ] -------------------
    public function studentsListing() {
        $students       =       Student::all();
        if(count($students) > 0) {
            return response()->json(["status" => $this->status, "success"
=> true, "count" => count($students), "data" => $students]);
        }
        else {
            return response()->json(["status" => "failed", "success" =>
false, "message" => "Whoops! no record found"]);
        }
    }

    // --------------- [ Student Detail ] ----------------
    public function studentDetail($id) {
        $student        =       Student::find($id);
        if(!is_null($student)) {
            return response()->json(["status" => $this->status, "success"
=> true, "data" => $student]);
        }
        else {
            return response()->json(["status" => "failed", "success" =>
false, "message" => "Whoops! no student found"]);
        }
    }
//---------------- [ Delete Student ] ----------
    public function studentDelete($id) {
        $student        =       Student::find($id);
        if(!is_null($student)) {
            $delete_status      =       Student::where("id", $id)-
>delete();
            if($delete_status == 1) {
                return response()->json(["status" => $this->status,
"success" => true, "message" => "student record deleted successfully"]);
            }
            else{
                return response()->json(["status" => "failed", "message" =>
"failed to delete, please try again"]);
            }
        }
        else {
            return response()->json(["status" => "failed", "message" =>
"Whoops! no student found with this id"]);
        }
    }
}
```

Laravel 7 Upload Multiple Images with Image Validation

Add Routes

So, for the **StudentController**, we will have to create the routes for the specified functions. Actually, we are creating the RESTful APIs hence, we will have to add the routes inside the **api.php** file

```
Route::post("create-student","StudentController@createStudent");

Route::get("students", "StudentController@studentsListing");

Route::get("student/{id}", "StudentController@studentDetail");

Route::delete("student/{id}", "StudentController@studentDelete");
```

So, the APIs endpoint has been defined and it is ready to implement. Now, let's move to the **Reactjs CRUD App**.

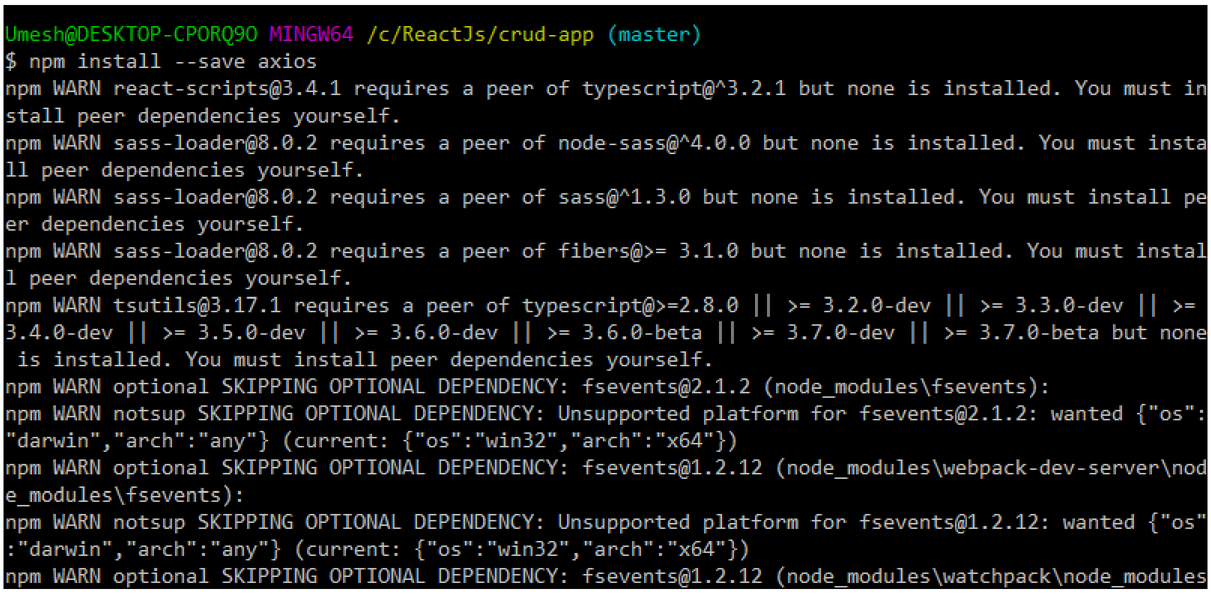
In React.js we will be using the **Axios library** for handling the HTTP requests.

Install Axios Library in React.js App

The npm provides a library named Axios. This library is used to handle the HTTP requests in the React.js. Basically, this can be used for handling any third-party API requests which will be using inside the React.js app.

Hence, install it by using the below command.

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



```
Umesh@DESKTOP-CPORQ90 MINGW64 /c/ReactJs/crud-app (master)
$ npm install --save axios
npm WARN react-scripts@3.4.1 requires a peer of typescript@^3.2.1 but none is installed. You must in
stall peer dependencies yourself.
npm WARN sass-loader@8.0.2 requires a peer of node-sass@^4.0.0 but none is installed. You must insta
ll peer dependencies yourself.
npm WARN sass-loader@8.0.2 requires a peer of sass@^1.3.0 but none is installed. You must install pe
er dependencies yourself.
npm WARN sass-loader@8.0.2 requires a peer of fibers@>= 3.1.0 but none is installed. You must instal
l peer dependencies yourself.
npm WARN tsutils@3.17.1 requires a peer of typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >=
3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta but none
 is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: wanted {"os":
"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.12 (node_modules\webpack-dev-server\nod
e_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.12: wanted {"os"
:"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.12 (node_modules\watchpack\node_modules
```

Install Axios Library

So, here Axios has been added inside the app. Now, our application is ready to handle the **RESTful APIs** requests. Now, go to the src folder you will see all the available files here. One of the most important things is the file structuring so first, we will structure our file.

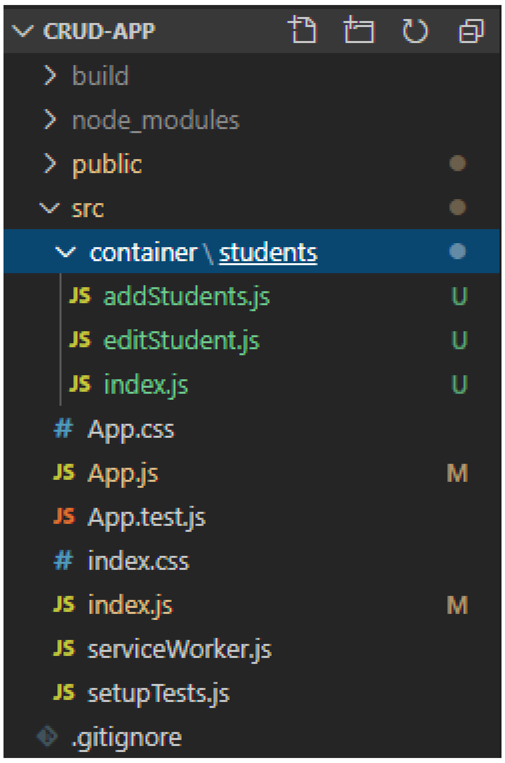
Create Component in Reactjs CRUD App

Navigate to the src folder in the react.js crud app. now, create a separate folder for the component that we'll create. Here, I am creating a folder named container. inside this create another folder named **students**.

Now, we will create the components for the student inside this **students** folder. Hence, According to our application, I am going to create three components as showing below.

1. **addStudents.js** (To add a new student)
2. **editStudent.js**,
3. **index.js**

After creating the above components, the folder structure will be looking like this.


```
CRUD-APP
  build
  node_modules
  public
  src
    container \ students
      addStudents.js      U
      editStudent.js      U
      index.js            U
    App.css
    App.js                M
    App.test.js
    index.css
    index.js              M
    serviceWorker.js
    setupTests.js
  .gitignore
```

Folder Structure of React.js CRUD app

Now, let's move to the implementation of **CRUD in React js**.

Import Student Component in App.js

Firstly, we will import the **Student** component in the **App.js** file.

```
/*------------ App.js -------*/
import React, { Component } from 'react';
import "./App.css";
import Student from './containers/students'

class App extends Component {
  render(){
    return (
      <div className="App">
        <Student/>
      </div>
    );
  }
  }

export default App;
```

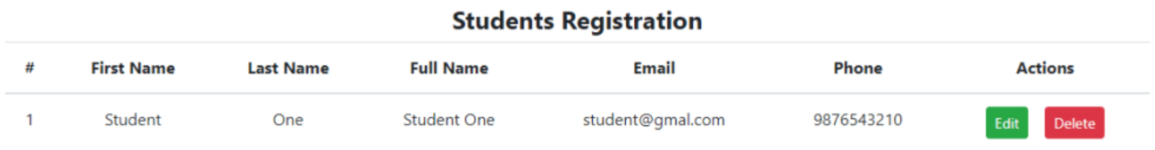
Create Layout Design

In **index.js** file, we'll create the design in which the students record will be listed out. Here, I will be using the table layout to display the students record.

So, we will also require to import the **Table** and **Button** component from the **Reactstrap.**

```
/*------- index.js ---------*/
import React, { Component } from "react";
import { Table, Button } from "reactstrap";

export default class Student extends Component {
  render() {
    return (
      <div className="App container mt-4">
           <h4 className="font-weight-bold">Students Registration</h4>
        <Table>
          <thead>
            <tr>
              <th>#</th>
              <th>First Name</th>
              <th>Last Name</th>
              <th>Full Name</th>
              <th>Email</th>
              <th>Phone</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td>1</td>
              <td>Student </td>
              <td>One</td>
              <td>Student One</td>
              <td>student@gmal.com</td>
              <td>9876543210</td>
              <td>
                <Button color="success" size="sm" className="mr-3">
                  Edit
                </Button>
                <Button color="danger" size="sm">
                  Delete
                </Button>
              </td>
            </tr>
          </tbody>
        </Table>
      </div>
    );
  }
}
```

Here, I have passed a static record of the student to see how the table will look. When you will run the application you will get the result like below.



Students Registration

#	First Name	Last Name	Full Name	Email	Phone	Actions
1	Student	One	Student One	student@gmal.com	9876543210	Edit Delete

Student Listing in Table

Now, we will be creating a design to add student data. I will be using the modal in which I will create a form and the input fields.

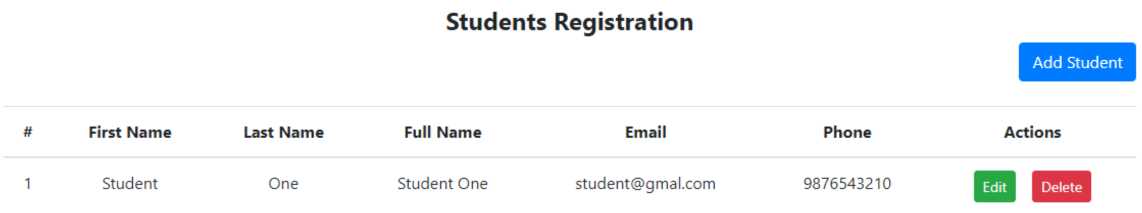
Create a Modal to Save Student

Open the **addStudent.js** file and paste the below code. This snippet will be generating a button to open the modal. Inside the modal, I have created the form fields to take the inputs for the student.

```
/* ----- addStudents.js -----*/
import React, { Component } from "react";
import {
  Button,
  Modal,
  ModalHeader,
  ModalBody,
  ModalFooter,
  FormGroup,
  Label,
  Input,
} from "reactstrap";

export default class addStudents extends Component {
  render() {
    return (
      <div>
        <Button className="float-right mb-4" color="primary">
          Add Student
        </Button>
        <Modal>
          <ModalHeader>Add new Student</ModalHeader>
          <ModalBody>
            <FormGroup>
              <Label for="first_name">First Name</Label>
              <Input id="first_name" name="first_name" />
            </FormGroup>
            <FormGroup>
              <Label for="last_name">Last Name</Label>
              <Input id="last_name" name="last_name" />
            </FormGroup>

            <FormGroup>
              <Label for="email">Email</Label>
              <Input id="email" name="email" />
            </FormGroup>
            <FormGroup>
              <Label for="phone">Phone</Label>
              <Input id="phone" name="phone" />
            </FormGroup>
          </ModalBody>
          <ModalFooter>
            <Button color="primary"> Add </Button>
            <Button color="secondary"> Cancel </Button>
          </ModalFooter>
        </Modal>
      </div>
    );
  }
}
```

Now, import the **addStudents.js** in the **index.js** file. So, that this component will be called in the index file where we have already created the table layout. When you will run the application, the index.js will be looking like this.

Here, a button has been added to open the modal to insert the student.


Students Registration

#	First Name	Last Name	Full Name	Email	Phone	Actions
1	Student	One	Student One	student@gmal.com	9876543210	Edit Delete

index.js file (design for students listing)

Here, we are just hardcoded the student data, Now, we will fetch students data from our API which have been created. So let's see how to make an API call in React.js.

HTTP Requests in Reactjs CRUD App

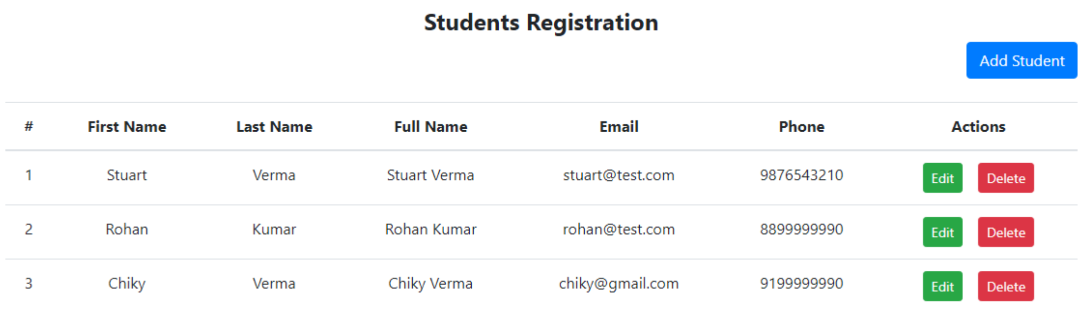
Before making any API request, we will have to import the **Axios** from **Axios library** in the **index.js** file

```
import axios from "axios";
```

Now, add the required states and functions here.

```
/*-------- index.js ----------*/
import React, { Component } from "react";
import { Table, Button } from "reactstrap";
import axios from "axios";
import AddStudent from './addStudents';

export default class Student extends Component {
    constructor(props){
        super(props);
        this.state = {
            students: [],
        }
    }

    componentDidMount() {
        this.getStudents();
      }
    getStudents() {
    axios.get("http://localhost:8000/api/students").then((response) => {
        if (response.status === 200) {
        this.setState({
            students: response.data.data ? response.data.data : [],
        });
        }
        if (
        response.data.status === "failed" &&
        response.data.success === false
        ) {
        this.setState({
            noDataFound: response.data.message,
        });
        }
    });
    }
  render() {
    const { noDataFound, students} = this.state;
      let studentsDetails = [];
      if (students.length) {
        studentsDetails = students.map((student) => {
          return (
            <tr key={student.id}>
              <td>{student.id}</td>
              <td>{student.first_name}</td>
              <td>{student.last_name}</td>
              <td>{student.full_name}</td>
              <td>{student.email}</td>
              <td>{student.phone}</td>
              <td>
                <Button
                  color="success"
                  className="mr-3"
                  size="sm"
                >
                  Edit
                </Button>
                <Button
                  color="danger"
                  size="sm"
                >
                  Delete
                </Button>
              </td>
            </tr>
          );
        });
      }

    return (
      <div className="App container mt-4">
           <h4 className="font-weight-bold">Students Registration</h4>
           <AddStudent/>
        <Table>
          <thead>
            <tr>
              <th>#</th>
              <th>First Name</th>
              <th>Last Name</th>
              <th>Full Name</th>
              <th>Email</th>
              <th>Phone</th>
              <th>Actions</th>
            </tr>
          </thead>
          {students.length === 0 ? (
            <tbody>
              <h3>{noDataFound}</h3>
            </tbody>
          ) : (
            <tbody>{studentsDetails}</tbody>
          )}
        </Table>
      </div>
    );
  }
}
```

Here, we are mapping the fields of the students and making an **HTTP** request to get the data from the **API**. The request type is **GET** because we are retrieving the students data.



Students Registration

Add Student

#	First Name	Last Name	Full Name	Email	Phone	Actions
1	Stuart	Verma	Stuart Verma	stuart@test.com	9876543210	Edit Delete
2	Rohan	Kumar	Rohan Kumar	rohan@test.com	8899999990	Edit Delete
3	Chiky	Verma	Chiky Verma	chiky@gmail.com	9199999990	Edit Delete

Here the students data is displaying which is coming from the **API**.

Next, we will make our Add Student model workable. So that we can insert the data from the frontend (Reactjs CRUD app).

Add Functionality to the Add Student Modal

At this level, the modal functionality is not completed to take any inputs. So, let's do that. Add the below code in the **index.js** file.

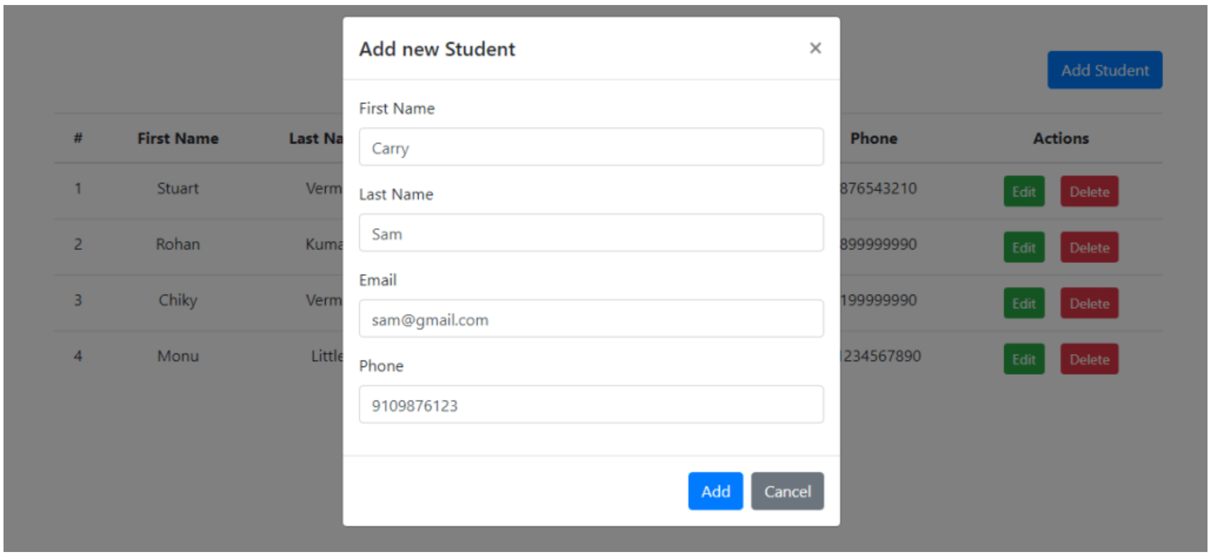
```
/*------------ index.js ---------------*/
import React, { Component } from "react";
import { Table, Button } from "reactstrap";
import axios from "axios";
import AddStudents from './addStudents';

export default class Student extends Component {
    constructor(props){
        super(props);
        this.state = {
            students: [],
            newStudentData: {
              first_name: "",
              last_name: "",
              email: "",
              phone: "",
            },
            isLoading: false,
            status: "",
            newStudentModal: false,
        }
    }

    componentDidMount() {
        this.getStudents();
      }
    getStudents() {
    axios.get("http://localhost:8000/api/students").then((response) => {
        if (response.status === 200) {
        this.setState({
            students: response.data.data ? response.data.data : [],
        });
        }
        if (
        response.data.status === "failed" &&
        response.data.success === false
        ) {
        this.setState({
            noDataFound: response.data.message,
        });
        }
    });
    }

    toggleNewStudentModal = () => {
        this.setState({
          newStudentModal: !this.state.newStudentModal,
        });
      };
    onChangeAddStudentHandler = (e) => {
        let { newStudentData } = this.state;
        newStudentData[e.target.name] = e.target.value;
        this.setState({ newStudentData });
    };
    addStudent = () => {
        axios
          .post(
            "http://localhost:8000/api/create-student",
            this.state.newStudentData
          )
          .then((response) => {
            const { students } = this.state;
            const newStudents = [...students];
            newStudents.push(response.data);
            this.setState(
              {
                students: newStudents,
                newStudentModal: false,
                newStudentData: {
                  first_name: "",
                  last_name: "",
                  email: "",
                  phone: "",
                },
              },
              () => this.getStudents()
            );
          });
      };

  render() {
    const { newStudentData, noDataFound, students} = this.state;
```

```
      let studentsDetails = [];
      if (students.length) {
        studentsDetails = students.map((student) => {
          return (
            <tr key={student.id}>
              <td>{student.id}</td>
              <td>{student.first_name}</td>
              <td>{student.last_name}</td>
              <td>{student.full_name}</td>
              <td>{student.email}</td>
              <td>{student.phone}</td>
              <td>
                <Button
                  color="success"
                  className="mr-3"
                  size="sm"
                >
                  Edit
                </Button>
                <Button
                  color="danger"
                  size="sm"
                >
                  Delete
                </Button>
              </td>
            </tr>
          );
        });
      }

      if (this.state.isLoading) {
        return <div className="spinner-border text-center" role="status">
<span className="sr-only">Loading...</span>
      </div>
      }
    return (
      <div className="App container mt-4">
           <h4 className="font-weight-bold">Students Registration</h4>
           <AddStudents
                toggleNewStudentModal={this.toggleNewStudentModal}
                newStudentModal={this.state.newStudentModal}
                onChangeAddStudentHandler={this.onChangeAddStudentHandler}
                addStudent={this.addStudent}
                newStudentData={newStudentData}
          />
        <Table>
          <thead>
            <tr>
              <th>#</th>
              <th>First Name</th>
              <th>Last Name</th>
              <th>Full Name</th>
              <th>Email</th>
              <th>Phone</th>
              <th>Actions</th>
            </tr>
          </thead>
          {students.length === 0 ? (
            <tbody>
              <h3>{noDataFound}</h3>
            </tbody>
          ) : (
            <tbody>{studentsDetails}</tbody>
          )}
        </Table>
      </div>
    );
  }
}
```

In **<Addstudents/>** we will pass the data as a props.

```
 <AddStudents
       toggleNewStudentModal={this.toggleNewStudentModal}
       newStudentModal={this.state.newStudentModal}
       onChangeAddStudentHandler={this.onChangeAddStudentHandler}
       addStudent={this.addStudent}
       newStudentData={newStudentData}
 />
/*----- addStudent.js ---------*/
import React, { Component } from "react";
import {
    Button,
    Modal,
    ModalHeader,
    ModalBody,
    ModalFooter,
    FormGroup,
    Label,
    Input,
  } from "reactstrap";

export default class addStudents extends Component {
  render() {
    return (
      <div>
        <Button
          className="float-right mb-4"
          color="primary"
          onClick={this.props.toggleNewStudentModal}
        >
          Add Student
        </Button>
        <Modal
          isOpen={this.props.newStudentModal}
          toggle={this.props.toggleNewStudentModal}
        >
          <ModalHeader toggle={this.props.toggleNewStudentModal}>
            Add new Student
          </ModalHeader>
          <ModalBody>
            <FormGroup>
              <Label for="first_name">First Name</Label>
              <Input
                id="first_name"
                name="first_name"
                value={this.props.newStudentData.first_name}
                onChange={this.props.onChangeAddStudentHandler}
              />
            </FormGroup>
            <FormGroup>
              <Label for="last_name">Last Name</Label>
              <Input
                id="last_name"
                name="last_name"
                value={this.props.newStudentData.last_name}
                onChange={this.props.onChangeAddStudentHandler}
              />
            </FormGroup>

            <FormGroup>
              <Label for="email">Email</Label>
              <Input
                id="email"
                name="email"
                value={this.props.newStudentData.email}
                onChange={this.props.onChangeAddStudentHandler}
              />
            </FormGroup>
            <FormGroup>
              <Label for="phone">Phone</Label>
              <Input
                id="phone"
                name="phone"
                value={this.props.newStudentData.phone}
                onChange={this.props.onChangeAddStudentHandler}
              />
            </FormGroup>
          </ModalBody>
          <ModalFooter>
            <Button color="primary" onClick={() =>
this.props.addStudent()}>
              Add
            </Button>{" "}
            <Button color="secondary"
onClick={this.props.toggleNewStudentModal}>
              Cancel
            </Button>
          </ModalFooter>
        </Modal>
      </div>
    );
  }
}
```

When you will click on the **Add student** button, a Model will be opened to add the student data.


Add new Student

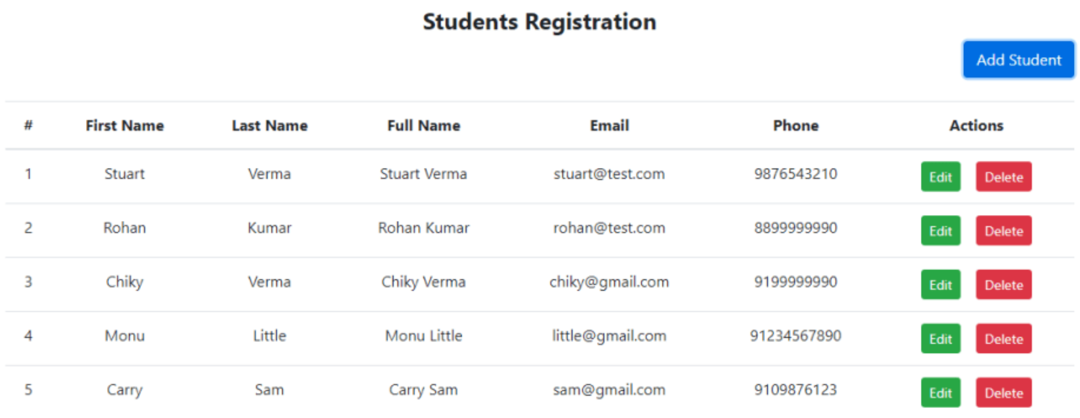
First Name: Carry

Last Name: Sam

Email: sam@gmail.com

Phone: 9109876123

After filling the student data just click on the **Add** button. The API will trigger after taking form data and the student data will be saved into the database.


Students Registration

#	First Name	Last Name	Full Name	Email	Phone	Actions
1	Stuart	Verma	Stuart Verma	stuart@test.com	9876543210	Edit Delete
2	Rohan	Kumar	Rohan Kumar	rohan@test.com	8899999990	Edit Delete
3	Chiky	Verma	Chiky Verma	chiky@gmail.com	9199999990	Edit Delete
4	Monu	Little	Monu Little	little@gmail.com	91234567890	Edit Delete
5	Carry	Sam	Carry Sam	sam@gmail.com	9109876123	Edit Delete

Student data Listing

Now, we will add the functionality for **Edit**, **Update** and **Delete**..

```
/* ------------- index.js -------------- */
import React, { Component } from "react";
import { Table, Button } from "reactstrap";
import axios from "axios";
import AddStudents from './addStudents';
import EditStudent from './editStudent';

export default class Student extends Component {
    constructor(props) {
        super(props);
        this.state = {
          students: [],
          newStudentData: {
            first_name: "",
            last_name: "",
            email: "",
            phone: "",
          },
          isLoading: false,
          status: "",
          newStudentModal: false,
          editStudentData: {
            id: "",
            first_name: "",
            last_name: "",
            email: "",
            phone: "",
          },
          editStudentModal: false,
          noDataFound: "",
        };
      }

    componentDidMount() {
        this.getStudents();
      }
    getStudents() {
    axios.get("http://localhost:8000/api/students").then((response) => {
        if (response.status === 200) {
        this.setState({
            students: response.data.data ? response.data.data : [],
        });
        }
        if (
        response.data.status === "failed" &&
        response.data.success === false
        ) {
        this.setState({
            noDataFound: response.data.message,
        });
        }
    });
    }

    toggleNewStudentModal = () => {
        this.setState({
          newStudentModal: !this.state.newStudentModal,
        });
      };
    onChangeAddStudentHandler = (e) => {
        let { newStudentData } = this.state;
        newStudentData[e.target.name] = e.target.value;
        this.setState({ newStudentData });
    };
    addStudent = () => {
        axios
          .post(
            "http://localhost:8000/api/create-student",
            this.state.newStudentData
          )
          .then((response) => {
            const { students } = this.state;
            const newStudents = [...students];
            newStudents.push(response.data);
            this.setState(
              {
                students: newStudents,
                newStudentModal: false,
                newStudentData: {
```

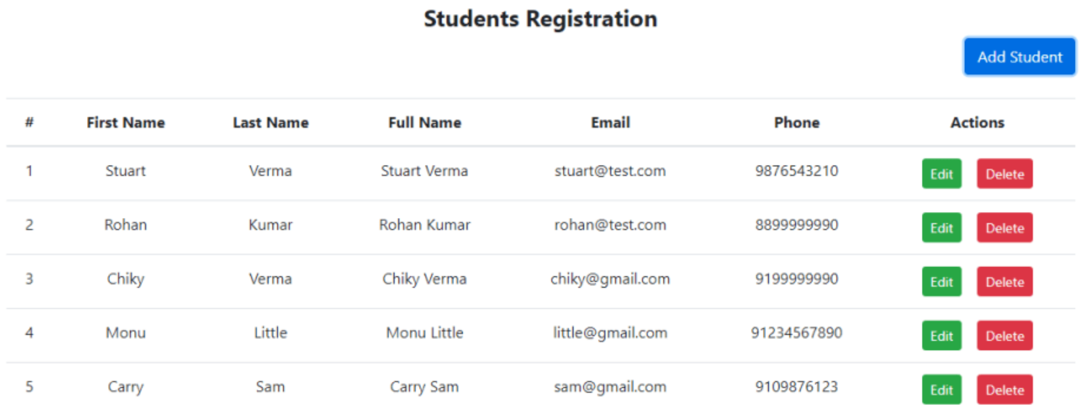
```
            first_name: "",
            last_name: "",
            email: "",
            phone: "",
          },
        },
        () => this.getStudents()
      );
    });
};

toggleEditStudentModal = () => {
  this.setState({
    editStudentModal: !this.state.editStudentModal,
  });
};

onChangeEditStudentHanler = (e) => {
  let { editStudentData } = this.state;
  editStudentData[e.target.name] = e.target.value;
  this.setState({ editStudentData });
};

editStudent = (id, first_name, last_name, email, phone) => {
  this.setState({
    editStudentData: { id, first_name, last_name, email, phone },
    editStudentModal: !this.state.editStudentModal,
  });
};

updateStudent = () => {
  let {
    id,
    first_name,
    last_name,
    email,
    phone,
  } = this.state.editStudentData;
  this.setState({
    isLoading: true,
  });
  axios
    .post("http://localhost:8000/api/create-student", {
      first_name,
      last_name,
      email,
      phone,
      id,
    })
    .then((response) => {
      this.getStudents();
      this.setState({
        editStudentModal: false,
        editStudentData: { first_name, last_name, email, phone },
        isLoading:false,
      });
    })
    .catch((error) => {
      this.setState({isLoading:false})
      console.log(error.response);
    });
```

```
      };

      deletStudent = (id) => {
        this.setState({
          isLoading: true,
        });
        axios
          .delete("http://localhost:8000/api/student/" + id)
          .then((response) => {
            this.setState({
              isLoading: false,
            });
            this.getStudents();
          })
          .catch((error) => {
            this.setState({
              isLoading: false,
            });
          });
      };

  render() {
    const { newStudentData,editStudentData,noDataFound,students} =
this.state;
      let studentsDetails = [];
      if (students.length) {
        studentsDetails = students.map((student) => {
          return (
            <tr key={student.id}>
              <td>{student.id}</td>
              <td>{student.first_name}</td>
              <td>{student.last_name}</td>
              <td>{student.full_name}</td>
              <td>{student.email}</td>
              <td>{student.phone}</td>
              <td>
                <Button
                  color="success"
                  className="mr-3"
                  size="sm"
                  onClick={() =>
                    this.editStudent(
                      student.id,
                      student.first_name,
                      student.last_name,
                      student.email,
                      student.phone
                    )
                  }
                >
                  Edit
                </Button>
                <Button
                  color="danger"
                  size="sm"
                  onClick={() => this.deletStudent(student.id)}
                >
                  Delete
                </Button>
              </td>
            </tr>
          );
        });
      }

      if (this.state.isLoading) {
        return <div className="spinner-border text-center" role="status">
<span className="sr-only">Loading...</span>
      </div>
      }
    return (
      <div className="App container mt-4">
           <h4 className="font-weight-bold">Students Registration</h4>
            {/* Model for Add Studnet Record */}
           <AddStudents
                toggleNewStudentModal={this.toggleNewStudentModal}
                newStudentModal={this.state.newStudentModal}
                onChangeAddStudentHandler={this.onChangeAddStudentHandler}
                addStudent={this.addStudent}
                newStudentData={newStudentData}
          />
         {/* Model for Edit Studnet Record */}
            <EditStudent
            toggleEditStudentModal={this.toggleEditStudentModal}
            editStudentModal={this.state.editStudentModal}
            onChangeEditStudentHanler={this.onChangeEditStudentHanler}
            editStudent={this.editStudent}
            editStudentData={editStudentData}
            updateStudent={this.updateStudent}
            />
        <Table>
          <thead>
            <tr>
              <th>#</th>
              <th>First Name</th>
              <th>Last Name</th>
              <th>Full Name</th>
              <th>Email</th>
              <th>Phone</th>
              <th>Actions</th>
            </tr>
          </thead>
          {students.length === 0 ? (
            <tbody>
              <h3>{noDataFound}</h3>
            </tbody>
          ) : (
            <tbody>{studentsDetails}</tbody>
          )}
        </Table>
      </div>
    );
  }
}
```

Add below code in editStudent.js file to **Edit** and **Delete** student data.

```
/* ------- editStudent.js ----------- */

import React, { Component } from "react";
import {
  Button,
  Modal,
  ModalHeader,
  ModalBody,
  ModalFooter,
  FormGroup,
  Label,
  Input,
} from "reactstrap";

export default class editStudent extends Component {
  render() {
    return (
      <div>
        <Modal
          isOpen={this.props.editStudentModal}
          toggle={this.props.toggleEditStudentModal}
        >
          <ModalHeader toggle={this.props.toggleEditStudentModal}>
            Update Student
          </ModalHeader>
          <ModalBody>
            <FormGroup>
              <Label for="first_name">First Name</Label>
              <Input
                id="first_name"
                name="first_name"
                value={this.props.editStudentData.first_name}
                onChange={this.props.onChangeEditStudentHanler}
              />
            </FormGroup>
            <FormGroup>
              <Label for="last_name">Last Name</Label>
              <Input
                id="last_name"
                name="last_name"
                value={this.props.editStudentData.last_name}
                onChange={this.props.onChangeEditStudentHanler}
              />
            </FormGroup>

            <FormGroup>
              <Label for="email">Email</Label>
              <Input
                id="email"
                name="email"
                value={this.props.editStudentData.email}
                onChange={this.props.onChangeEditStudentHanler}
              />
            </FormGroup>
            <FormGroup>
              <Label for="phone">Phone</Label>
              <Input
                id="phone"
                name="phone"
                value={this.props.editStudentData.phone}
                onChange={this.props.onChangeEditStudentHanler}
              />
            </FormGroup>
          </ModalBody>
          <ModalFooter>
```

```
            <Button
              color="primary"
              onClick={this.props.updateStudent}
            >
              Update
            </Button>
            <Button
              color="secondary"
              onClick={this.props.toggleEditStudentModal}
            >
              Cancel
            </Button>
          </ModalFooter>
        </Modal>
      </div>
    );
  }
}
```



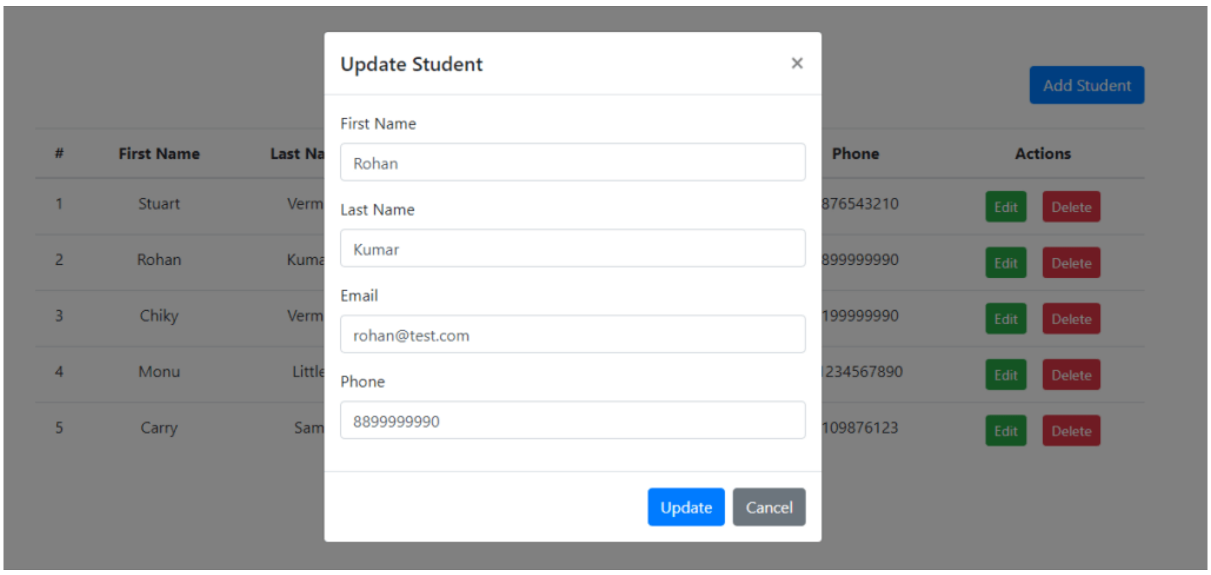
Students Registration

Add Student

#	First Name	Last Name	Full Name	Email	Phone	Actions
1	Stuart	Verma	Stuart Verma	stuart@test.com	9876543210	Edit Delete
2	Rohan	Kumar	Rohan Kumar	rohan@test.com	8899999990	Edit Delete
3	Chiky	Verma	Chiky Verma	chiky@gmail.com	9199999990	Edit Delete
4	Monu	Little	Monu Little	little@gmail.com	91234567890	Edit Delete
5	Carry	Sam	Carry Sam	sam@gmail.com	9109876123	Edit Delete

Student data Listing

Click on **Edit** button update Student model will be open .



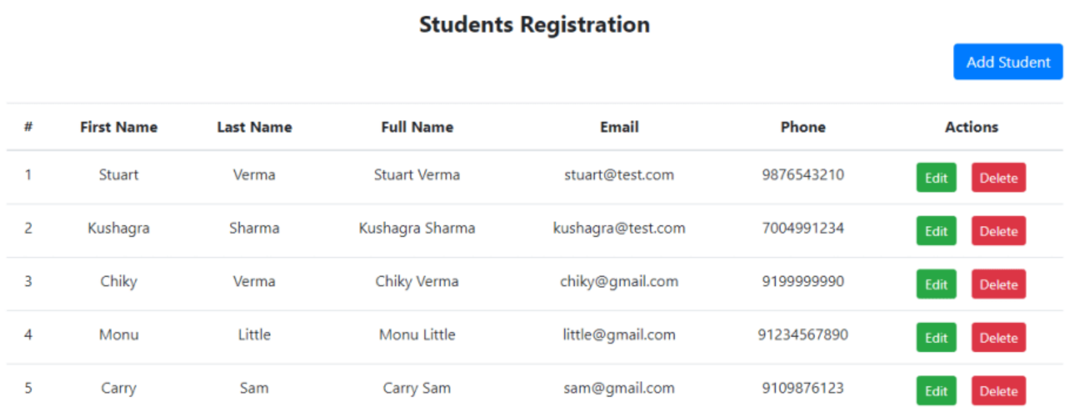
Update Student ×

First Name	Rohan
Last Name	Kumar
Email	rohan@test.com
Phone	8899999990

Update Cancel

Edit Student data

Edit the student data and **update** it. Data will be updated in the database.


Students Registration

#	First Name	Last Name	Full Name	Email	Phone	Actions
1	Stuart	Verma	Stuart Verma	stuart@test.com	9876543210	Edit Delete
2	Kushagra	Sharma	Kushagra Sharma	kushagra@test.com	7004991234	Edit Delete
3	Chiky	Verma	Chiky Verma	chiky@gmail.com	9199999990	Edit Delete
4	Monu	Little	Monu Little	little@gmail.com	91234567890	Edit Delete
5	Carry	Sam	Carry Sam	sam@gmail.com	9109876123	Edit Delete

Student Data Listing

When you will click on the **Delete** button student data will be deleted from the database