# Full Stack Tutorial on CRUD Operations

CRUD Tutorial using Node JS, Express, React JS, and MySQL (Full-Stack)
by M Fikri Setiadi

Tutorial adapted to Ubuntu 20 cloud instances
by Maruthi S. Inukonda, Nilesh S. Kale

**CAUTION: Donot use sudo with nvm, npm (used to install nodejs packages). Any nodejs packages running as root user may compromise the system via your webapp.**

## Setup VSCode & Remote ssh terminal to your cloud instance

Open two VSCode terminals (say #1 - frontend, #2 - backend, #3 - misclaneous)

## Install NodeJS using Node Version Manager

~~In the VSCode terminal # 3:~~

```
~~$ curl -o-~~
~~https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash~~

~~$ exec bash~~
~~$ nvm install v16.15.0~~
~~$ npm install -g npm@8.12.0~~
~~$ npm install -g npm@8.12.1~~
$ npm install -g nodemon

~~$ npm install -g create-react-app~~
~~$ npm install -g react-router-dom~~
```

## Auto generate (front-end only) app

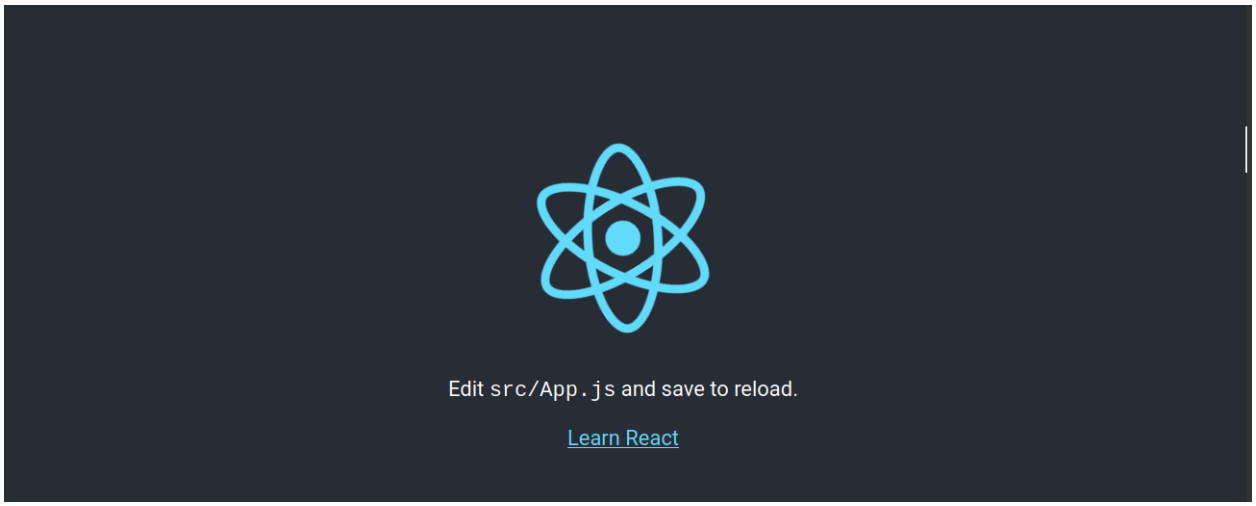
In the VSCode terminal # 1:

```
$ mkdir -p ~/fullstack/frontend
$ cd ~/fullstack
$ npx create-react-app frontend
$ cd ~/fullstack/frontend
$ npm start
Compiled successfully!
You can now view frontend in the browser.
  Local:        http://localhost:3000
  On Your Network: http://192.168.51.xx:3000
```

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

webpack compiled successfully
```

## Test the auto-generated app Frontend

Open the browser and enter your cloud instances' IP-address:portno (192.168.51.xx:3000) in address bar




## Develop database tier using MariaDB (MySQL)

## #1 Setup MySQL database and tables

In the VSCode terminal # 3:

```
$ mkdir -p ~/fullstack/backend_mysql

$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

```
Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] Y
New password: 123123
Re-enter new password: 123123
Password updated successfully!
Reloading privilege tables..
 ... Success!

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y

Remove anonymous users? [Y/n] Y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.
```

```
Thanks for using MariaDB!
```

Now authorize user ubuntu to manage databases and tables.

```
$ sudo mysql -u root -p
Enter Password: 123123
MariaDB [(none)]> use mysql;
MariaDB [mysql]> SELECT User, Host, plugin FROM mysql.user;

MariaDB [mysql]> CREATE USER 'ubuntu'@'localhost' IDENTIFIED BY '321321';
MariaDB [mysql]> GRANT ALL PRIVILEGES ON *.* TO 'ubuntu'@'localhost';
MariaDB [mysql]> UPDATE user SET plugin='mysql_native_password' WHERE User='ubuntu';
MariaDB [mysql]> SELECT User, Host, plugin FROM mysql.user;
+--------+-----------+-----------------------+
| User   | Host      | plugin                |
+--------+-----------+-----------------------+
| root   | localhost | unix_socket           |
| ubuntu | localhost | mysql_naive_password |
+--------+-----------+---------t-------------+
MariaDB [mysql]> FLUSH PRIVILEGES;
MariaDB [mysql]> exit;
```

Create database and tables required for the CRUD demo application.

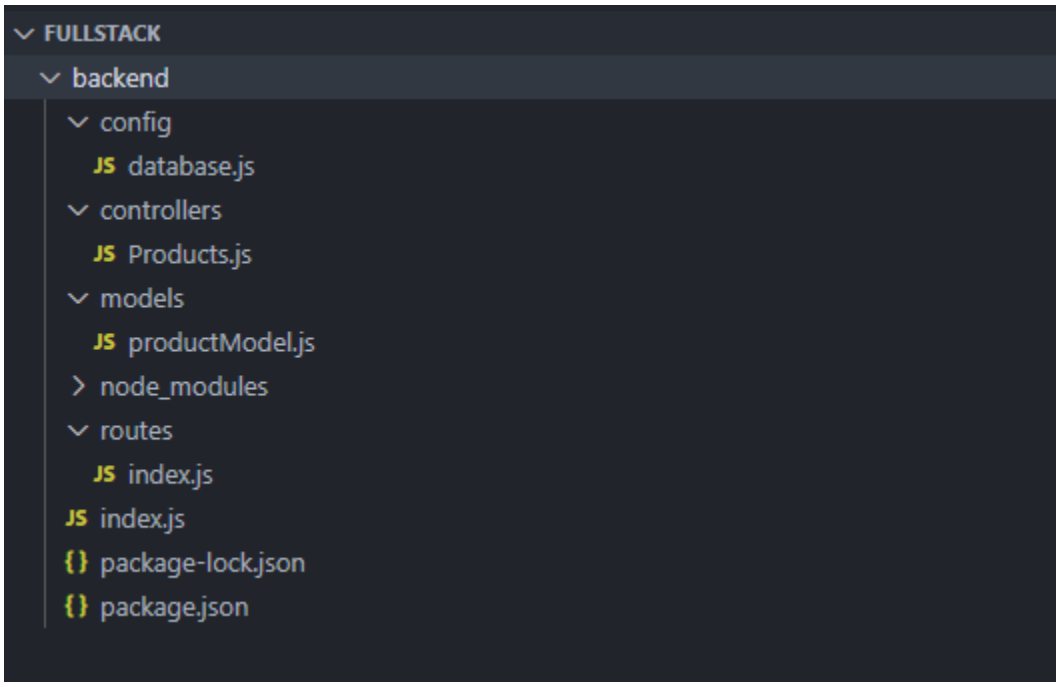
```
$ mysql -u ubuntu -p
Enter password: 321321
MariaDB [(none)]> CREATE DATABASE mern_db;
MariaDB [(none)]> use mern_db;
MariaDB [mern_db]> use mern_db;
MariaDB [mern_db]> CREATE TABLE products(
       id INT(11) PRIMARY KEY AUTO_INCREMENT,
       title VARCHAR(200),
       price DOUBLE,
       createdAt DATE,
       updatedAt DATE
       )ENGINE=INNODB;
MariaDB [mern_db]> exit
```

# #2 Develop the application tier

To make the application more structured neatly, we will apply the MVC (Model-View-Controllers) pattern.

From VSCode IDE, in the “backend” folder, create the following sub-folders : “config”, “controllers”, “models”, and “routes”.

Then create a “database.js” file in the “config” folder, create a “Products.js” file in the “controllers” folder, create a “productModel.js” file in the “models” folder, create a “index.js” file in the “routes” folder, and create the “index.js” file in the “backend” folder.




## #2.1 Database connection

Open the "database.js" file in the "config" folder, then type the following code:

```
import { Sequelize } from "sequelize";
const db = new Sequelize('mern_db', 'ubuntu', '321321', {
   host: "localhost",
   dialect: "mysql"
});
export default db;
```

## #2.2 MVC Model

Open the model file "productModel.js" which is in the "models" folder, then type the following code:

```
import { Sequelize } from "sequelize";
import db from "../config/database.js";
```

```
const { DataTypes } = Sequelize;
const Product = db.define('products', {
   title: {
       type: DataTypes.STRING
   },
   price: {
       type: DataTypes.DOUBLE
   }
}, {
   freezeTableName: true
});
export default Product;
```

## #2.3 MVC Controller

Open the controller file "Products.js" which is in the "controllers" folder, then type the following code:

```
import Product from "../models/productModel.js";
export const getAllProducts = async (req, res) => {
   try {
       const products = await Product.findAll();
       res.json(products);
   } catch (error) {
       res.json({ message: error.message });
   }
}
export const getProductById = async (req, res) => {
   try {
       const product = await Product.findAll({
           where: {
               id: req.params.id
           }
       });
       res.json(product[0]);
   } catch (error) {
       res.json({ message: error.message });
   }
}
export const createProduct = async (req, res) => {
```

```
    try {
        await Product.create(req.body);
        res.json({
            "message": "Product Created"
        });
    } catch (error) {
        res.json({ message: error.message });
    }
}
export const updateProduct = async (req, res) => {
    try {
        await Product.update(req.body, {
            where: {
                id: req.params.id
            }
        });
        res.json({
            "message": "Product Updated"
        });
    } catch (error) {
        res.json({ message: error.message });
    }
}
export const deleteProduct = async (req, res) => {
    try {
        await Product.destroy({
            where: {
                id: req.params.id
            }
        });
        res.json({
            "message": "Product Deleted"
        });
    } catch (error) {
        res.json({ message: error.message });
    }
}
```

## #2.4 Routes (REST API endpoints)

Open the “index.js” file located in the “backend_mysql/routes” folder, then type the following code:

```
import express from "express";
import {
  getAllProducts,
  createProduct,
  getProductById,
  updateProduct,
  deleteProduct
} from "../controllers/Products.js";
const router = express.Router();
router.get('/', getAllProducts);
router.get('/:id', getProductById);
router.post('/', createProduct);
router.patch('/:id', updateProduct);
router.delete('/:id', deleteProduct);
export default router;
```

## #2.5 Entry endpoint

Open the “index.js” file located in the “backend_mysql” folder, then type the following code:

```
import express from "express";
import db from "./config/database.js";
import productRoutes from "./routes/index.js";
import cors from "cors";
const app = express();
try {
   await db.authenticate();
   console.log('Database connected...');
} catch (error) {
   console.error('Connection error:', error);
}
app.use(cors());
app.use(express.json());
app.use('/products', productRoutes);
app.listen(5000, () => console.log('Server running at port 5000'));
```

## #2.6 Test the application tier (RESTFul APIs)

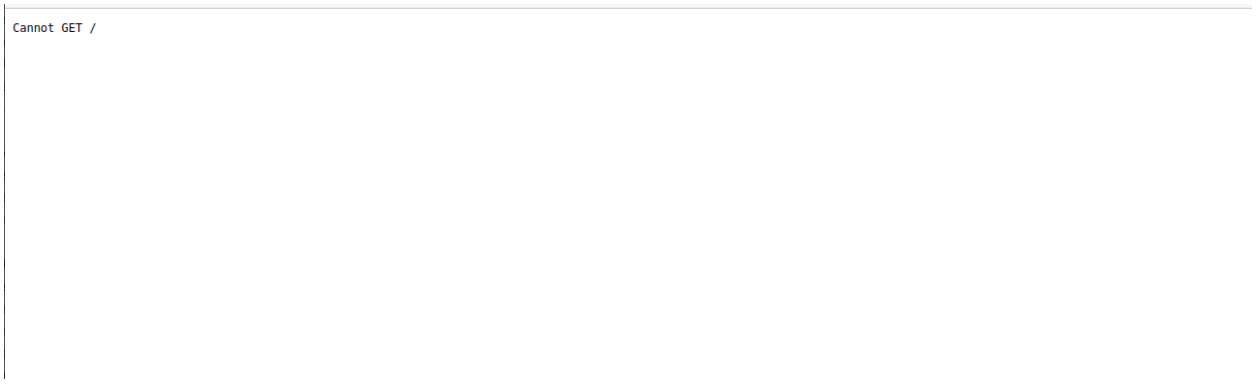
In the VSCode terminal # 2:

```
$ cd ~/fullstack/backend_mysql
$ npm init -y
$ npm install express mysql2 sequelize cors
```

Add the following line below the “description” to the package.json file in the backend folder.
“type” : “module”,

```
$ nodemon index
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Executing (default): SELECT 1+1 AS result
Database connected...
Server running at port 5000
```

Open the browser and enter your cloud instances’ IP-address:portno (192.168.51.xx:5000) in address bar




# Develop database tier using MongoDB

## #1 Setup MongoDB database and collection

In the VSCode terminal # 3:

```
$ mkdir -p ~/fullstack/backend_mongo

ubuntu@sdfxx:~/fullstack.mongo$ mongo --quiet -host localhost
>
show dbs
admin   0.000GB
config  0.000GB
local   0.000GB

> use mern_db
switched to db mern_db

> db.createCollection("products")
{ "ok" : 1 }

> show collections
products

> db.products.find()

> quit()
```

# #2 Develop the application tier

## #2.1 Database connection

```
import { MongoClient } from "mongodb";

var url = "mongodb://localhost:27017/mern_db";
var client = new MongoClient(url);
await client.connect();
const db = client.db();
console.log("database connected!");

export default db;
```

## #2.2 MVC Model

Open the model file "productModel.js" which is in the "backend_mongo/models" folder, then type the following code:

```
import db from "../config/database.js";
```

```
const Product = db.collection('products',
{
   title: "",
   price: 0.0
}, {
   freezeTableName: true
});
console.log("collection retrieved!");

export default Product;
```

## #2.3 MVC Controller

Open the controller file "Products.js" which is in the "backend_mongo/controllers" folder, then type the following code:

```
import Product from "../models/productModel.js";
import { createRequire } from 'module';
const require = createRequire(import.meta.url);

export const getAllProducts = async (req, res) => {
   try {
       const products = await Product.find().toArray();
         const retProducts = products.map(item => ({ id: item._id, title:
item.title, price: item.price }));
       // console.log("retProducts:", retProducts)
       res.json(retProducts);
   } catch (error) {
       console.log(error.message)
       res.json({ message: error.message });
   }
}
export const getProductById = async (req, res) => {
   try {
       const mongodb = require("mongodb");
       const prodid = new mongodb.ObjectId(req.params.id);
       const product = await Product.findOne({
           _id: prodid
       });
```

```
        const retProduct = { id: product._id, title: product.title, price:
product.price };
        // console.log("Returning:", retProduct)
        res.json(retProduct);
    } catch (error) {
        console.log(error.message)
        res.json({ message: error.message });
    }
}
export const createProduct = async (req, res) => {
    try {
        await Product.insertOne(req.body);
        res.json({
            "message": "Product Created"
        });
    } catch (error) {
        console.log(error.message)
        res.json({ message: error.message });
    }
}
export const updateProduct = async (req, res) => {
    try {
        const mongodb = require("mongodb");
        const prodid = new mongodb.ObjectId(req.params.id);
        await Product.updateOne(
            {
                _id: prodid
            },
            {
                $set: {
                    title: req.body.title,
                    price: req.body.price
                }
            }
        );
        res.json({
            "message": "Product Updated"
        });
    } catch (error) {
        console.log(error.message)
```

```
        res.json({ message: error.message });
    }
}
export const deleteProduct = async (req, res) => {
    try {
        const mongodb = require("mongodb");
        const prodid = new mongodb.ObjectId(req.params.id);
        Product.deleteOne(
            {
                _id: prodid
            },
            function (err, obj) {
                if (err) throw err;
            });
        res.json({
            "message": "Product Deleted"
        });
    } catch (error) {
        console.log(error.message);
        res.json({ message: error.message });
    }
}
```

## #2.4 Routes (REST API endpoints)

Open the “index.js” file located in the “backend_mongo/routes” folder, then type the following code:

```
import express from "express";
import {
    getAllProducts,
    createProduct,
    getProductById,
    updateProduct,
    deleteProduct
} from "../controllers/Products.js";
const router = express.Router();
router.get('/', getAllProducts);
router.get('/:id', getProductById);
router.post('/', createProduct);
```

```
router.patch('/:id', updateProduct);
router.delete('/:id', deleteProduct);
console.log("routes setup!");
export default router;
```

## #2.5 Entry endpoint

Open the “index.js” file located in the “backend_mongo” folder, then type the following code:

```
import express from "express";
import db from "./config/database.js";
import productRoutes from "./routes/index.js";
import cors from "cors";
const app = express();
app.use(cors());
app.use(express.json());
app.use('/products', productRoutes);
app.listen(5000, () => console.log('Server running at port 5000'));
```

## #2.6 Test the application tier

In the VSCode terminal # 2:

Stop the `nodemon index` command if it was running within the backend_sql

```
$ cd ~/fullstack/backend_mongo
$ npm init -y
$ npm install express cors mongodb

$ nodemon index
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
database connected!
collection retrieved!
routes setup!
Server running at port 5000
```

## Develop the frontend code

In the VSCode terminal # 1:

```
$ cd ~/fullstack/frontend
$ npm install -S --save react-router-dom
$ npm install axios
```

From VSCode IDE, in the “frontend/src” sub-folder, create the sub-folder: “components”.

Then create “AddProduct.js”, “EditProduct.js”, “ProductList.js” files in the “frontend/src/components” folder. Also create “App.js” in the “frontend/src” folder.

Open the “AddProduct.js” file located in the “components” folder, then type the following code:

```
import { useState } from 'react'
import axios from "axios";
import { useNavigate } from 'react-router-dom';
const AddProduct = () => {
   const [title, setTitle] = useState('');
   const [price, setPrice] = useState('');
   const history = useNavigate();
   const saveProduct = async (e) => {
       e.preventDefault();
       await axios.post('http://192.168.51.xx:5000/products', {
           title: title,
           price: price
       });
       history.push("/");
   }
   return (
       <div>
           <form onSubmit={saveProduct}>
               <div className="field">
                   <label className="label">Title</label>
                   <input
                       className="input"
                       type="text"
                       placeholder="Title"
                       value={title}
                       onChange={(e) => setTitle(e.target.value)}
                   />
               </div>
```

```
                <div className="field">
                    <label className="label">Price</label>
                    <input
                        className="input"
                        type="text"
                        placeholder="Price"
                        value={price}
                        onChange={(e) => setPrice(e.target.value)}
                    />
                </div>
                <div className="field">
                    <button className="button is-primary">Save</button>
                </div>
            </form>
        </div>
    )
}

export default AddProduct
```

Open the “EditProduct.js” file located in the “components” folder, then type the following code:

```
/* eslint-disable react-hooks/exhaustive-deps */
import { useState, useEffect } from 'react'
import axios from "axios";
import { useNavigate, useParams } from 'react-router-dom';
const EditProduct = () => {
    const [title, setTitle] = useState('');
    const [price, setPrice] = useState('');
    const history = useNavigate();
    const { id } = useParams();
    const updateProduct = async (e) => {
        e.preventDefault();
        await axios.patch(`http://192.168.51.xx:5000/products/${id}`, {
            title: title,
            price: price
        });
        history.push("/");
    }
    useEffect(() => {
```

```
        getProductById();
    }, []);
    const getProductById = async () => {
                                        const      response      =      await
axios.get(`http://192.168.51.xx:5000/products/${id}`);
        setTitle(response.data.title);
        setPrice(response.data.price);
    }
    return (
        <div>
            <form onSubmit={updateProduct}>
                <div className="field">
                    <label className="label">Title</label>
                    <input
                        className="input"
                        type="text"
                        placeholder="Title"
                        value={title}
                        onChange={(e) => setTitle(e.target.value)}
                    />
                </div>
                <div className="field">
                    <label className="label">Price</label>
                    <input
                        className="input"
                        type="text"
                        placeholder="Price"
                        value={price}
                        onChange={(e) => setPrice(e.target.value)}
                    />
                </div>
                <div className="field">
                    <button className="button is-primary">Update</button>
                </div>
            </form>
        </div>
    )
}
export default EditProduct
```

Open the “ProductList.js” file located in the “components” folder, then type the following code:

```
import { useState, useEffect } from 'react'
import axios from "axios";
import { Link } from "react-router-dom";
const ProductList = () => {
   const [products, setProduct] = useState([]);
   useEffect(() => {
       getProducts();
       }, []
   );
   const getProducts = async () => {
                                       const     response     =     await
axios.get('http://192.168.51.xx:5000/products');
       setProduct(response.data);
   }
   const deleteProduct = async (id) => {
       await axios.delete(`http://192.168.51.xx:5000/products/${id}`);
       getProducts();
   }
   return (
       <div>
           <h1> Product List </h1>
                  <Link  to="/add"  className="button  is-primary  mt-2">Add
New</Link>
           <table className="table is-striped is-fullwidth">
           <thead>
               <tr>
               <th>No</th>
               <th>Title</th>
               <th>Price</th>
               <th>Actions</th>
               </tr>
           </thead>
           <tbody>
                   { products.map((product, index) => (
                       <tr key={ product.id }>
                       <td>{ index + 1 }</td>
                       <td>{ product.title }</td>
```

```
                        <td>{ product.price }</td>
                        <td>
                         <Link to={`/edit/${product.id}`} className="button
is-small
                        is-info">Edit</Link>
                        <button onClick={ () => deleteProduct(product.id) }
className="button
                        is-small is-danger">Delete</button>
                        </td>
                        </tr>
                    )) }
            </tbody>
            </table>
        </div>
    )
}
export default ProductList
```

Open the “App.js” file located in the “frontend/src” folder, then type the following code:

```
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import ProductList from "./components/ProductList";
import AddProduct from "./components/AddProduct";
import EditProduct from "./components/EditProduct";

function App() {
 return (
   <Router>
     <div className="container">
       <div className="columns">
         <div className="column is-half is-offset-one-quarter">
           <Routes>
             <Route exact path="/" element={<ProductList />} />

             <Route path="/add" element={<AddProduct />} />

             <Route path="/edit/:id" element={<EditProduct />} />

           </Routes>
```

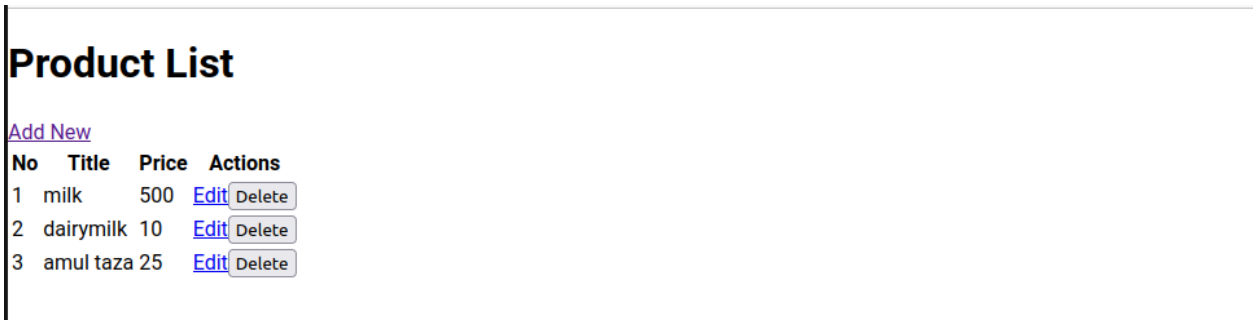
```
          </div>
        </div>
      </div>
    </Router>
  );
}



export default App
```

## #8 Test the complete App

In the VSCode terminal # 1:

```
$ cd ~/fullstack/frontend
$ npm start
```

Open the browser and enter your cloud instances' IP-address:portno (192.168.51.xx:3000) in address bar




## #9 Extend the App to include authentication

**NOTICE: This part is optional for the assignment.**

## References

https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04
https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04

https://stackoverflow.com/questions/63124161/attempted-import-error-switch-is-not-exported-from-react-router-dom
https://stackoverflow.com/questions/55568697/blank-page-after-running-build-on-create-react-app

https://www.w3schools.com/js/js_json_intro.asp
https://www.w3schools.com/nodejs/nodejs_mongodb.asp