React CRUD Example | MERN Stack Tutorial

React CRUD Example | MERN Stack Tutorial is today's leading topic. This article's goal is to explain the by making the **CRUD** application. Most applications are **CRUD** applications. If you don't know what **CRUD** is, it's short for **Create, Read, Update and Delete.** This application is also called a **MERN Stack** application because we are using **MongoDB**, **Express**, **React**, and **Node.js** tech stack. This article will show you the steps to create an app where the user can create new business, fetch that business, edit business and delete business using React.js. Let us understand the basics of **React.js.**

What is React?

React.js is UI Library and not a complete Framework like **Angular**. React is the JavaScript library for building user interfaces. It is maintained by Facebook, Instagram and a community of individual developers and corporations. You can find its official documentation here.

Why use React?

React.js allows you to express how your app should look at any given point in time. React will automatically manage all UI updates when your underlying data changes. When the data changes, React conceptually hits the "refresh" button and knows to only update the changed parts. The main reason to use React for your next project is following.

- Fast Learning Curve.
- Reusable Components.
- Fast render with Virtual DOM.
- Clean Abstraction.
- Redux: A State Management Library For Complex UIs.
- Great Developer Tools.
- React Native: You can build a cross-platform native mobile application for **Android** or **iOS**.

Who uses React?

Aside from **Facebook** and **Instagram**, many well-known companies use React including **Dropbox**,**PayPal**, **Netflix**, **Airbnb** and many more.

MERN Stack Tutorial

We will use the following Technologies with its version.

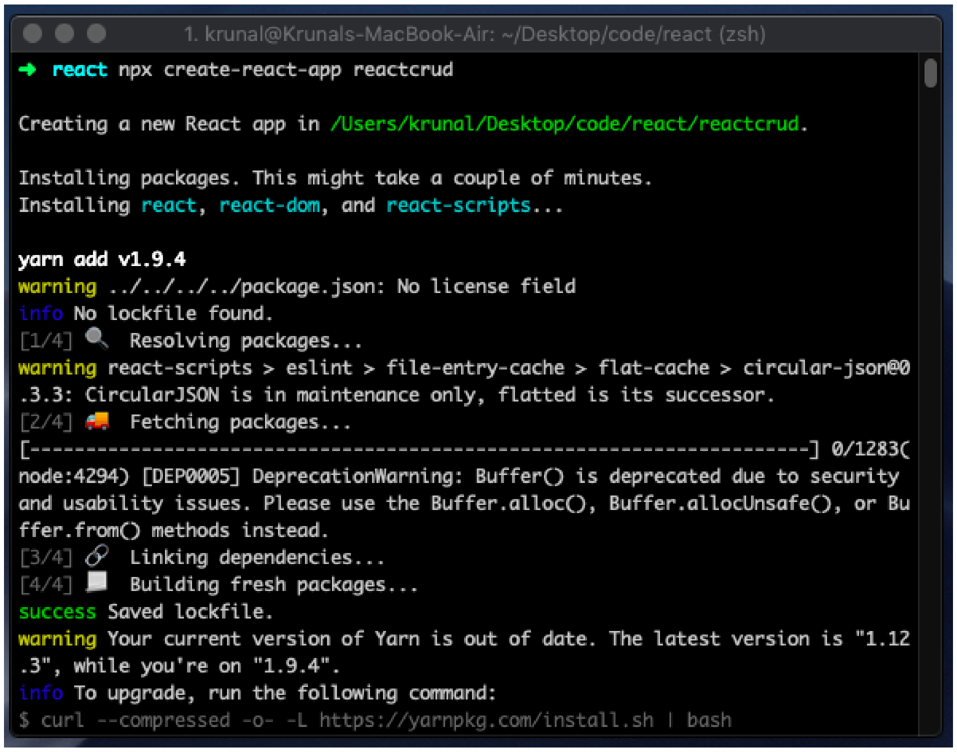
- NodeJS
- NPM
- MongoDB shell
- MongoDB
- React.js

React CRUD Example

Create a new React app using the following command.

#1: Install React Application

```
npx create-react-app reactcrud
cd reactcrud
npm start
```

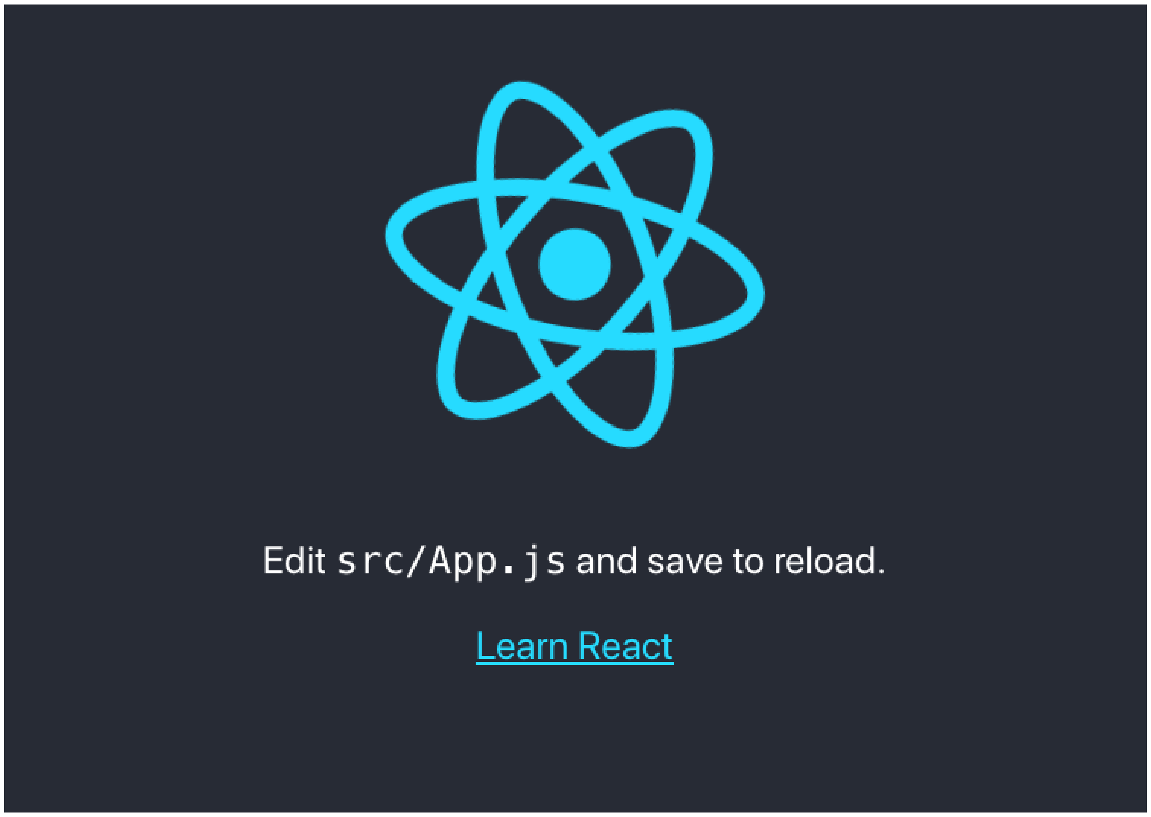


```
1. krunal@Krunals-MacBook-Air: ~/Desktop/code/react (zsh)
➜ react npx create-react-app reactcrud

Creating a new React app in /Users/krunal/Desktop/code/react/reactcrud.

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

yarn add v1.9.4
warning ../../../../package.json: No license field
info No lockfile found.
[1/4] 🔍  Resolving packages...
warning react-scripts > eslint > file-entry-cache > flat-cache > circular-json@0
.3.3: CircularJSON is in maintenance only, flatted is its successor.
[2/4] 🚚  Fetching packages...
[------------------------------------------------------------------] 0/1283(
node:4294) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security
and usability issues. Please use the Buffer.alloc(), Buffer.allocUnsafe(), or Bu
ffer.from() methods instead.
[3/4] 🔗  Linking dependencies...
[4/4] 📃  Building fresh packages...
success Saved lockfile.
warning Your current version of Yarn is out of date. The latest version is "1.12
.3", while you're on "1.9.4".
info To upgrade, run the following command:
$ curl --compressed -o- -L https://yarnpkg.com/install.sh | bash
```

If you are facing any issue on compiling then, please create a **.env** file on the root and add the following line of code.

```
SKIP_PREFLIGHT_CHECK=true
```

Go to this URL: **http://localhost:3000/**


Edit src/App.js and save to reload.

Learn React

Now, install the Bootstrap 4 Framework using the following command.

```
yarn add bootstrap

# or

npm install bootstrap --save
```

Import the **Bootstrap CSS Framework** inside our project.
Modify the code inside the **src >> App.js** file.

```
// App.js

import React, { Component } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';

class App extends Component {
  render() {
    return (
      <div className="container">
          <h2>React CRUD Tutorial</h2>
        </div>
      );
    }
  }

 export default App;
```

Save the file and go to the browser and you can see that we have successfully integrated **bootstrap 4** in our react application.

#2: Configure React routing

Type the following command to install the **react- router-dom module**. If you want to find more information about **the react-router-dom** module, then you can follow this documentation.

```
yarn add react-router-dom

# or

npm install react-router-dom --save
```

Go to the **index.js** file and Wrap the **BrowserRouter** object around the **App.js** component.

```
// index.js

import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';

import App from './App';
import * as serviceWorker from './serviceWorker';

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

serviceWorker.unregister();
```

Now, create three components.

Inside the **src** folder, create one directory called **components** and inside that folder, make three components.

- create.component.js
- edit.component.js
- index.component.js

```
// create.component.js

import React, { Component } from 'react';

export default class Create extends Component {
    render() {
        return (
            <div>
                <p>Welcome to Create Component!!</p>
            </div>
        )
    }
}
```

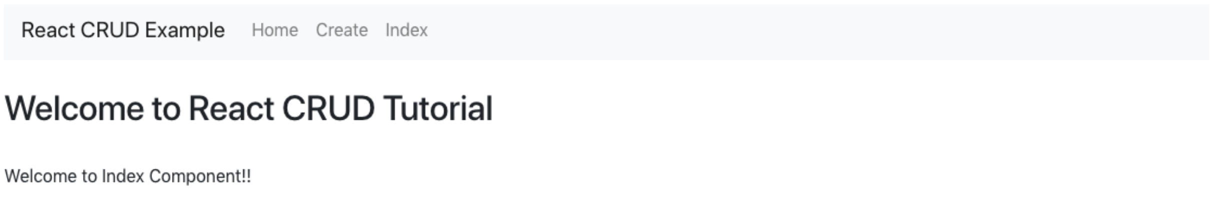
```
// edit.component.js

import React, { Component } from 'react';

export default class Edit extends Component {
    render() {
        return (
            <div>
                <p>Welcome to Edit Component!!</p>
             </div>
         )
     }
 }
```

```
// index.component.js

import React, { Component } from 'react';

export default class Index extends Component {
    render() {
        return (
            <div>
                <p>Welcome to Index Component!!</p>
             </div>
         )
     }
 }
```

Now, add the navigation bar in our **React CRUD example**. Write a following code inside the **App.js** file.

```
// App.js

import React, { Component } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom';

import Create from './components/create.component';
import Edit from './components/edit.component';
import Index from './components/index.component';

class App extends Component {
  render() {
    return (
      <Router>
        <div className="container">
          <nav className="navbar navbar-expand-lg navbar-light bg-light">
            <Link to={'/'} className="navbar-brand">React CRUD Example</Link>
            <div className="collapse navbar-collapse" id="navbarSupportedContent">
              <ul className="navbar-nav mr-auto">
              <li className="nav-item">
                  <Link to={'/'} className="nav-link">Home</Link>
                </li>
                <li className="nav-item">
                  <Link to={'/create'} className="nav-link">Create</Link>
                </li>
                <li className="nav-item">
                  <Link to={'/index'} className="nav-link">Index</Link>
                </li>
              </ul>
            </div>
          </nav> <br/>
          <h2>Welcome to React CRUD Tutorial</h2> <br/>
          <Switch>
              <Route exact path='/create' component={ Create } />
              <Route path='/edit/:id' component={ Edit } />
              <Route path='/index' component={ Index } />
          </Switch>
        </div>
      </Router>
    );
  }
}

export default App;
```

Save the file and go to the browser.



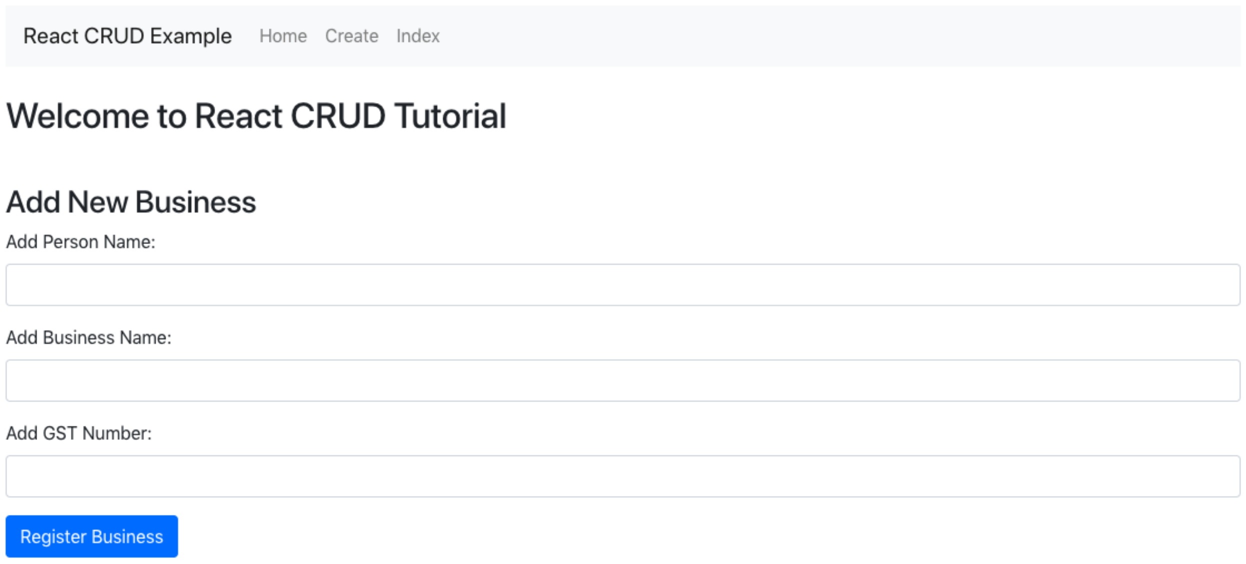
Welcome to React CRUD Tutorial

Welcome to Index Component!!

#3: Create the bootstrap form

Write the code to generate the bootstrap form inside the **create.component.js** file.

```
// create.component.js

import React, { Component } from 'react';

export default class Create extends Component {
    render() {
        return (
            <div style={{marginTop: 10}}>
                <h3>Add New Business</h3>
                 <form>
                     <div className="form-group">
                         <label>Add Person Name:  </label>
                         <input type="text" className="form-control"/>
                     </div>
                     <div className="form-group">
                         <label>Add Business Name: </label>
                         <input type="text" className="form-control"/>
                     </div>
                     <div className="form-group">
                         <label>Add GST Number: </label>
                         <input type="text" className="form-control"/>
                     </div>
                     <div className="form-group">
                         <input type="submit" value="Register Business"
className="btn btn-primary"/>
                     </div>
                 </form>
             </div>
         )
     }
 }
```



Welcome to React CRUD Tutorial

Add New Business

Add Person Name:

Add Business Name:

Add GST Number:

#4: Submit the Form

Okay, now we have three fields.

- person name
- business name
- gst number

So we need to create four functions that can track the values of the textbox and set that state according to it. Also, the fourth function will send the **POST** request to the node express server.

Now, first, we will define the constructor and set the initial state and then also bind this to the different events inside the constructor.

Then define the different functions with each input text values. So when the user types inside the textbox, we set the state according to it.

So, let say, the user is typing the person name inside the textbox, we are changing the state value of person name. Finally, same for all of the inputs and when we submit the form, we get the values from the state and send to the POST request.

```
// App.js

import React, { Component } from 'react';

export default class Create extends Component {
  constructor(props) {
      super(props);
      this.onChangePersonName = this.onChangePersonName.bind(this);
      this.onChangeBusinessName = this.onChangeBusinessName.bind(this);
        this.onChangeGstNumber = this.onChangeGstNumber.bind(this);
        this.onSubmit = this.onSubmit.bind(this);

        this.state = {
            person_name: '',
            business_name: '',
            business_gst_number:''
        }
    }
    onChangePersonName(e) {
      this.setState({
        person_name: e.target.value
      });
    }
    onChangeBusinessName(e) {
      this.setState({
        business_name: e.target.value
      })
    }
    onChangeGstNumber(e) {
      this.setState({
        business_gst_number: e.target.value
      })
    }

    onSubmit(e) {
      e.preventDefault();
      console.log(`The values are ${this.state.person_name},
${this.state.business_name}, and ${this.state.business_gst_number}`)
      this.setState({
        person_name: '',
        business_name: '',
        business_gst_number: ''
      })
    }

    render() {
        return (
            <div style={{ marginTop: 10 }}>
                <h3>Add New Business</h3>
                <form onSubmit={this.onSubmit}>
                    <div className="form-group">
                        <label>Person Name:  </label>
                        <input
                          type="text"
                          className="form-control"
                          value={this.state.person_name}
                          onChange={this.onChangePersonName}
                          />
                    </div>
                    <div className="form-group">
                        <label>Business Name: </label>
                        <input type="text"
                          className="form-control"
                          value={this.state.business_name}
                          onChange={this.onChangeBusinessName}
                          />
                    </div>
                    <div className="form-group">
                        <label>GST Number: </label>
                        <input type="text"
                          className="form-control"
                          value={this.state.business_gst_number}
                          onChange={this.onChangeGstNumber}
                          />
                    </div>
                    <div className="form-group">
                        <input type="submit" value="Register Business" className="btn btn-primary"/>
                    </div>
                </form>
            </div>
        )
    }
  }
```

#5: Create a backend on Node.js

Inside our **reactcrud** project root, create one folder called **api** and go inside that folder and initialize the **package.json** file.

```
npm init -y
```

Now, install the following **node.js** dependencies.

```
yarn add express body-parser cors mongoose

# or

npm install express body-parser cors mongoose --save
```

Also, install the **nodemon** as a development dependency. So that we do not need to restart every time, we change our server code.

```
npm install nodemon --save-dev
```

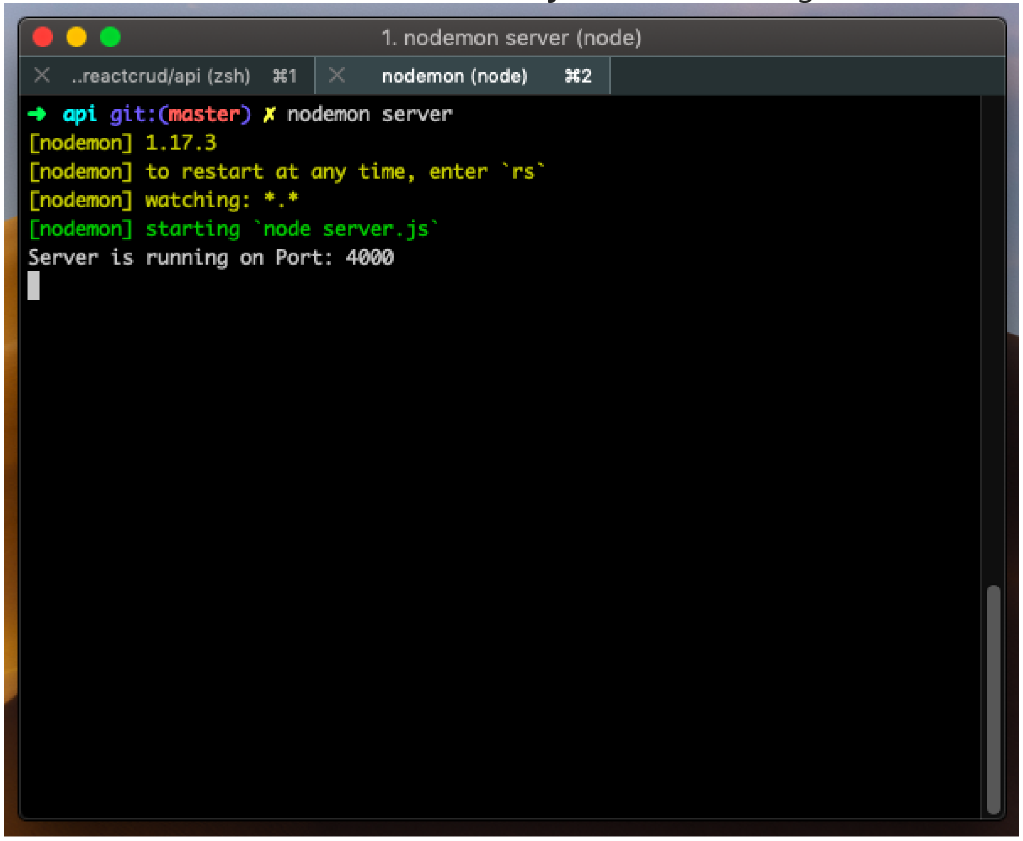
Now, inside the **api** folder, create one file called the **server.js** and add the following code inside it.

```
// server.js

const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const PORT = 4000;
const cors = require('cors');

app.use(cors());
app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());

app.listen(PORT, function(){
  console.log('Server is running on Port:',PORT);
});
```

Save the file and open a new tab terminal inside the **api** folder and hit the following command to spin up the node.js server.

```
nodemon server
```

You can see inside the terminal that, our **node.js** server is running.



```
➜ api git:(master) ✗ nodemon server
[nodemon] 1.17.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`
Server is running on Port: 4000
```

#6: Setup a MongoDB database

If you are a beginner in **MongoDB** database, then please check out by below tutorial.
Related Post: NoSQL MongoDB Tutorial
I have already installed the **MongoDB .** So I am starting the **MongoDB** server by the following command.

```
mongod
```

Inside the **api** folder, create one file called the **DB.js** and add the following line of code.

```
// DB.js

module.exports = {
    DB: 'mongodb://localhost:27017/reactcrud'
}
```

In my local **MongoDB** database, the username and password are empty, but in the production, you need to create one admin user and assign the database to that user.
Now, import this **DB.js** file into the **server.js** file.

```
// server.js

const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const PORT = 4000;
const cors = require('cors');
const mongoose = require('mongoose');
const config = require('./DB.js');

mongoose.Promise = global.Promise;
mongoose.connect(config.DB, { useNewUrlParser: true }).then(
  () => {console.log('Database is connected') },
  err => { console.log('Can not connect to the database'+ err)}
);

app.use(cors());
app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());

app.listen(PORT, function(){
  console.log('Server is running on Port:',PORT);
});
```

Save the file, and you can see inside the terminal that our **node.js** application is connected to a**mongodb** database.

#7: Create a Mongoose Schema

Next step is that we need to create a schema for the mongodb database. For that, create a file inside the api project root called **business.model.js** and add the following code.

```
// business.model.js

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

// Define collection and schema for Business
let Business = new Schema({
  person_name: {
    type: String
   },
   business_name: {
     type: String
   },
   business_gst_number: {
     type: Number
   }
 },{
     collection: 'business'
 });

 module.exports = mongoose.model('Business', Business);
```

We have taken three fields called **person_name, business_name,** and **business_gst_number** with its datatypes.

#8: Define the route for Node.js Express application

Write the CRUD code inside the **business.route.js** file.

```
// business.route.js

const express = require('express');
const businessRoutes = express.Router();

// Require Business model in our routes module
let Business = require('./business.model');

// Defined store route
businessRoutes.route('/add').post(function (req, res) {
   let business = new Business(req.body);
   business.save()
     .then(business => {
       res.status(200).json({'business': 'business in added successfully'});
     })
     .catch(err => {
     res.status(400).send("unable to save to database");
     });
});

// Defined get data(index or listing) route
businessRoutes.route('/').get(function (req, res) {
     Business.find(function(err, businesses){
     if(err){
       console.log(err);
     }
     else {
       res.json(businesses);
     }
   });
});

// Defined edit route
businessRoutes.route('/edit/:id').get(function (req, res) {
   let id = req.params.id;
   Business.findById(id, function (err, business){
       res.json(business);
   });
});

//  Defined update route
businessRoutes.route('/update/:id').post(function (req, res) {
     Business.findById(req.params.id, function(err, business) {
     if (!business)
       res.status(404).send("data is not found");
     else {
         business.person_name = req.body.person_name;
         business.business_name = req.body.business_name;
         business.business_gst_number = req.body.business_gst_number;

```

```
          business.save().then(business => {
            res.json('Update complete');
        })
        .catch(err => {
              res.status(400).send("unable to update the database");
        });
      }
    });
});

// Defined delete | remove | destroy route
businessRoutes.route('/delete/:id').get(function (req, res) {
    Business.findByIdAndRemove({_id: req.params.id}, function(err, business){
        if(err) res.json(err);
        else res.json('Successfully removed');
    });
});

module.exports = businessRoutes;
```

Here, we have defined the **CRUD** operations for **MongoDB**. So when the request hits the server, it maps the URI and according to **URI**, the above function will be executed, and database operation will be performed and send the response to the client.
Here, we have used a **Mongoose ORM** to save, update, delete the data from the database.**Mongoose** is an **ORM** used in **MongoDB** database. Now, we have all the CRUD operations set up on the route file; we need to import inside the **server.js** file. So, our final **server.js** file looks like this.

```
// server.js

const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const PORT = 4000;
const cors = require('cors');
const mongoose = require('mongoose');
const config = require('./DB.js');
const businessRoute = require('./business.route');

mongoose.Promise = global.Promise;
mongoose.connect(config.DB, { useNewUrlParser: true }).then(
  () => {console.log('Database is connected') },
  err => { console.log('Can not connect to the database'+ err)}
);

app.use(cors());
app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());

app.use('/business', businessRoute);

app.listen(PORT, function(){
  console.log('Server is running on Port:',PORT);
});
```

Save the file and see the terminal and check if we got any errors.

#9: Install Axios library and send a POST request.

Install the **Axios library** and send the **POST** request. If you want to learn about the **Axios** library, then check out my **Getting Started With Axios Promise Based HTTP Client Tutorial Example** article.

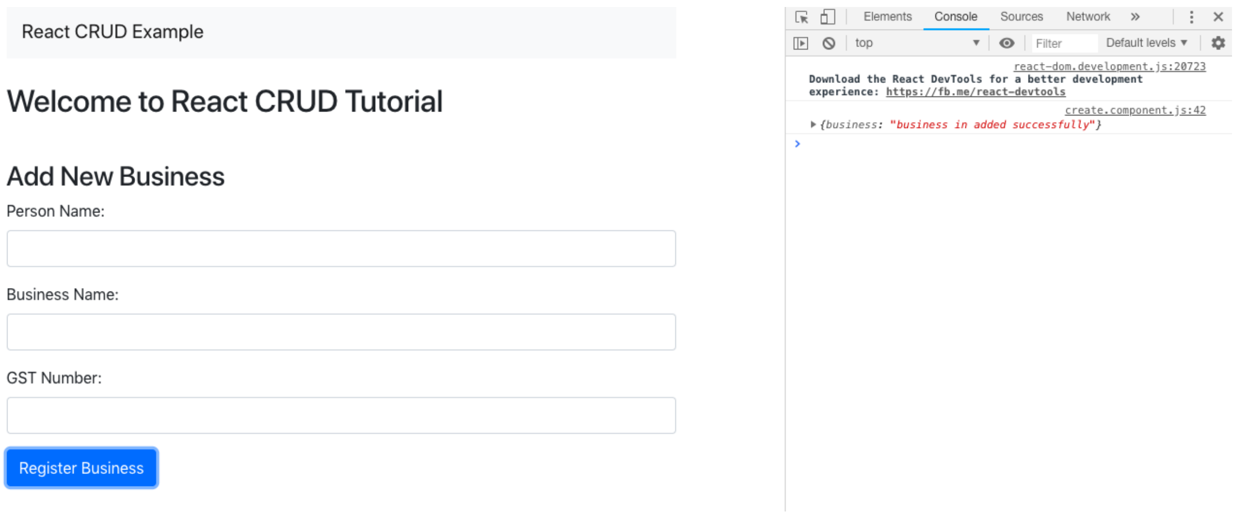
```
yarn add axios

# or

npm install axios --save
```

Now, send the HTTP POST request along with the form data to the **node js server**. We will send the data as an object because we have used the body-parser at the backend to pluck the data from the request and save it in the database.
Write the following code inside the **create.component.js** file.

```
// create.component.js

import React, { Component } from 'react';
import axios from 'axios';

export default class Create extends Component {
  constructor(props) {
    super(props);
    this.onChangePersonName = this.onChangePersonName.bind(this);
      this.onChangeBusinessName = this.onChangeBusinessName.bind(this);
      this.onChangeGstNumber = this.onChangeGstNumber.bind(this);
      this.onSubmit = this.onSubmit.bind(this);

      this.state = {
        person_name: '',
        business_name: '',
        business_gst_number:''
      }
    }
    onChangePersonName(e) {
      this.setState({
        person_name: e.target.value
      });
    }
    onChangeBusinessName(e) {
      this.setState({
        business_name: e.target.value
      })
    }
    onChangeGstNumber(e) {
      this.setState({
        business_gst_number: e.target.value
      })
    }

    onSubmit(e) {
      e.preventDefault();
      const obj = {
        person_name: this.state.person_name,
        business_name: this.state.business_name,
        business_gst_number: this.state.business_gst_number
      };
      axios.post('http://localhost:4000/business/add', obj)
          .then(res => console.log(res.data));

      this.setState({
        person_name: '',
        business_name: '',
        business_gst_number: ''
      })
    }

    render() {
      return (
          <div style={{ marginTop: 10 }}>
              <h3>Add New Business</h3>
              <form onSubmit={this.onSubmit}>
                  <div className="form-group">
                      <label>Person Name:  </label>
                      <input
                        type="text"
                        className="form-control"
                        value={this.state.person_name}
                        onChange={this.onChangePersonName}
                        />
                  </div>
                  <div className="form-group">
                      <label>Business Name: </label>
                      <input type="text"
                        className="form-control"
                        value={this.state.business_name}
                        onChange={this.onChangeBusinessName}
                        />
                  </div>
                  <div className="form-group">
                      <label>GST Number: </label>
                      <input type="text"
                        className="form-control"
                        value={this.state.business_gst_number}
                        onChange={this.onChangeGstNumber}
                        />
                  </div>
                  <div className="form-group">
                      <input type="submit" value="Register Business"
  className="btn btn-primary"/>
                  </div>
              </form>
          </div>
      )
    }
```

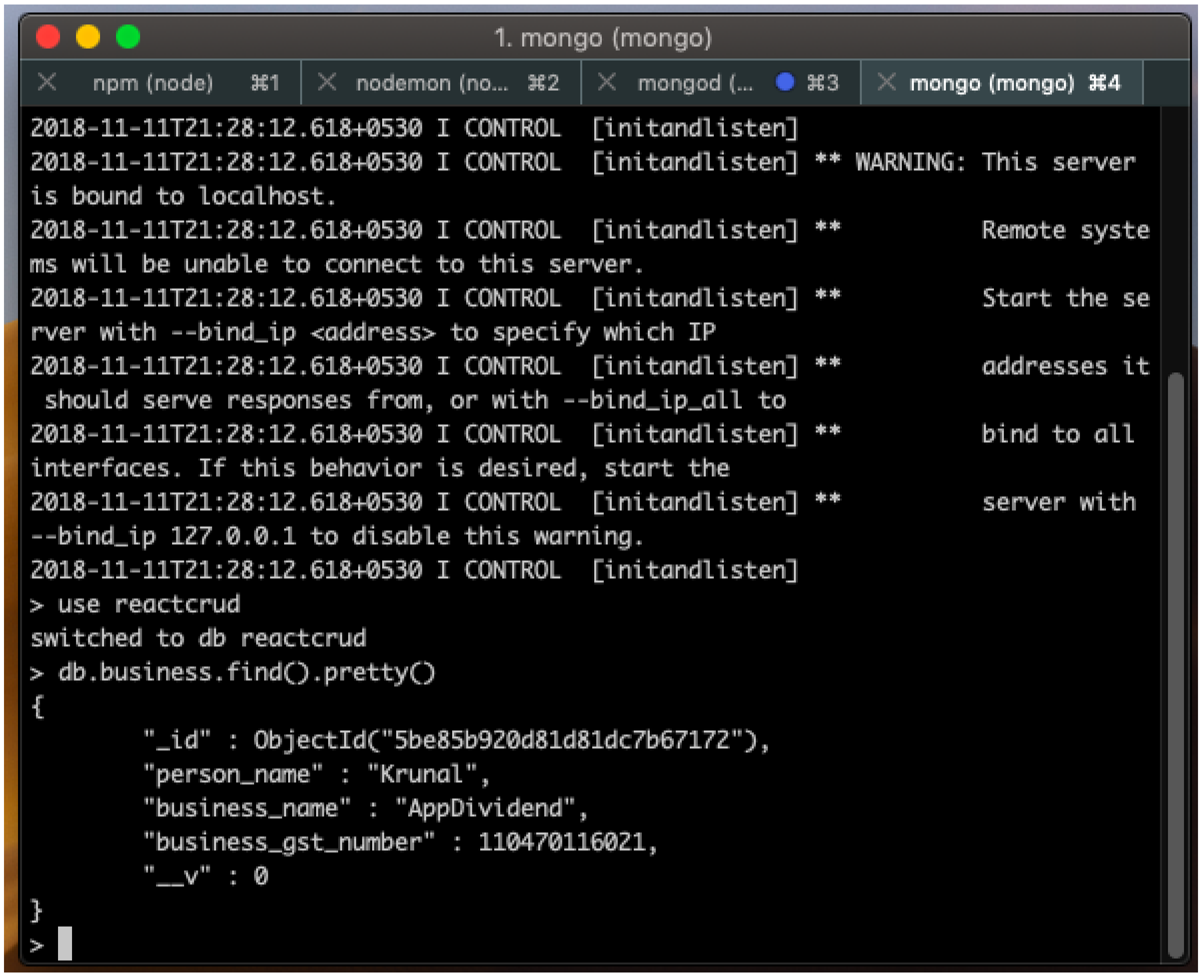
```
}
```

Now, submit the form with proper values and open your browser console panel and see the response.



```
                                               react-dom.development.js:20723
Download the React DevTools for a better development
experience: https://fb.me/react-devtools
                                                      create.component.js:42
▶{business: "business in added successfully"}
>
```

Also, now check inside the mongodb database and see the values.
To see the values inside the database, we can start a mongoshell and look at the values in the database.



```
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten]
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten] ** WARNING: This server
is bound to localhost.
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten] **          Remote syste
ms will be unable to connect to this server.
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten] **          Start the se
rver with --bind_ip <address> to specify which IP
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten] **          addresses it
 should serve responses from, or with --bind_ip_all to
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten] **          bind to all
interfaces. If this behavior is desired, start the
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten] **          server with
--bind_ip 127.0.0.1 to disable this warning.
2018-11-11T21:28:12.618+0530 I CONTROL  [initandlisten]
> use reactcrud
switched to db reactcrud
> db.business.find().pretty()
{
        "_id" : ObjectId("5be85b920d81d81dc7b67172"),
        "person_name" : "Krunal",
        "business_name" : "AppDividend",
        "business_gst_number" : 110470116021,
        "__v" : 0
}
>
```

So, we can see that our data is added successfully.

#10: Display the backend data

Write the following code inside the **index.component.js** file.

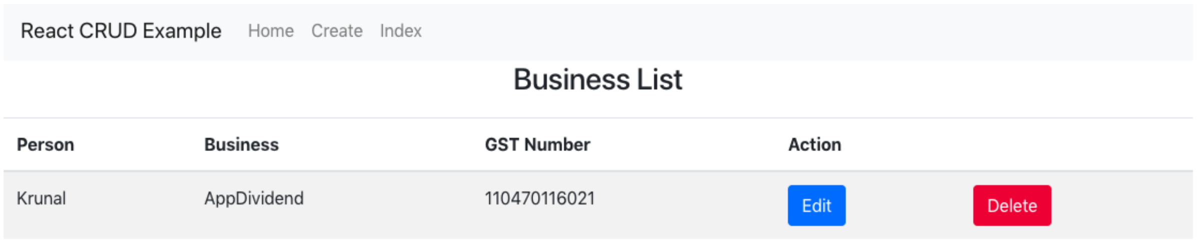
```
// index.component.js

import React, { Component } from 'react';
import axios from 'axios';
import TableRow from './TableRow';

export default class Index extends Component {

  constructor(props) {
        super(props);
        this.state = {business: []};
      }
      componentDidMount(){
        axios.get('http://localhost:4000/business')
          .then(response => {
            this.setState({ business: response.data });
          })
          .catch(function (error) {
            console.log(error);
          })
      }
      tabRow(){
        return this.state.business.map(function(object, i){
            return <TableRow obj={object} key={i} />;
        });
      }

      render() {
        return (
          <div>
            <h3 align="center">Business List</h3>
            <table className="table table-striped" style={{ marginTop: 20 }}>
              <thead>
                <tr>
                  <th>Person</th>
                  <th>Business</th>
                  <th>GST Number</th>
                  <th colSpan="2">Action</th>
                </tr>
              </thead>
              <tbody>
                { this.tabRow() }
              </tbody>
            </table>
          </div>
        );
      }
    }
```

So, here, we have sent the **GET** request to the **node.js** server and fetch that data from an API.

We have imported the **TableRow.js** component. So let us create that component. Inside the **components** folder, create one more component called **TableRow.js** and add the following code inside it.

```
// TableRow.js

import React, { Component } from 'react';

class TableRow extends Component {
  render() {
    return (
        <tr>
          <td>
            {this.props.obj.person_name}
          </td>
          <td>
            {this.props.obj.business_name}
          </td>
          <td>
            {this.props.obj.business_gst_number}
          </td>
          <td>
            <button className="btn btn-primary">Edit</button>
          </td>
          <td>
            <button className="btn btn-danger">Delete</button>
          </td>
        </tr>
    );
    }
  }

  export default TableRow;
```

This component is responsible for display the row data fetched from the backend. Save the file and go to the browser and see this URL: **http://localhost:3000/index.**



Business List

Person	Business	GST Number	Action	
Krunal	AppDividend	110470116021	Edit	Delete

#11: Edit and Update Functionality

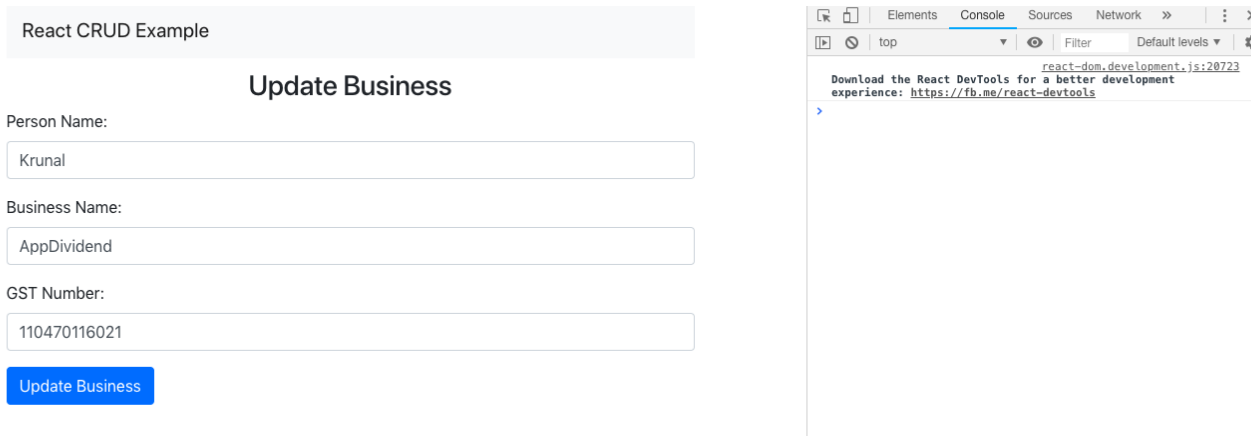
First, add the Link to the **TableRow.js** file.

```
// TableRow.js

import { Link } from 'react-router-dom';

<Link to={"/edit/"+this.props.obj._id} className="btn btn-
primary">Edit</Link>
```

Add the following code to the **edit.component.js** file.

```
// edit.component.js

import React, { Component } from 'react';
import axios from 'axios';

export default class Edit extends Component {
  constructor(props) {
    super(props);
    this.onChangePersonName = this.onChangePersonName.bind(this);
      this.onChangeBusinessName = this.onChangeBusinessName.bind(this);
      this.onChangeGstNumber = this.onChangeGstNumber.bind(this);
      this.onSubmit = this.onSubmit.bind(this);

      this.state = {
        person_name: '',
        business_name: '',
        business_gst_number:''
      }
    }

    componentDidMount() {
axios.get('http://localhost:4000/business/edit/'+this.props.match.params.id)
            .then(response => {
                this.setState({
                  person_name: response.data.person_name,
                  business_name: response.data.business_name,
                  business_gst_number: response.data.business_gst_number});
            })
            .catch(function (error) {
                console.log(error);
            })
      }

    onChangePersonName(e) {
      this.setState({
        person_name: e.target.value
      });
    }
    onChangeBusinessName(e) {
      this.setState({
        business_name: e.target.value
      })
    }
    onChangeGstNumber(e) {
      this.setState({
        business_gst_number: e.target.value
      })
    }

    onSubmit(e) {
      e.preventDefault();
      const obj = {
        person_name: this.state.person_name,
        business_name: this.state.business_name,
        business_gst_number: this.state.business_gst_number
      };
axios.post('http://localhost:4000/business/update/'+this.props.match.params.id, obj)
          .then(res => console.log(res.data));

      this.props.history.push('/index');
    }

    render() {
      return (
          <div style={{ marginTop: 10 }}>
              <h3 align="center">Update Business</h3>
              <form onSubmit={this.onSubmit}>
                  <div className="form-group">
                      <label>Person Name:  </label>
                      <input
                        type="text"
                        className="form-control"
                        value={this.state.person_name}
                        onChange={this.onChangePersonName}
                        />
                  </div>
                  <div className="form-group">
                      <label>Business Name: </label>
                      <input type="text"
                        className="form-control"
                        value={this.state.business_name}
                        onChange={this.onChangeBusinessName}
                        />
                  </div>
                  <div className="form-group">
                      <label>GST Number: </label>
                      <input type="text"
                        className="form-control"
                        value={this.state.business_gst_number}
                        onChange={this.onChangeGstNumber}
                        />
                  </div>
                  <div className="form-group">
                      <input type="submit"
                        value="Update Business"
                        className="btn btn-primary"/>
                  </div>
              </form>
          </div>
      )
    }
}
```

So, what we have done is, we have used the **component lifecycle method** to fetch the data from the **API**.
That data needs to be displayed inside the textbox because this is edit form. Next, it is the same thing as we have written the code in the **create.component.js** file.
Save the file and go to the edit page from the index or listing page. If the data is not displayed then please first check that you have the running node.js server on the backend.



```
react-dom.development.js:20723
Download the React DevTools for a better development
experience: https://fb.me/react-devtools
```

We have also written the code that will update the data because we have written the function that will send the request to the **node.js** server and update the data based on the ID.

#12: Delete the data

Now, the only thing remaining is to delete the data. So define the delete function inside **TableRow.js** file.

```
// TableRow.js

import React, { Component } from 'react';
import { Link } from 'react-router-dom';
import axios from 'axios';

class TableRow extends Component {

  constructor(props) {
          super(props);
          this.delete = this.delete.bind(this);
      }
      delete() {
axios.get('http://localhost:4000/business/delete/'+this.props.obj._id)
              .then(console.log('Deleted'))
              .catch(err => console.log(err))
      }
    render() {
      return (
          <tr>
            <td>
              {this.props.obj.person_name}
            </td>
            <td>
              {this.props.obj.business_name}
            </td>
            <td>
              {this.props.obj.business_gst_number}
            </td>
            <td>
              <Link to={"/edit/"+this.props.obj._id} className="btn btn-primary">Edit</Link>
            </td>
            <td>
              <button onClick={this.delete} className="btn btn-danger">Delete</button>
            </td>
          </tr>
      );
    }
  }

  export default TableRow;
```

Now, we have completed **React CRUD Example** or **MERN Stack Tutorial** From Scratch. I have put this code on **Github**. So you can check that code as well.
Finally, **React CRUD Example Tutorial** is over. Thanks for taking.