Name: Sanjay Khanna S

Emp Id: 12124

Spring Boot & Microservices

Lab Assessment

Question: Case Study

9) Inventory Management System

The three actors in this system are the customer, the Dealer, and the retailer. Retailers can list their products or business, which the admin can check. The product is available for purchase from the merchant.

Dealer:

Dealer Id: Int

Dealer Name: String

Dealer Mobile Number: Long

Dealer Address: String

Retailer:

ProductId: Int

ProductName: String

ProductType: String

Price: Float

Quantity: Int

Customer:

CustomerId: Int

CustomerName: String

Customer Mobile Number: Long

Payment method: String

W3H Analysis

What	How
Who are the Actors involving in this management System? **Ans:** Dealer Retailer Customer **What can we do with Retailer in this management system?** **Ans:** List product Add new product Remove new product Update the product specifications **What can we do with dealer in this management System?** **Ans:** List dealer Details Add dealer Delete dealer Modify dealer details **What can we do with customer in this management system?** **Ans:** List customer details Add customer Delete customer Update customer **What is the Relationship between Product and Customers?** **Ans:** Many to one Customer can have Many Product **What is the Relationship Between products and dealers?** **Ans:** Many to one Dealer can provide many product	**How can we do the add/delete/update of dealer?** **Ans:** by using dealer id by using dealer name By using mobile name **How can we auto populate in dealer for product relation?** **Ans:** By using product id By using product name; By using product type **How can we do the add/delete/update of product?** **Ans:** By using product id By using product type By using product name **How can we do the add/delete/update of Customer?** Ans: By using customer id By using customer name By using mobile number **How can we auto populate in customer for product relation?** **Ans:** By using product id By using product name; By using product type **How can we do filter the list process in all tables?** **Ans:** By using their id By using their name By other fields

	How can we achieve many to one relationship between dealer and product? **Ans:** By joining column product id with dealer By joining column product name with dealer By joining column product name with dealer **How can we achieve many to one relationship between customer and product?** **Ans:** By joining column product id with customer By joining column product name with customer By joining column product name with customer
Why	**Why Not**
How can we do the add/delete/update of dealer? **Ans:** by using dealer id (Because Dealer Id is a unique one, so we can set as a primary key) **How can we auto populate in dealer for product relation?** **Ans:** By using product id By using product name; (Because product names are easy understandable by all, easy to select) **How can we do the add/delete/update of product?** **Ans:** By using product id (Because product Id is a unique one, so we can set as a primary key)	**How can we do the add/delete/update of dealer?** **Ans:** (Because other fields may not be unique one, Chance of duplication) **How can we auto populate in dealer for product relation?** **Ans:** (Because product ids are auto generated, so not found easy to select) **How can we do the add/delete/update of product?** **Ans:** (Because other fields may not be unique one, Chance of duplication) **How can we do the add/delete/update of Customer?** Ans:

How can we do the add/delete/update of Customer? Ans: By using customer id (Because Customer Id is a unique one, so we can set as a primary key) **How can we auto populate in customer for product relation?** **Ans:** By using product name; (Because product names are easy understandable by all, easy to select) **How can we do filter the list process in all tables?** **Ans:** By using their name (Because, Names are string, it can filter out using some inbuilt methods) **How can we achieve many to one relationship between dealer and product?** **Ans:** By joining column product id with dealer (Because product Id is a primary key which is unique and not null, so it can be a foreign key for joining) **How can we achieve many to one relationship between customer and product?** **Ans:** By joining column product id with customer (Because product Id is a primary key which is unique and not null, so it can be a foreign key for joining)	(Because other fields may not be unique one, Chance of duplication) **How can we auto populate in customer for product relation?** **Ans:** (Because product ids are auto generated, so not found easy to select) **How can we do filter the list process in all tables?** **Ans:** (Because other field may or may not be string so some method may not available) **How can we achieve many to one relationship between dealer and product?** **Ans:** (Because primary keys which are unique and not null can be foreign keys for joining) **How can we achieve many to one relationship between customer and product?** **Ans:** (Because primary keys which are unique and not null can be foreign keys for joining)

Algorithms

Similar Algorithms for the Customer, Product and Dealer

Customer:

Step 0: Start

Step 1: Open the browser

Step 2: Goto Main Page

Step 3: Click to customer in Navbar

Step 4: Redirect to customer page

Step 5: List of Customer record with action buttons

Step 6: click add customer

Step 7: Insert form box will popup

Step 8: Enter their details

Step 9: select product name;

Step 10: click insert to submit

Step 11: message will popup

Step 11.1: if insertion successful then success message popup

Step 11: 2: if insertion not successful the failure message popup

Step 12: Able to view the added record in list with edit and delete button

Step 13: Click edit

Step 14: Update form box will pop up

Step 15: Auto populate the record details

Step 16: Modify the details

Step 17: Click update to submit

Step 18: message will popup

Step 11.1: if update successful then success message popup

Step 18 2: if update not successful the failure message popup

Step 19: Able to view the updated record in list with edit and delete button

Step 20: Click delete

Step 21: delete form box will pop up with confirmation

Step 22: click confirm to submit

Step 23: message will popup

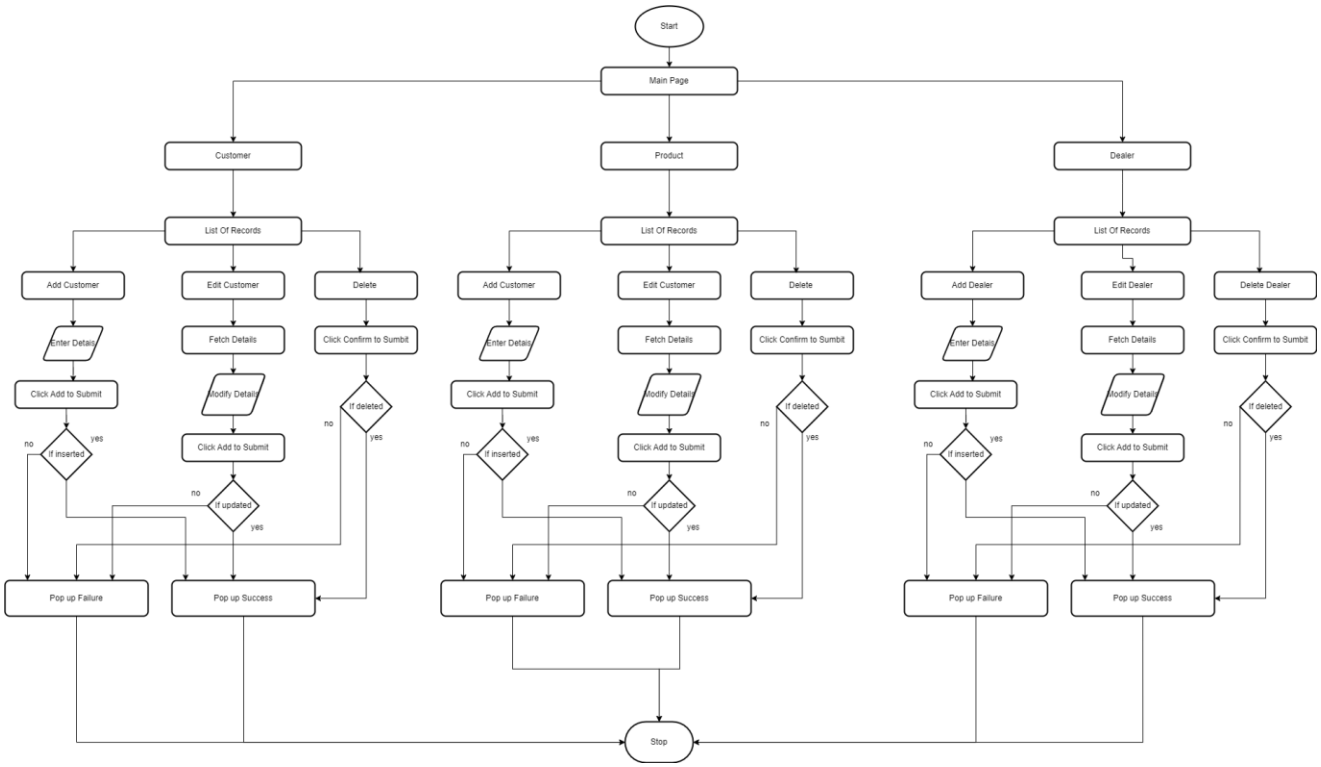
Step 23.1: if deletion successful then success message popup

Step 23: 2: if deletion not successful the failure message popup

Step 24: Able to view the list

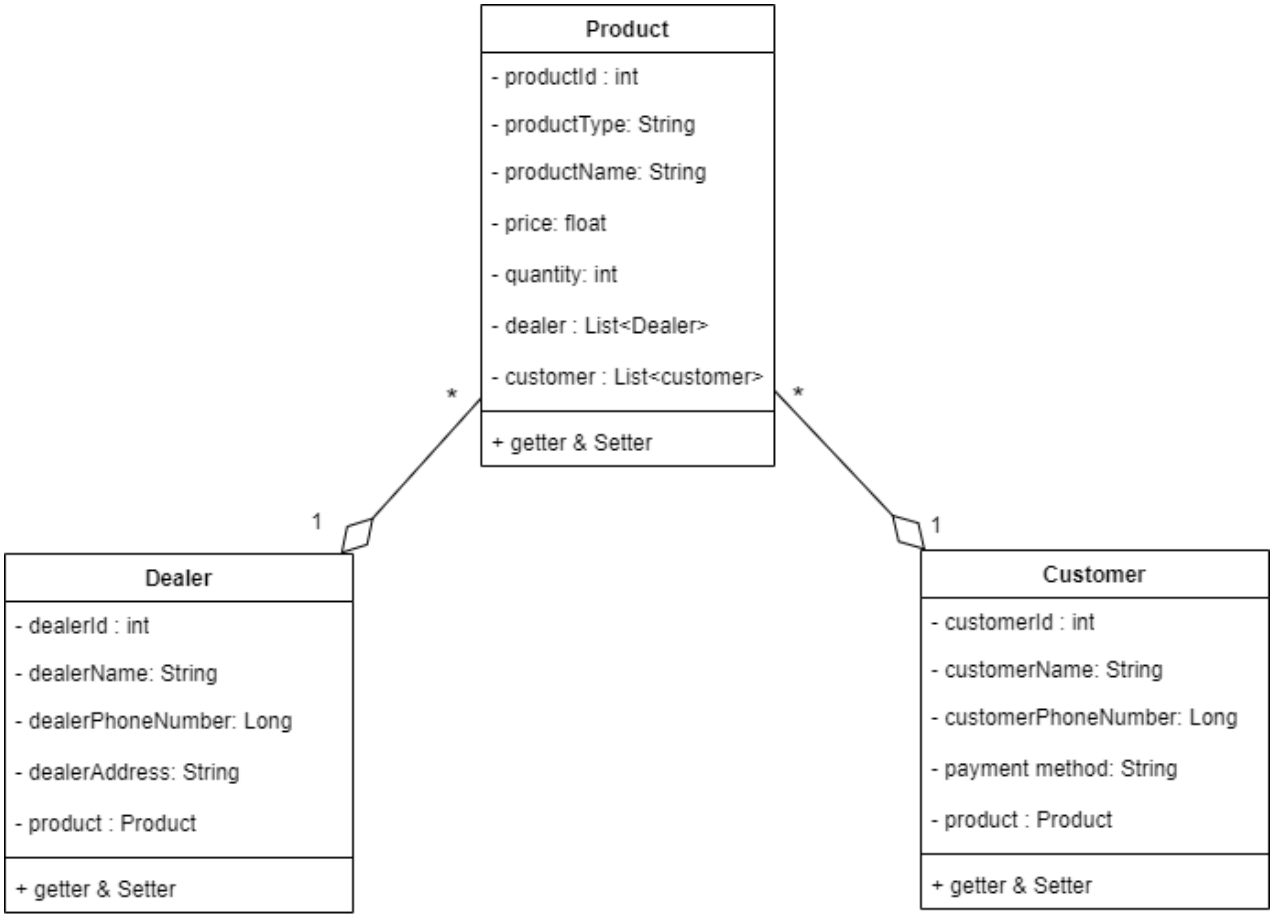
Step 25: stop

Flow Chart:

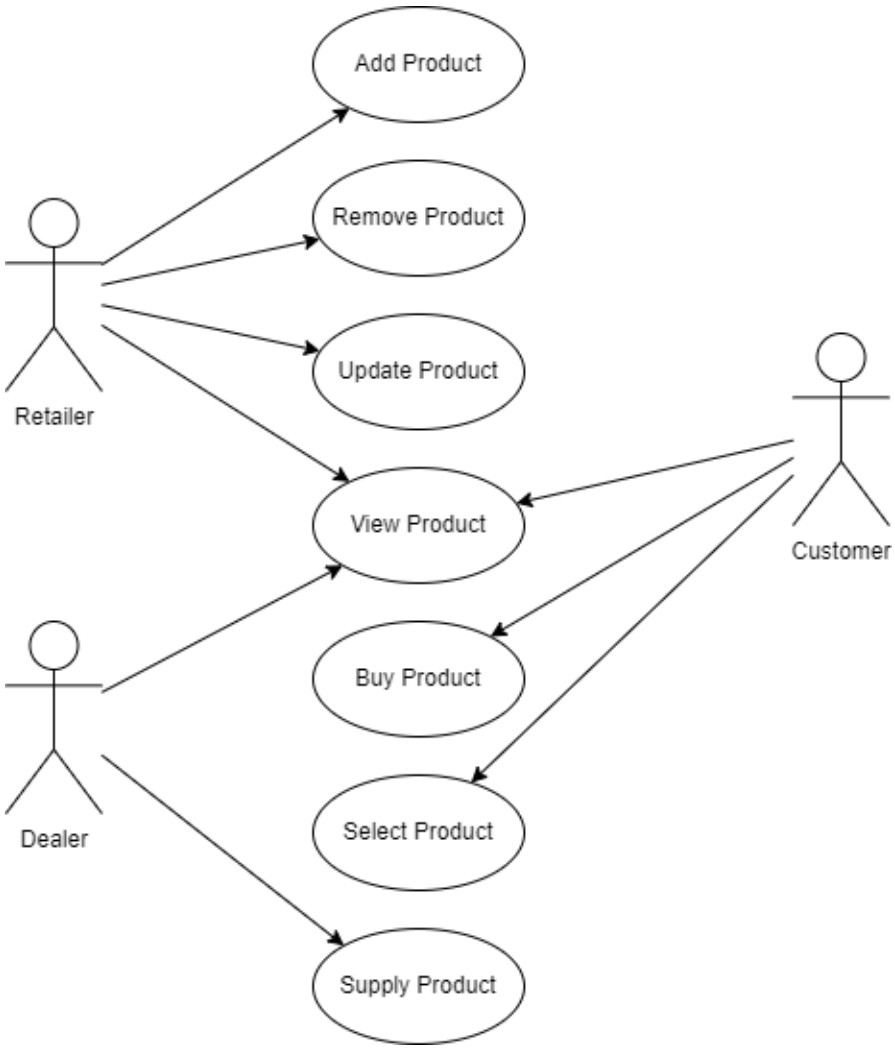


UML Diagrams

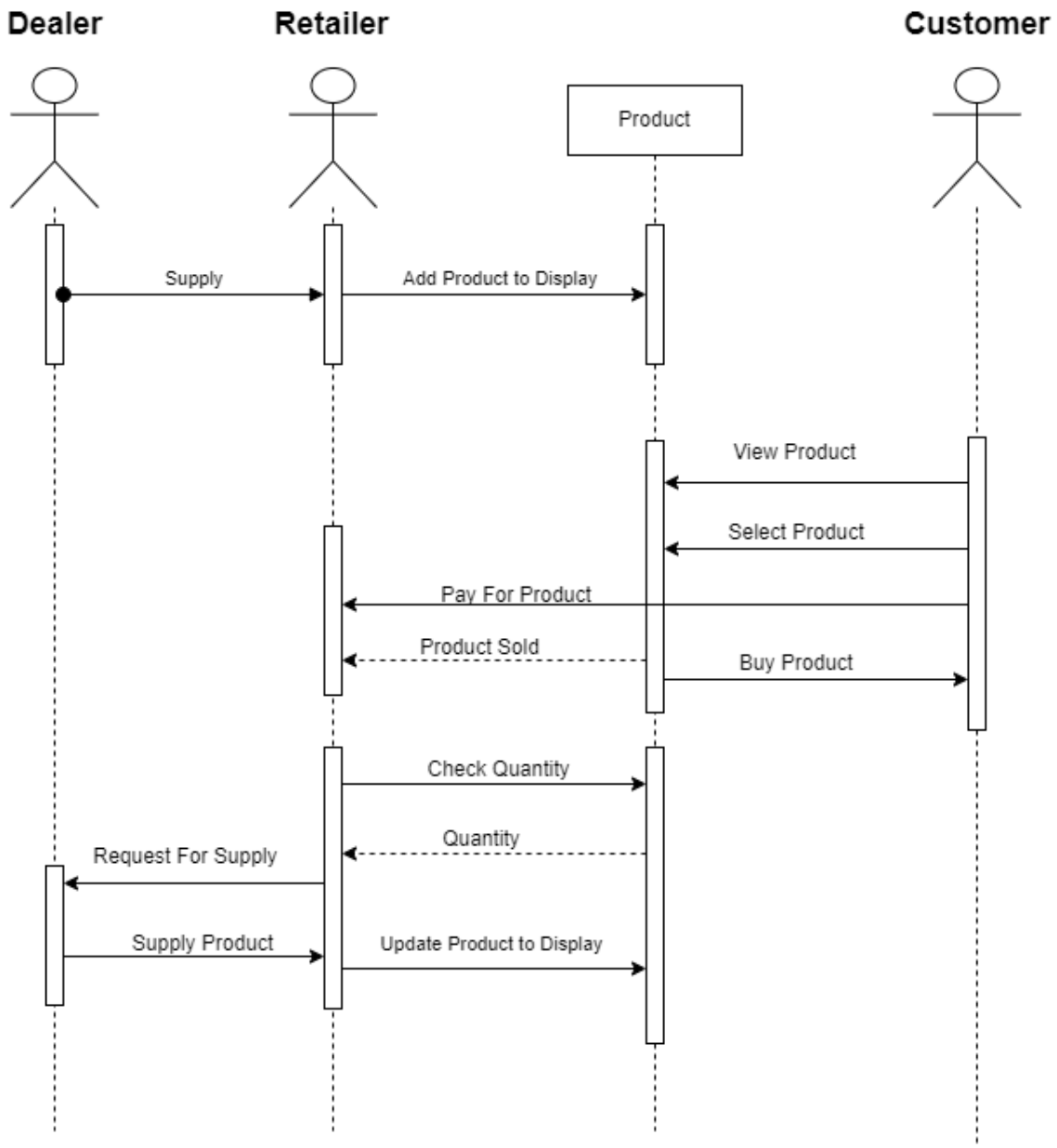
Class Diagrams:



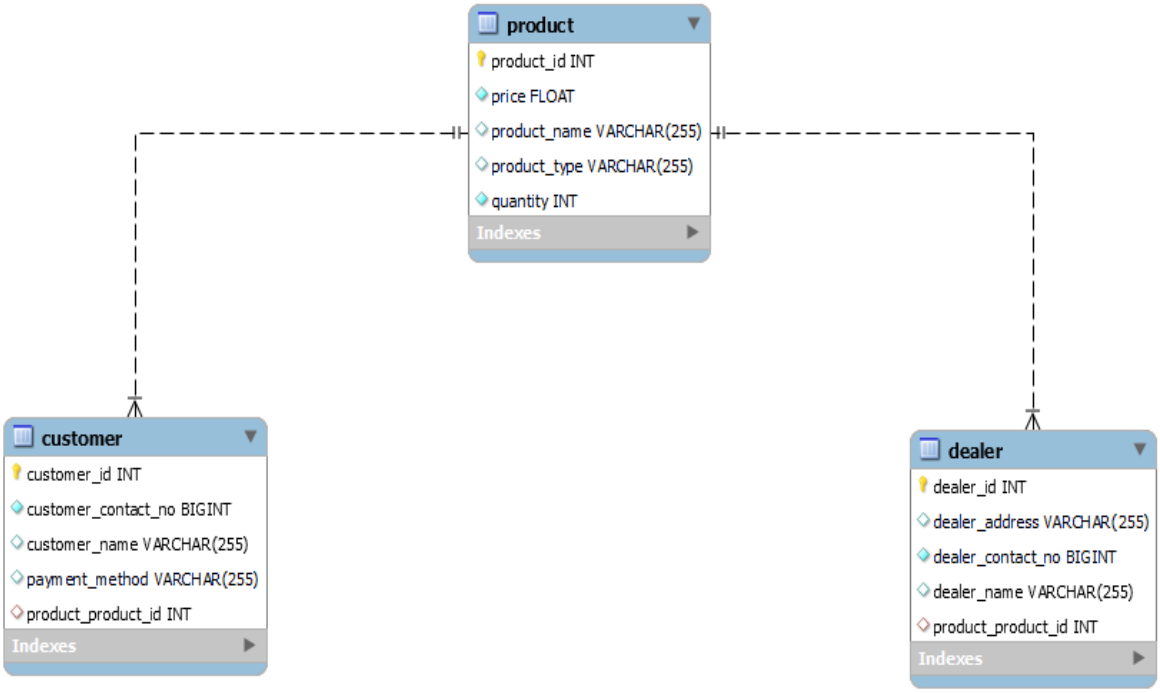
Use Case Diagram:



Sequence Diagram:



Database Design: ER Diagram



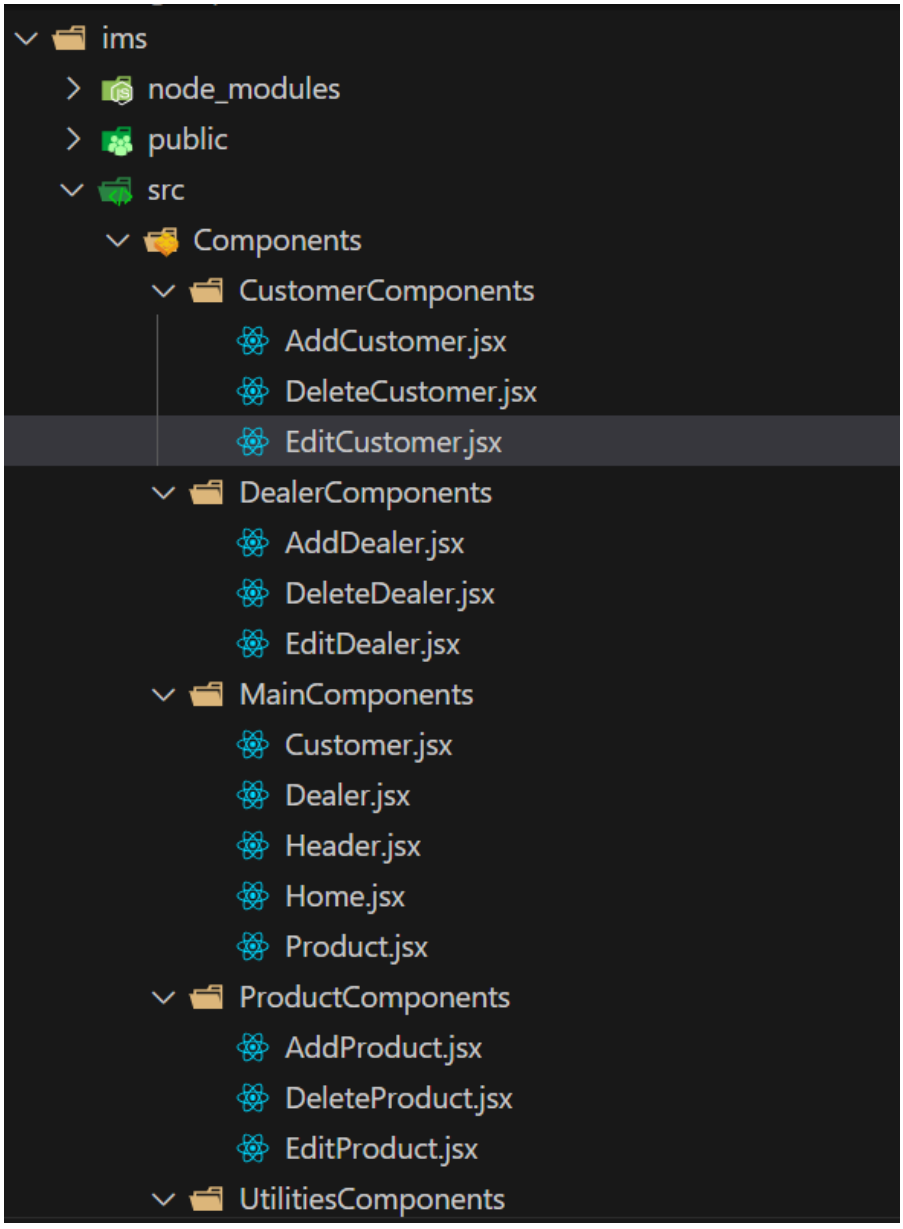
Database Structure:

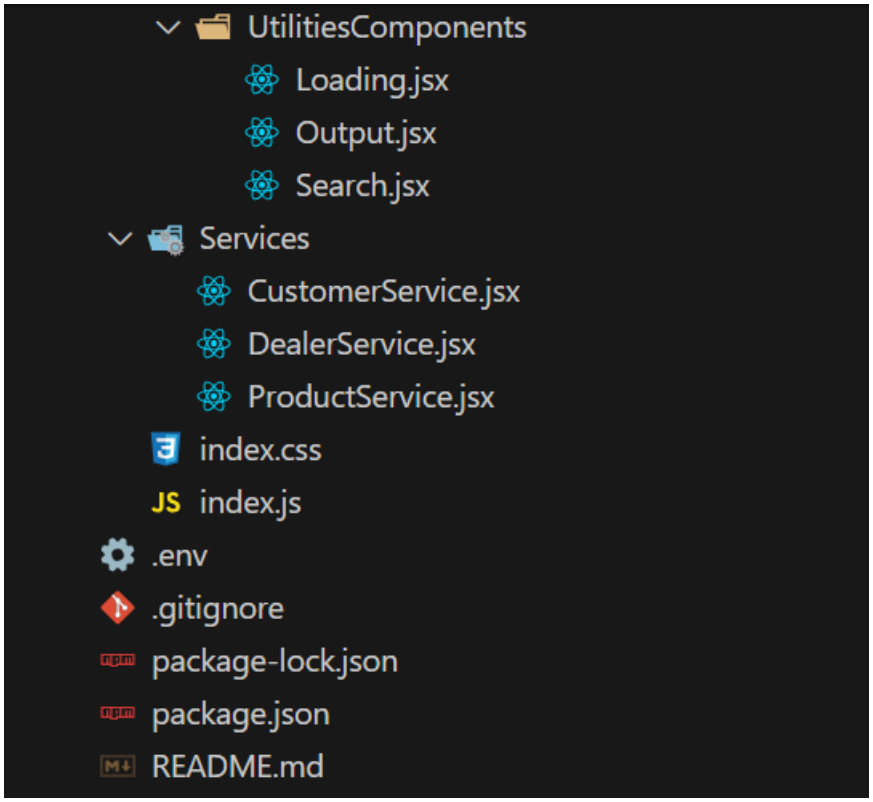
MySQL 8.0 Command Line Client

```
mysql> create database ims_db;
Query OK, 1 row affected (0.01 sec)

mysql> use ims_db;
Database changed
mysql> show tables;
+------------------+
| Tables_in_ims_db |
+------------------+
| customer         |
| dealer           |
| product          |
+------------------+
3 rows in set (0.00 sec)

mysql> desc customer;
+---------------------+--------------+------+-----+---------+----------------+
| Field               | Type         | Null | Key | Default | Extra          |
+---------------------+--------------+------+-----+---------+----------------+
| customer_id         | int          | NO   | PRI | NULL    | auto_increment |
| customer_contact_no | bigint       | NO   |     | NULL    |                |
| customer_name       | varchar(255) | YES  |     | NULL    |                |
| payment_method      | varchar(255) | YES  |     | NULL    |                |
| product_product_id  | int          | YES  | MUL | NULL    |                |
+---------------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> desc dealer;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| dealer_id          | int          | NO   | PRI | NULL    | auto_increment |
| dealer_address     | varchar(255) | YES  |     | NULL    |                |
| dealer_contact_no  | bigint       | NO   |     | NULL    |                |
| dealer_name        | varchar(255) | YES  |     | NULL    |                |
| product_product_id | int          | YES  | MUL | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> desc product;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| product_id   | int          | NO   | PRI | NULL    | auto_increment |
| price        | float        | NO   |     | NULL    |                |
| product_name | varchar(255) | YES  |     | NULL    |                |
| product_type | varchar(255) | YES  |     | NULL    |                |
| quantity     | int          | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

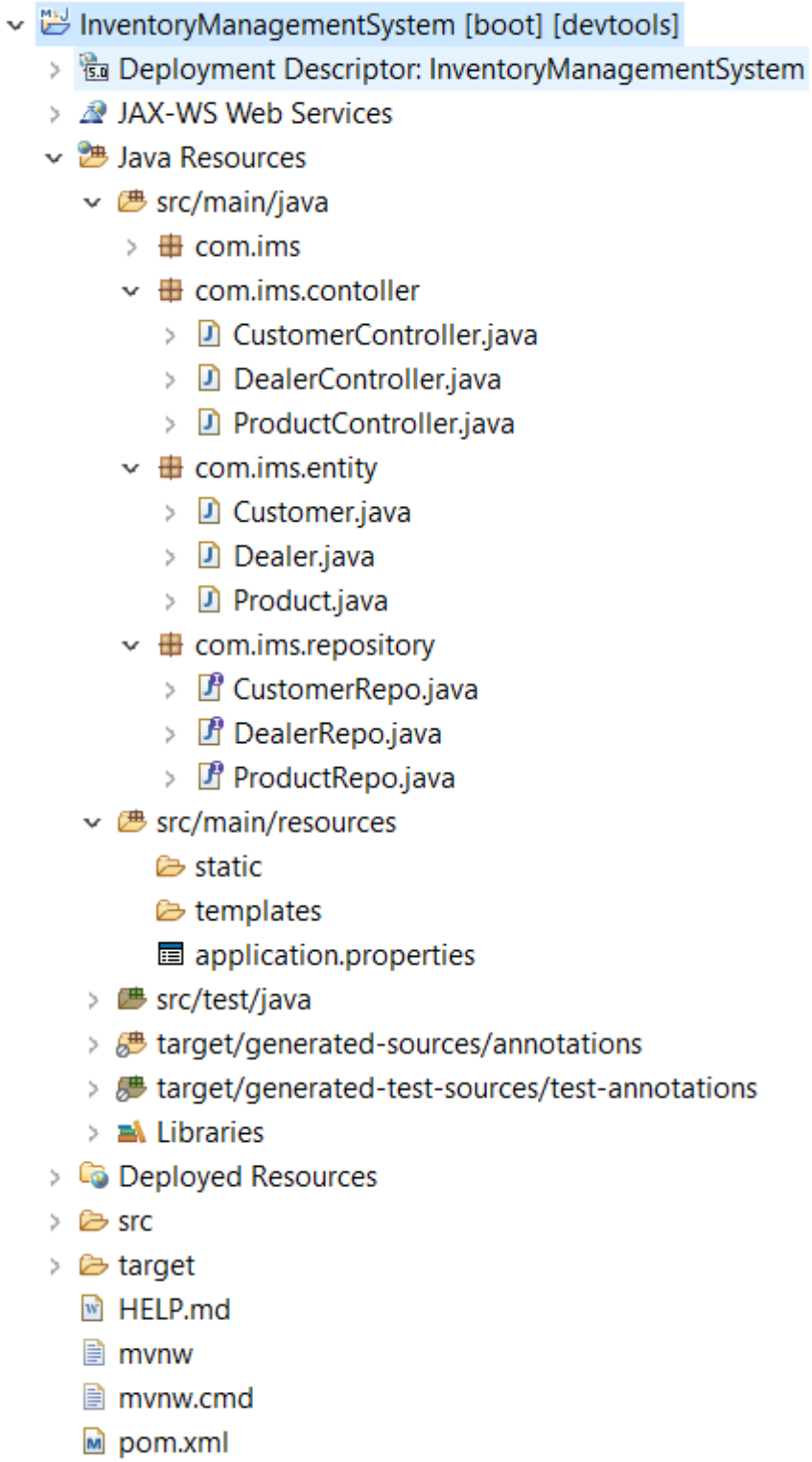
Folder Structure:

Frontend:




UtilitiesComponents
Loading.jsx
Output.jsx
Search.jsx
Services
CustomerService.jsx
DealerService.jsx
ProductService.jsx
index.css
index.js
.env
.gitignore
package-lock.json
package.json
README.md

Backend:


InventoryManagementSystem [boot] [devtools]
Deployment Descriptor: InventoryManagementSystem
JAX-WS Web Services
Java Resources
src/main/java
com.ims
com.ims.contoller
CustomerController.java
DealerController.java
ProductController.java
com.ims.entity
Customer.java
Dealer.java
Product.java
com.ims.repository
CustomerRepo.java
DealerRepo.java
ProductRepo.java
src/main/resources
static
templates
application.properties
src/test/java
target/generated-sources/annotations
target/generated-test-sources/test-annotations
Libraries
Deployed Resources
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml

Coding:

Frontend:

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Header from './Components/MainComponents/Header';
import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Header />
  </React.StrictMode>
);
```

MainComponents:

Header.jsx

```
import React, { useState } from 'react';
import { BrowserRouter, Link, Route, Routes } from "react-router-dom";
import Customer from './Customer';
import Dealer from './Dealer';
import Home from './Home';
import Product from './Product';



const Header = () => {
    const [navbarOpen, setNavbarOpen] = useState(false);
```

```
    return (
        <BrowserRouter>
        <div className='container-fluid sticky-top' id="header">
            <nav className="navbar navbar-expand-lg mt-2">
            <div className="navbar-brand fw-bold text-white h3">Inventory
Management System</div>
                    <button
                        className="navbar-toggler ms-auto bg-light"
                        type="button"
                        data-bs-toggle="collapse"
                        data-bs-target="#navbarNav"
                        aria-controls="navbarNav"
                        aria-expanded="false"
                        aria-label="Toggle navigation"
                        onClick={() => setNavbarOpen(!navbarOpen)}
                    >
                        <i className="bi bi-list"></i>
                    </button>
                    <div
                        className={`collapse navbar-collapse ms-auto
${navbarOpen ? "show" : ""}`}
                    >
                        <ul className="navbar-nav ms-auto">
                            <li className="nav-item me-3">
                                <Link
                                    className="nav-link btn btn-outline-dark
border-0 text-white rounded-pill"
                                    to="/"
                                >
                                <i className="bi bi-house"></i>Home Page
                                </Link>
                            </li>
```

```
                            <li className="nav-item me-3">
                                <Link
                                    className="nav-link btn btn-outline-dark
border-0 text-white rounded-pill"
                                    to="/product"
                                >
                                <i className="bi bi-backpack"></i>Product Data
                                </Link>
                            </li>
                            <li className="nav-item">
                                <Link
                                    className="nav-link btn btn-outline-dark
border-0 text-white rounded-pill"
                                    to="/customer"
                                >
                                <i className="bi bi-file-earmark-bar-
graph"></i>Customer Data
                                </Link>
                            </li>
                            <li className="nav-item">
                                <Link
                                    className="nav-link btn btn-outline-dark
border-0 text-white rounded-pill"
                                    to="/dealer"
                                >
                                <i className="bi bi-file-earmark-bar-
graph"></i>dealer Data
                                </Link>
                            </li>
                        </ul>
                    </div>
            </nav>
            </div>
```

```
            <div className='container-fluid'>
    <Routes>
        <Route  path="/" element={<Home />}/>
        <Route  path="/product" element={<Product />}/>
        <Route   path="/customer" element={<Customer />} />
        <Route   path="/dealer" element={<Dealer />} />
    </Routes>
    </div>
    </BrowserRouter>
    );
}

export default Header;
```

Home.jsx

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import Loading from '../UtilitiesComponents/Loading';

const Home = () => {

    const navigate = useNavigate();

    const[flag,setFlag] = useState();

    const handleClick = (e) =>{
        e.preventDefault();
        setFlag(true);
        setTimeout(()=>{navigate('/product')},2000)
    };

    return (
```

```
        <div id='home' className='container mt-5'>
            <div className='card w-75 shadow mx-auto'>
                <div className='card-title text-center  mt-5 mb-3'>
            <h1>Welcome to the home page!</h1></div>
            <div className='card-body ms-5 h5'>
            <p>This is a simple website created using HTML and CSS. The purpose of this site is to provide an example for beginners</p>
            <p>This is a simple application that showcases some of my skills in web development.  The main features include:</p>
            <p>This is a simple example of a Landing Page site built with React.js.</p>
            <p>This is a Ssimple website for Inventory Management</p>
            <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Facilis nihil sequi praesentium ratione nostrum exercitationem, sapiente odit temporibus, dicta enim mollitia cum vitae maxime unde laborum repudiandae quibusdam libero assumenda!</p>
            </div>
            <div className='card-footer border-0 text-center'>
            <button className="btn btn-primary" onClick={(e)=>handleClick(e)}>Get Started</button>
            </div>
            </div>
            <Loading  flag={flag}/>
        </div>
    );
}

export default Home;
```

Product.jsx

```
import { Button } from '@mui/material';
import React, { useEffect, useState } from 'react';
import productService from '../../Services/ProductService';
```

```
import AddProduct from '../ProductComponents/AddProduct';
import DeleteProduct from '../ProductComponents/DeleteProduct';
import EditProduct from '../ProductComponents/EditProduct';
import Search from '../UtilitiesComponents/Search';

const Product = () => {

    const [data, setData] = useState([]);
    const [eId, setEventId] = useState(0);
    const [search, setSearch] = useState("");

    const [Flags, setFlags] = useState({
        isAddable: false,
        isEditable: false,
        isDeletable: false
    });

    useEffect(() => {
        productService.doReadAll()
            .then((res) => setData(res.data))
            .catch((err) => console.log(err))
    }, [])


    return (
        <div>
            <section className="container container-lg">
                <div className="card card-lg mt-5 shadow-lg p-2 mx-auto">
                    <div className="card-header card-title border-0 text-
center h1">Product Record List</div>
                    <div className="card-body">
                        <div className="container row mt-3">
```

```
                            <div className="col-4" style={{ "textAlign":
"left" }}>
                                <Button variant="contained" onClick={() => {
setFlags({ isAddable: true }) }} color="success">Add product</Button></div>
                            <div className="col-4" style={{ "textAlign":
"center" }}>
                                <h5>No of Record: {data.length} </h5>
                            </div>
                            <div className="col-4">
                                <Search search={search} setSearch={setSearch}
/>
                            </div>
                        </div>
                        <div className='container'>
                            <table className="table table-bordered table-
striped table-hover mt-3">
                                <thead className="bg-dark text-white text-
center">
                                    <tr>
                                        <td>Product Name</td>
                                        <td>Product Type</td>
                                        <td>Quantity</td>
                                        <td>Price</td>
                                        <td className="col-2">Action</td>
                                    </tr>
                                </thead>
                                <tbody>
                                    {
                                        data.filter((st) =>
st.productName.toLowerCase().includes(search) ||
st.productName.toUpperCase().includes(search)).map((item) => (
                                            <tr key={item.productId}>
                                                <td>{item.productName}</td>
                                                <td>{item.productType}</td>
```

```
                                                    <td>{item.quantity}</td>
                                                    <td>Rs: {item.price}/-</td>
                                                    <td className="d-flex justify-
content-evenly"><Button variant="contained" onClick={() => { setFlags({
isEditable: true }); setEventId(item.productId) }}>edit</Button><Button
variant="contained" color="error" onClick={() => { setFlags({ isDeletable:
true }); setEventId(item.productId) }}>delete</Button></td>
                                                </tr>
                                            ))
                                        }
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
                </section>
                {Flags.isAddable && <AddProduct Flags={Flags} />}
                {Flags.isEditable && <EditProduct Flags={Flags} id={eId} />}
                {Flags.isDeletable && <DeleteProduct Flags={Flags} id={eId} />}
            </div>
        );
}

export default Product;
```

Customer.jsx

```
import { Button } from '@mui/material';
import React, { useEffect, useState } from 'react';
import customerService from '../../Services/CustomerService';
import AddCustomer from '../CustomerComponents/AddCustomer';
import DeleteCustomer from '../CustomerComponents/DeleteCustomer';
import EditCustomer from '../CustomerComponents/EditCustomer';
```

```
import Search from '../UtilitiesComponents/Search';


const Customer = () => {


    const [customerData,setCustomerData] = useState([]);
    const [eId,setEventId] = useState(0);
    const [search, setSearch] = useState("");


    const [Flags, setFlags] = useState({
        isAddable: false,
        isEditable: false,
        isDeletable: false
    });


    useEffect(()=>{
        customerService.doReadAll()
        .then((res) => {setCustomerData(res.data)})
        .catch((err)=>console.log(err));
    },[])



    return (
        <div>
            <section className="container container-lg">
                <div className="card card-lg mt-5 shadow-lg p-2 mx-auto">
                    <div className="card-header card-title border-0 text-
center h1">Customer Record List</div>
                    <div className="card-body">
                        <div className="container row mt-3">
                            <div className="col-4" style={{ "textAlign":
"left" }}>
```

```
                                <Button variant="contained" onClick={() => {
setFlags({ isAddable: true }) }} color="success">Add Customer</Button></div>
                                <div className="col-4" style={{ "textAlign":
"center" }}>
                                    <h5>No of Record: {customerData.length} </h5>
                                </div>
                                <div className="col-4">
                                    <Search search={search} setSearch={setSearch}
/>
                                </div>
                            </div>
                            <div className='container'>
                                <table className="table table-bordered table-
striped table-hover mt-3">
                                    <thead className="bg-dark text-white text-
center">
                                        <tr>
                                            <td>Customer Name</td>
                                            <td>Contact Number</td>
                                            <td>Purchased Product</td>
                                            <td>Product Type</td>
                                            <td>Price</td>
                                            <td>Payment method</td>
                                            <td className="col-2">Action</td>
                                        </tr>
                                    </thead>
                                    <tbody>
                                        {

customerData.filter((st)=>st.customerName.toLowerCase().includes(search)||st.c
ustomerName.toUpperCase().includes(search)).map((item)=>(
                                                <tr key={item.customerId}>
                                                    <td>{item.customerName}</td>
```

```
<td>{item.customerContactNo}</td>

<td>{item.product.productName}</td>

<td>{item.product.productType}</td>
                                                    <td>Rs: {item.product.price}/-
</td>
                                                    <td>{item.paymentMethod}</td>
                                                    <td className="d-flex justify-
content-evenly"><Button variant="contained" onClick={() => {
setFlags({isEditable:true});setEventId(item.customerId)}}>edit</Button><Button
variant="contained" color="error" onClick={() => {
setFlags({isDeletable:true});setEventId(item.customerId)
}}>delete</Button></td>
                                                </tr>
                                            ))
                                        }
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
                </section>
                {Flags.isAddable && <AddCustomer Flags={Flags} />}
                {Flags.isEditable && <EditCustomer Flags={Flags} id={eId} />}
                {Flags.isDeletable && <DeleteCustomer Flags={Flags} id={eId} />}
            </div>
        );
}

export default Customer;
```

Dealer.jsx

```
import { Button } from '@mui/material';
import React, { useEffect, useState } from 'react';
import dealerService from '../../Services/DealerService';
import AddDealer from '../DealerComponents/AddDealer';
import DeleteDealer from '../DealerComponents/DeleteDealer';
import EditDealer from '../DealerComponents/EditDealer';
import Search from '../UtilitiesComponents/Search';

const Dealer = () => {

    const [data, setData] = useState([]);
    const [eId, setEventId] = useState(0);
    const [search, setSearch] = useState("");

    const [Flags, setFlags] = useState({
        isAddable: false,
        isEditable: false,
        isDeletable: false
    });

    useEffect(() => {
        dealerService.doReadAll()
            .then((res) => setData(res.data))
            .catch((err) => console.log(err))
    }, [])

    return (
        <div>
            <section className="container container-lg">
                <div className="card card-lg mt-5 shadow-lg p-2 mx-auto">
                    <div className="card-header card-title border-0 text-
center h1">Dealer Record List</div>
```

```
                    <div className="card-body">
                        <div className="container row mt-3">
                            <div className="col-4" style={{ "textAlign":
"left" }}>
                                <Button variant="contained" onClick={() => {
setFlags({ isAddable: true }) }} color="success">Add dealer</Button></div>
                            <div className="col-4" style={{ "textAlign":
"center" }}>
                                <h5>No of Record: {data.length} </h5>
                            </div>
                            <div className="col-4">
                                <Search search={search} setSearch={setSearch}
/>
                            </div>
                        </div>
                        <div className='container'>
                            <table className="table table-bordered table-
striped table-hover mt-3">
                                <thead className="bg-dark text-white text-
center">
                                    <tr>
                                        <td>Dealer Name</td>
                                        <td>Contact Number</td>
                                        <td>Address</td>
                                        <td>Product Name</td>
                                        <td>Product Type</td>
                                        <td>Quantity</td>
                                        <td className="col-2">Action</td>
                                    </tr>
                                </thead>
                                <tbody>
                                    {
```

```
                                        data.filter((st) =>
st.dealerName.toLowerCase().includes(search) ||
st.dealerName.toUpperCase().includes(search)).map((item) => (
                                            <tr key={item.dealerId}>
                                                <td>{item.dealerName}</td>

<td>{item.dealerContactNo}</td>
                                                <td>{item.dealerAddress}</td>

<td>{item.product.productName}</td>

<td>{item.product.productType}</td>

<td>{item.product.quantity}</td>
                                                <td className="d-flex justify-
content-evenly"><Button variant="contained" onClick={() => { setFlags({
isEditable: true }); setEventId(item.dealerId) }}>edit</Button><Button
variant="contained" color="error" onClick={() => { setFlags({ isDeletable:
true }); setEventId(item.dealerId) }}>delete</Button></td>
                                            </tr>
                                        ))
                                    }
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
            </section>
            {Flags.isAddable && <AddDealer Flags={Flags} />}
            {Flags.isEditable && <EditDealer Flags={Flags} id={eId} />}
            {Flags.isDeletable && <DeleteDealer Flags={Flags} id={eId} />}
        </div>
    );
}
```

```
export default Dealer;
```

Product Components:

AddProduct.jsx

```
import { Button, Dialog, DialogActions, DialogContent, DialogTitle, TextField
} from '@mui/material';
import React, { useState } from 'react';
import productService from '../../Services/ProductService';
import Loading from '../UtilitiesComponents/Loading';
import Output from '../UtilitiesComponents/Output';


const AddProduct = (props) => {


    const [flag, setFlag] = useState(props.Flags.isAddable);
    const [msg, setMsg] = useState("");
    const [loadingFlag, setLoadingFlag] = useState(false)
    const [rightFlag, setRightFlag] = useState()
    const [wrongFlag, setWrongFlag] = useState()
    const [data,setData] = useState({
        productName: "",
        productType:"",
        quantity:  0,
        price:0
    });


    const handleInsert = (e) => {
        e.preventDefault()
        setFlag(false)
        setLoadingFlag(true)
        setTimeout(() => {
            setLoadingFlag(false)
```

```
            productService.doCreate(data).then((res) => {
                if (res.data === "success") {
                    setMsg("Successfully added the Product information.")
                    setRightFlag(true)
                }
                else {
                    setMsg("Failed to add the Product Information!")
                    setWrongFlag(true)
                }
            })
        }, 2000)
    }



    return (
        <div>
            <Dialog
                open={flag}
                PaperProps={{
                    component: 'form',
                    onSubmit: (e) => handleInsert(e),
                }}
            >
                <DialogTitle className="text-center h2">Enter details to
Insert</DialogTitle>
                <DialogContent>
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="productName"
                        name="productName"
```

```
                        label="Product Name"
                        autoComplete="off"
                        type="text"
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="productType"
                        name="productType"
                        label="Product Type"
                        autoComplete="off"
                        type="text"
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="quantity"
                        name="quantity"
                        label="Quantity"
                        autoComplete="off"
                        type="number"
                        fullWidth
```

```
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="price"
                        name="price"
                        label="Price"
                        autoComplete="off"
                        type="number"
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
                    <Button type="submit" variant="contained"
color="success">Insert</Button>
                </DialogActions>
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
        </div>
    );
}
```

```
export default AddProduct;
```

EditProduct.jsx

```
import { Button, Dialog, DialogActions, DialogContent, DialogTitle, TextField
} from '@mui/material';

import React, { useEffect, useState } from 'react';

import productService from '../../Services/ProductService';

import Loading from '../UtilitiesComponents/Loading';

import Output from '../UtilitiesComponents/Output';


const EditProduct = (props) => {

    const [flag, setFlag] = useState(props.Flags.isEditable);

    const [msg, setMsg] = useState("");

    const [loadingFlag, setLoadingFlag] = useState(false)

    const [rightFlag, setRightFlag] = useState()

    const [wrongFlag, setWrongFlag] = useState()


    const [data,setData] = useState({

        productId:0,

        productName: "",

        productType:"",

        quantity:  0,

        price:0

    });


    useEffect(()=>{

        productService.doRead(props.id)

        .then((res)=>{setData(res.data)})

        .catch((err)=>console.log(err))

    },[props.id])
```

```
    const  handleEdit= (e)=>{
        e.preventDefault()
        setFlag(false)
        setLoadingFlag(true)
        setTimeout(()=>{
            setLoadingFlag(false)
            productService.doUpdate(data).then((res)=>{
                if (res.data === "success"){
                    setMsg("Successfully Updated the Product information.")
                    setRightFlag(true)
                }
            else {
                setMsg("Failed to Update the Product information!")
                setWrongFlag(true)
            }
            })
        },2000)
    }


    return (
        <div>
            <Dialog
                open={flag}
                PaperProps={{
                    component: 'form',
                    onSubmit: (e) => handleEdit(e),
                }}
            >
                <DialogTitle className="text-center h2">Enter details to
Update</DialogTitle>
                <DialogContent>
```

```
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="productName"
                        name="productName"
                        label="Product Name"
                        autoComplete="off"
                        type="text"
                        fullWidth
                        value={data.productName}
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="productType"
                        name="productType"
                        label="Product Type"
                        autoComplete="off"
                        type="text"
                        fullWidth
                        value={data.productType}
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
```

```
                        autoFocus
                        required
                        id="quantity"
                        name="quantity"
                        label="Quantity"
                        autoComplete="off"
                        type="number"
                        fullWidth
                        value={data.quantity}
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="price"
                        name="price"
                        label="Price"
                        autoComplete="off"
                        type="number"
                        fullWidth
                        value={data.price}
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
```

```
                    <Button type="submit" variant="contained"
color="success">Update</Button>
                </DialogActions>
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
        </div>
    );
}

export default EditProduct;
```

DeleteProduct.jsx

```
import { Alert, Button, Dialog, DialogActions, DialogContent,
DialogContentText, DialogTitle } from '@mui/material';
import React, { useState } from 'react';
import productService from '../../Services/ProductService';
import Loading from '../UtilitiesComponents/Loading';
import Output from '../UtilitiesComponents/Output';

const DeleteProduct = (props) => {
    const [flag, setFlag] = useState(props.Flags.isDeletable);
    const [msg,setMsg] = useState("");
    const [loadingFlag, setLoadingFlag] = useState(false)
    const [rightFlag, setRightFlag] = useState()
    const [wrongFlag, setWrongFlag] = useState()

    const handleDelete = (e) =>{
        e.preventDefault()
        setFlag(false)
        setLoadingFlag(true)
```

```
        setTimeout(()=>{
            setLoadingFlag(false)
            productService.doDelete(props.id).then((res)=>{
                if (res.data === "success"){
                    setMsg("Successfully Deleted the Product information.")
                    setRightFlag(true)
                }
            else {
                setMsg("Failed to Delete the Product information!")
                setWrongFlag(true)
            }
            })
        },2000)
    }
    return (
        <div>
            <Dialog
                open={flag}
                aria-labelledby="alert-dialog-title"
                aria-describedby="alert-dialog-description"
            >
                <DialogTitle id="alert-dialog-title">
                <Alert severity="error"  className='fs-6'>
                        Are you confirm to delete ?
                    </Alert>
                </DialogTitle>
                <DialogContent>
                    <DialogContentText id="alert-dialog-description">
                        Warning: this action cannot be undone. Are you sure to
delete the Product Data?
                        <Alert variant="outlined" severity="warning"
className='mt-2' >
```

```
                            Note: All the Customer & Dealer Record which
belongs to this Product will also be deleted.

                        </Alert>

                    </DialogContentText>

                </DialogContent>

                <DialogActions className="d-flex d-flex justify-content-
between">

                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>

                    <Button onClick={(e) => { handleDelete(e) }}
variant="contained" color="success">Confirm</Button>

                </DialogActions>

            </Dialog>

            {<Loading flag={loadingFlag} />}

            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}

        </div>

    );

}


export default DeleteProduct;
```

Dealer Components:

AddDealer.jsx

```
import { Button, Dialog, DialogActions, DialogContent, DialogTitle,
FormControl, InputLabel, MenuItem, Select, TextField } from '@mui/material';

import React, { useEffect, useState } from 'react';

import dealerService from '../../Services/DealerService';

import productService from '../../Services/ProductService';

import Loading from '../UtilitiesComponents/Loading';

import Output from '../UtilitiesComponents/Output';


const AddDealer = (props) => {
```

```
const [insertData, setInsertData] = useState({
    dealerName: "",
    dealerContactNo: 0,
    dealerAddress:"",
    product: {
        productId: 0,
    }
});

const [flag, setFlag] = useState(props.Flags.isAddable);
const [msg, setMsg] = useState("");
const [loadingFlag, setLoadingFlag] = useState(false)
const [rightFlag, setRightFlag] = useState()
const [wrongFlag, setWrongFlag] = useState()
const [productData, setProductData] = useState([]);

useEffect(() => {
    productService.doReadAll()
        .then((res) => setProductData(res.data))
        .catch((err) => console.log(err))
}, [])

const handleInsert = (e) =>{
    e.preventDefault();
    setFlag(false)
    setLoadingFlag(true)
    setTimeout(() => {
        setLoadingFlag(false)
        dealerService.doCreate(insertData).then((res) => {
            if (res.data === "success") {
```

```
                    setMsg("Successfully added the Dealer Information.")
                    setRightFlag(true)
                }
                else {
                    setMsg("Failed to add the Dealer Information!")
                    setWrongFlag(true)
                }
            })
        }, 2000)
    }

    return (
        <div>
            <Dialog
                open={flag}
                PaperProps={{
                    component: 'form',
                    onSubmit: (e) => handleInsert(e),
                }}
            >
                <DialogTitle className="text-center h2">Enter details to
Insert</DialogTitle>
                <DialogContent>
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="dealerName"
                        name="dealerName"
                        label="Dealer Name"
                        autoComplete="off"
```

```
                        type="text"
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setInsertData({ ...insertData,
[e.target.name]: e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="dealerContactNo"
                        name="dealerContactNo"
                        label="Contact Number"
                        autoComplete="off"
                        type="number"
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setInsertData({ ...insertData,
[e.target.name]: e.target.value }) }}
                    />

                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="dealerAddress"
                        name="dealerAddress"
                        label="Address"
                        autoComplete="off"
                        type="text"
```

```
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setInsertData({ ...insertData,
[e.target.name]: e.target.value }) }}
                    />


                    <FormControl fullWidth className='mt-3'>
                        <InputLabel id="product-label">Product
Name</InputLabel>
                        <Select
                            labelId="product-label"
                            id='productId'
                            name='productId'
                            label='Choose Product'
                            required
                            onChange={(e) => { setInsertData({ ...insertData,
product: { [e.target.name]: e.target.value } }) }}
                        >
                            {productData.map((item) => (<MenuItem
key={item.productId} value={item.productId}>{item.productName}</MenuItem>))}
                        </Select>
                    </FormControl>
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
                    <Button type="submit" variant="contained"
color="success">Insert</Button>
                </DialogActions>
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
```

```
        </div>
    );
}

export default AddDealer;
```

EditDealer.jsx

```
import { Button, Dialog, DialogActions, DialogContent, DialogTitle,
FormControl, InputLabel, MenuItem, Select, TextField } from '@mui/material';
import React, { useEffect, useState } from 'react';
import dealerService from '../../Services/DealerService';
import productService from '../../Services/ProductService';
import Loading from '../UtilitiesComponents/Loading';
import Output from '../UtilitiesComponents/Output';

const EditDealer = (props) => {

    const [flag, setFlag] = useState(props.Flags.isEditable);
    const [msg, setMsg] = useState("");
    const [loadingFlag, setLoadingFlag] = useState(false)
    const [rightFlag, setRightFlag] = useState()
    const [wrongFlag, setWrongFlag] = useState()
    const [productData, setProductData] = useState([]);

    const [data, setData] = useState({
        dealerId:0,
        dealerName: "",
        dealerContactNo: 0,
        dealerAddress:"",
        product: {
```

```
        productId: 0,
    }
});

useEffect(() => {
    productService.doReadAll()
        .then((res) => setProductData(res.data))
        .catch((err) => console.log(err))
}, [])



useEffect(()=>{
    dealerService.doRead(props.id)
    .then((res)=>setData(res.data))
    .catch((err)=>{console.error(err)})
},[props.id])

const handleEdit = (e) => {
    e.preventDefault()
    setFlag(false)
    setLoadingFlag(true)
    setTimeout(()=>{
        setLoadingFlag(false)
        dealerService.doUpdate(data).then((res)=>{
            if (res.data === "success"){
                setMsg("Successfully Updated the Dealer information.")
                setRightFlag(true)
            }
        else {
            setMsg("Failed to Update the Dealer!")
            setWrongFlag(true)
```

```
            }
            })
        },2000)
    }

    return (
        <div>
            <Dialog
                open={flag}
                PaperProps={{
                    component: 'form',
                    onSubmit: (e) => handleEdit(e),
                }}
            >
                <DialogTitle className="text-center h2">Enter details to
Update</DialogTitle>
                <DialogContent>
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="dealerName"
                        name="dealerName"
                        label="Dealer Name"
                        autoComplete="off"
                        type="text"
                        value={data.dealerName}
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
```

```
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="dealerContactNo"
                        name="dealerContactNo"
                        label="Contact Number"
                        autoComplete="off"
                        type="number"
                        value={data.dealerContactNo}
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />

                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="dealerAddress"
                        name="dealerAddress"
                        label="Address"
                        autoComplete="off"
                        type="text"
                        fullWidth
                        variant="standard"
                        value={data.dealerAddress}
```

```
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />


                    <FormControl fullWidth className='mt-3'>
                        <InputLabel id="product-label">Product
Name</InputLabel>
                        <Select
                            labelId="product-label"
                            id='productId'
                            name='productId'
                            label='Choose Product'
                            required
                            value={data.product.productId}
                            onChange={(e) => { setData({ ...data,
product:{[e.target.name]: e.target.value }}) }}
                        >
                            {productData.map((item) => (<MenuItem
key={item.productId} value={item.productId}>{item.productName}</MenuItem>))}
                        </Select>
                    </FormControl>
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
                    <Button type="submit" variant="contained"
color="success">Update</Button>
                </DialogActions>
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
        </div>
```

```
    );
}

export default EditDealer;
```

DeleteDealer.jsx

```
import { Alert, Button, Dialog, DialogActions, DialogContent,
DialogContentText, DialogTitle } from '@mui/material';
import React, { useState } from 'react';
import dealerService from '../../Services/DealerService';
import Loading from '../UtilitiesComponents/Loading';
import Output from '../UtilitiesComponents/Output';

const DeleteDealer = (props) => {
    const [flag, setFlag] = useState(props.Flags.isDeletable);
    const [msg,setMsg] = useState("");
    const [loadingFlag, setLoadingFlag] = useState(false)
    const [rightFlag, setRightFlag] = useState()
    const [wrongFlag, setWrongFlag] = useState()

    const handleDelete = (e) =>{
        e.preventDefault()
        setFlag(false)
        setLoadingFlag(true)
        setTimeout(()=>{
            setLoadingFlag(false)
            dealerService.doDelete(props.id).then((res)=>{
                if (res.data === "success"){
                    setMsg("Successfully Deleted the Dealer information.")
                    setRightFlag(true)
                }
```

```
            else {
                setMsg("Failed to Delete the Dealer Information!")
                setWrongFlag(true)
            }
            })
        },2000)
    }

    return (
        <div>
            <Dialog
                open={flag}
                aria-labelledby="alert-dialog-title"
                aria-describedby="alert-dialog-description"
            >
                <DialogTitle id="alert-dialog-title">
                    <Alert severity="error">
                        Are you confirm to delete ?
                    </Alert>
                </DialogTitle>
                <DialogContent>
                    <DialogContentText id="alert-dialog-description">
                        Warning: this action cannot be undone. Are you sure to
delete the Dealer Data?
                    </DialogContentText>
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
                    <Button onClick={(e) => { handleDelete(e) }}
variant="contained" color="success">Confirm</Button>
                </DialogActions>
```

```
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
        </div>
    );
}

export default DeleteDealer;
```

CustomerComponents:

AddCustomer.jsx

```
import { Button, Dialog, DialogActions, DialogContent, DialogTitle,
FormControl, InputLabel, MenuItem, Select, TextField } from '@mui/material';
import React, { useEffect, useState } from 'react';
import customerService from '../../Services/CustomerService';
import productService from '../../Services/ProductService';
import Loading from '../UtilitiesComponents/Loading';
import Output from '../UtilitiesComponents/Output';



const AddCustomer = (props) => {

    useEffect(() => {
        productService.doReadAll()
            .then((res) =>{setProductData(res.data)})
            .catch((err) => console.log(err))
    }, [])

    const [insertData, setInsertData] = useState({
        customerName: "",
```

```
    customerContactNo: 0,
    paymentMethod: "",
    product: {
        productId: 0,
    }
});

const [flag, setFlag] = useState(props.Flags.isAddable);
const [msg, setMsg] = useState("");
const [loadingFlag, setLoadingFlag] = useState(false)
const [rightFlag, setRightFlag] = useState()
const [wrongFlag, setWrongFlag] = useState()
const [productData, setProductData] = useState([]);

const handleInsert = (e) =>{
    e.preventDefault();
    setFlag(false)
    setLoadingFlag(true)
    setTimeout(() => {
        setLoadingFlag(false)
        customerService.doCreate(insertData).then((res) => {
            if (res.data === "success") {
                setMsg("Successfully added the Customer Information")
                setRightFlag(true)
            }
            else {
                setMsg("Failed to add the Customer Information!")
                setWrongFlag(true)
            }
        })
    }, 2000)
```

```
    }



    return (
        <div>
            <Dialog
                open={flag}
                PaperProps={{
                    component: 'form',
                    onSubmit: (e) => handleInsert(e),
                }}
            >
                <DialogTitle className="text-center h2">Enter details to
Insert</DialogTitle>
                <DialogContent>
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        id="customerName"
                        name="customerName"
                        label="Customer Name"
                        autoComplete='off'
                        type="text"
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setInsertData({ ...insertData,
[e.target.name]: e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
```

```
                        required
                        autoComplete='off'
                        id="customerContactNo"
                        name="customerContactNo"
                        label="Contact Number"
                        type="number"
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setInsertData({ ...insertData,
[e.target.name]: e.target.value }) }}
                    />
                    <FormControl fullWidth className="mt-3">
                        <InputLabel id="payment-label">Payment
Method</InputLabel>
                        <Select
                            labelId="payment-label"
                            id='paymentMethod'
                            name='paymentMethod'
                            label="Choose Payment Method"
                            required
                            onChange={(e) => { setInsertData({ ...insertData,
[e.target.name]: e.target.value }) }}
                        >
                            <MenuItem value="Cash">Cash</MenuItem>
                            <MenuItem value="GPay">GPay</MenuItem>
                            <MenuItem value="PayTM">PayTM</MenuItem>
                            <MenuItem value="AmazonPay">AmazonPay</MenuItem>
                            <MenuItem value="NetBanking">NetBanking</MenuItem>
                        </Select>
                    </FormControl>
                    <FormControl fullWidth className='mt-3'>
                        <InputLabel id="product-label">Product
Name</InputLabel>
```

```
                        <Select
                            labelId="product-label"
                            id='productId'
                            name='productId'
                            label='Choose Product'
                            required
                            onChange={(e) => { setInsertData({ ...insertData,
product: { [e.target.name]: e.target.value } }) }}
                        >
                            {productData.map((item) => (<MenuItem
key={item.productId} value={item.productId}>{item.productName}</MenuItem>))}
                        </Select>
                    </FormControl>
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
                    <Button type="submit" variant="contained"
color="success">Insert</Button>
                </DialogActions>
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
        </div>
    );
}

export default AddCustomer;
```

EditCustomer.jsx

```
import { Button, Dialog, DialogActions, DialogContent, DialogTitle,
FormControl, InputLabel, MenuItem, Select, TextField } from '@mui/material';
import React, { useEffect, useState } from 'react';
import customerService from '../../Services/CustomerService';
import productService from '../../Services/ProductService';
import Loading from '../UtilitiesComponents/Loading';
import Output from '../UtilitiesComponents/Output';

const EditCustomer = (props) => {

    const [flag, setFlag] = useState(props.Flags.isEditable);
    const [msg, setMsg] = useState("");
    const [loadingFlag, setLoadingFlag] = useState(false)
    const [rightFlag, setRightFlag] = useState()
    const [wrongFlag, setWrongFlag] = useState()
    const [productData, setProductData] = useState([]);

    const [data, setData] = useState({
        customerId:0,
        customerName: "",
        customerContactNo: 0,
        paymentMethod: "",
        product: {
            productId: 0,
        }
    });

    useEffect(() => {
        productService.doReadAll()
            .then((res) => setProductData(res.data))
            .catch((err) => console.log(err))
    }, [])
```

```
useEffect(()=>{
    customerService.doRead(props.id)
    .then((res)=>setData(res.data))
    .catch((err)=>{console.error(err)})
},[props.id])

const handleEdit = (e) => {
    e.preventDefault()
    setFlag(false)
    setLoadingFlag(true)
    setTimeout(()=>{
        setLoadingFlag(false)
        customerService.doUpdate(data).then((res)=>{
            if (res.data === "success"){
                setMsg("Successfully Updated the Customer information.")
                setRightFlag(true)
            }
        else {
            setMsg("Failed to Update the Customer Information!")
            setWrongFlag(true)
        }
        })
    },2000)
}

return (
    <div>
        <Dialog
            open={flag}
```

```
                PaperProps={{
                    component: 'form',
                    onSubmit: (e) => handleEdit(e),
                }}
            >
                <DialogTitle className="text-center h2">Enter details to Update</DialogTitle>
                <DialogContent>
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="customerName"
                        name="customerName"
                        label="Customer Name"
                        autoComplete="off"
                        type="text"
                        value={data.customerName}
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]: e.target.value }) }}
                    />
                    <TextField
                        className="mt-3"
                        autoFocus
                        required
                        margin="dense"
                        id="customerContactNo"
                        name="customerContactNo"
                        label="Contact Number"
```

```
                        autoComplete="off"
                        type="number"
                        value={data.customerContactNo}
                        fullWidth
                        variant="standard"
                        onChange={(e) => { setData({ ...data, [e.target.name]:
e.target.value }) }}
                    />
                    <FormControl fullWidth className="mt-3">
                        <InputLabel id="payment-label">Payment
Method</InputLabel>
                        <Select
                            labelId="payment-label"
                            id='paymentMethod'
                            name='paymentMethod'
                            label="Choose Payment Method"
                            value={data.paymentMethod}
                            required
                            onChange={(e) => { setData({ ...data,
[e.target.name]: e.target.value }) }}
                        >
                            <MenuItem value="Cash">Cash</MenuItem>
                            <MenuItem value="GPay">GPay</MenuItem>
                            <MenuItem value="PayTM">PayTM</MenuItem>
                            <MenuItem value="AmazonPay">AmazonPay</MenuItem>
                            <MenuItem value="NetBanking">NetBanking</MenuItem>
                        </Select>
                    </FormControl>
                    <FormControl fullWidth className='mt-3'>
                        <InputLabel id="product-label">Product
Name</InputLabel>
                        <Select
                            labelId="product-label"
```

```
                            id='productName'
                            name='productName'
                            label='Choose Product'
                            required
                            value={data.product.productId}
                            onChange={(e) => { setData({ ...data, product: {
[e.target.name]: e.target.value } }) }}
                        >
                            {productData.map((item) => (<MenuItem
key={item.productId} value={item.productId}>{item.productName}</MenuItem>))}
                        </Select>
                    </FormControl>
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
                    <Button type="submit" variant="contained"
color="success">Update</Button>
                </DialogActions>
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
        </div>
    );
}

export default EditCustomer;
```

DeleteCustomer.jsx

```
import { Alert, Button, Dialog, DialogActions, DialogContent,
DialogContentText, DialogTitle } from '@mui/material';
```

```
import React, { useState } from 'react';
import customerService from '../../Services/CustomerService';
import Loading from '../UtilitiesComponents/Loading';
import Output from '../UtilitiesComponents/Output';

const DeleteCustomer = (props) => {

    const [flag, setFlag] = useState(props.Flags.isDeletable);
    const [msg,setMsg] = useState("");
    const [loadingFlag, setLoadingFlag] = useState(false)
    const [rightFlag, setRightFlag] = useState()
    const [wrongFlag, setWrongFlag] = useState()

    const handleDelete = (e) =>{
        e.preventDefault()
        setFlag(false)
        setLoadingFlag(true)
        setTimeout(()=>{
            setLoadingFlag(false)
            customerService.doDelete(props.id).then((res)=>{
                if (res.data === "success"){
                    setMsg("Successfully Deleted the Customer Information.")
                    setRightFlag(true)
                }
            else {
                setMsg("Failed to Delete the Customer Information!")
                setWrongFlag(true)
            }
            })
        },2000)
    }
```

```
    return (
        <div>
            <Dialog
                open={flag}
                aria-labelledby="alert-dialog-title"
                aria-describedby="alert-dialog-description"
            >
                <DialogTitle id="alert-dialog-title">
                    <Alert severity="error">
                        Are you confirm to delete ?
                    </Alert>
                </DialogTitle>
                <DialogContent>
                    <DialogContentText id="alert-dialog-description">
                        Warning: this action cannot be undone. Are you sure to
delete the Customer Data?
                    </DialogContentText>
                </DialogContent>
                <DialogActions className="d-flex d-flex justify-content-
between">
                    <Button onClick={() => { window.location.reload() }}
variant="contained" >Cancel</Button>
                    <Button onClick={(e) => { handleDelete(e) }}
variant="contained" color="success">Confirm</Button>
                </DialogActions>
            </Dialog>
            {<Loading flag={loadingFlag} />}
            {(rightFlag || wrongFlag) && <Output success={rightFlag}
fail={wrongFlag} msg={msg} />}
        </div>
    );
}
```

```
export default DeleteCustomer;
```

UtilitiesComponents

Loading.jsx

```
import { Backdrop } from "@mui/material";
import React from 'react';
const Loading = (props) => {
    return (
        <Backdrop
        sx={{ color: '#fff', zIndex: (theme) => theme.zIndex.drawer + 1 }}
        open={props.flag}
    >
        <div className="center">
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
            <div className="wave"></div>
        </div>
        </Backdrop>
    );
}

export default Loading;
```

Output.jsx

```
import { Alert, Snackbar } from "@mui/material";

import React from 'react';


const Output = (props) => {

    return (

        <div>

            <Snackbar open={props.success} autoHideDuration={6000}
anchorOrigin={{ vertical: "bottom", horizontal: "left" }} onClose={() => {
window.location.reload(); }}>

                <Alert

                    onClose={() => { window.location.reload(); }}

                    severity="success"

                    variant="filled"

                    sx={{ width: '100%' }}

                >

                    {props.msg}

                </Alert>

            </Snackbar>

            <Snackbar open={props.fail} autoHideDuration={6000}
anchorOrigin={{ vertical: "bottom", horizontal: "left" }} style={{ marginTop:
"115px" }} onClose={() => { window.location.reload(); }}>

                <Alert

                    onClose={() => { window.location.reload(); }}

                    severity="error"

                    variant="filled"

                    sx={{ width: '100%' }}

                >

                    {props.msg}

                </Alert>

            </Snackbar>

        </div>
```

```
    );
}

export default Output;
```

Search.jsx

```
import { TextField } from '@mui/material';
import React from 'react';

const Search = (props) => {
    return (
            <TextField
            name="search"
            size='small'
            autoComplete='off'
            placeholder='Search...'
            value={props.search}
            onChange={(e)=>props.setSearch(e.target.value)}
            fullWidth
            variant="outlined" />
    );
}

export default Search;
```

ServicesComponents:

ProductService.jsx

```
import axios from 'axios';
import { Component } from 'react';

const origin = "http://localhost:1260/";
const createProduct = "createproduct";
```

```
const readAllProduct = "readallproduct";
const updateProduct = "updateproduct";
const deleteProduct = "deleteproduct?id=";
const readProduct = "readproduct?id=";

class ProductService extends Component {

    doCreate = (data) => {
        return axios.post(origin + createProduct, data);
    };

    doReadAll = () => {
        return axios.get(origin + readAllProduct);
    };

    doRead = (id) => {
        return axios.get(origin + readProduct + id);
    };

    doUpdate = (data) =>{
        return axios.put(origin+updateProduct, data);
    };

    doDelete = (id) => {
        return axios.delete(origin + deleteProduct + id)
    };

}
const  productService = new ProductService();
export default productService;
```

DealerService.jsx

```
import axios from 'axios';
import { Component } from 'react';

const origin = "http://localhost:1260/";
const createDealer = "createdealer";
const readAllDealer = "readalldealer";
const updateDealer = "updatedealer";
const deleteDealer = "deletedealer?id=";
const readDealer = "readdealer?id=";
class DealerService extends Component {

    doCreate = (data) => {
        return axios.post(origin + createDealer, data);
    };

    doReadAll = () => {
        return axios.get(origin + readAllDealer);
    };

    doRead = (id) => {
        return axios.get(origin + readDealer + id);
    };

    doUpdate = (data) => {
        return axios.put(origin + updateDealer, data);
    };

    doDelete = (id) => {
        return axios.delete(origin + deleteDealer + id)
    };
```

```
}
const  dealerService = new DealerService();
export  default dealerService;
```

CustomerServices.jsx

```
import axios from 'axios';
import { Component } from 'react';

const origin = "http://localhost:1260/";
const createCustomer = "createcustomer";
const readAllCustomer = "readallcustomer";
const updateCustomer = "updatecustomer";
const deleteCustomer = "deletecustomer?id=";
const readCustomer = "readcustomer?id=";

class CustomerService extends Component {

    doCreate = (data) => {
        return axios.post(origin + createCustomer, data);
    };

    doReadAll = () => {
        return axios.get(origin + readAllCustomer);
    };

    doRead = (id) => {
        return axios.get(origin + readCustomer + id);
    };
```

```
    doUpdate = (data) => {
        return axios.put(origin + updateCustomer, data);
    };

    doDelete = (id) => {
        return axios.delete(origin + deleteCustomer + id)
    };
}
const customerService = new CustomerService();
export default customerService;
```

Index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}

#home{
  box-sizing: inherit;
  background-color: rgb(250, 252, 254);
  box-shadow: rgb(0, 0, 0,0.9);
```

```
}

#header{
  background-color: blueviolet;
}

.center {
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}
.wave {
  width: 5px;
  height: 100px;
  background: linear-gradient(45deg, cyan, #fff);
  margin: 10px;
  animation: wave 1s linear infinite;
  border-radius: 20px;
}
.wave:nth-child(2) {
  animation-delay: 0.1s;
}
.wave:nth-child(3) {
  animation-delay: 0.2s;
}
.wave:nth-child(4) {
  animation-delay: 0.3s;
}
.wave:nth-child(5) {
  animation-delay: 0.4s;
```

```
}
.wave:nth-child(6) {
  animation-delay: 0.5s;
}
.wave:nth-child(7) {
  animation-delay: 0.6s;
}
.wave:nth-child(8) {
  animation-delay: 0.7s;
}
.wave:nth-child(9) {
  animation-delay: 0.8s;
}
.wave:nth-child(10) {
  animation-delay: 0.9s;
}

@keyframes wave {
  0% {
    transform: scale(0);
  }
  50% {
    transform: scale(1);
  }
  100% {
    transform: scale(0);
  }
}
```

Backend:

Com.ims.entity:

Customer.java

```
package com.ims.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;

@Entity
@Table
public class Customer {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)

private int customerId;
private String customerName;
private long customerContactNo;
private String paymentMethod;

@ManyToOne
private Product product;

public Customer() {
super();
// TODO Auto-generated constructor stub
}



public Customer(int customerId, String customerName, long
customerContactNo, String paymentMethod,
Product product) {
super();
this.customerId = customerId;
```

```
this.customerName = customerName;
this.customerContactNo = customerContactNo;
this.paymentMethod = paymentMethod;
this.product = product;
}


public int getCustomerId() {
return customerId;
}

public void setCustomerId(int customerId) {
this.customerId = customerId;
}

public String getCustomerName() {
return customerName;
}

public void setCustomerName(String customerName) {
this.customerName = customerName;
}


public long getCustomerContactNo() {
return customerContactNo;
}


public void setCustomerContactNo(long customerContactNo) {
this.customerContactNo = customerContactNo;
}


public String getPaymentMethod() {
return paymentMethod;
}
```

```
public void setPaymentMethod(String paymentMethod) {
this.paymentMethod = paymentMethod;
}

public Product getProduct() {
return product;
}

public void setProduct(Product product) {
this.product = product;
}

}
```

Product.java

```
package com.ims.entity;

import java.util.List;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;

@Entity
@Table
public class Product {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int productId;
private String productName;
private String productType;
private float price;
private int quantity;
```

```java
@OneToMany(mappedBy = "product",cascade = CascadeType.ALL)
private List<Customer> customer;

@OneToMany(mappedBy = "product",cascade = CascadeType.ALL)
private List<Dealer> dealer;

public Product() {
super();
}
public Product(int productId, String productName, String
productType, float price, int quantity) {
super();
this.productId = productId;
this.productName = productName;
this.productType = productType;
this.price = price;
this.quantity = quantity;
}
public int getProductId() {
return productId;
}
public void setProductId(int productId) {
this.productId = productId;
}
public String getProductName() {
return productName;
}
public void setProductName(String productName) {
this.productName = productName;
}
public String getProductType() {
return productType;
}
public void setProductType(String productType) {
this.productType = productType;
}
public float getPrice() {
return price;
}
```

```
public void setPrice(float price) {
this.price = price;
}
public int getQuantity() {
return quantity;
}
public void setQuantity(int quantity) {
this.quantity = quantity;
}

}
```

Dealer.java

```
package com.ims.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;

@Entity
@Table
public class Dealer {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int dealerId;
private String dealerName;
private String dealerAddress;
private long dealerContactNo;

@ManyToOne
private Product product;

public Dealer() {
super();
// TODO Auto-generated constructor stub
```

```
}

public Dealer(int dealerId, String dealerName, String
dealerAddress, long dealerContactNo, Product product) {
super();
this.dealerId = dealerId;
this.dealerName = dealerName;
this.dealerAddress = dealerAddress;
this.dealerContactNo = dealerContactNo;
this.product = product;
}

public int getDealerId() {
return dealerId;
}

public void setDealerId(int dealerId) {
this.dealerId = dealerId;
}

public String getDealerName() {
return dealerName;
}

public void setDealerName(String dealerName) {
this.dealerName = dealerName;
}

public String getDealerAddress() {
return dealerAddress;
}

public void setDealerAddress(String dealerAddress) {
this.dealerAddress = dealerAddress;
}

public long getDealerContactNo() {
return dealerContactNo;
}

public void setDealerContactNo(long dealerContactNo) {
```

```
this.dealerContactNo = dealerContactNo;
}

public Product getProduct() {
return product;
}

public void setProduct(Product product) {
this.product = product;
}

}
```

Com.ims.repository

CustomerRepo.java

```
package com.ims.repository;

import java.util.List;

import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.ims.entity.Customer;

public interface CustomerRepo extends
JpaRepository<Customer, Integer> {

@Query("from Customer order by customerName")
public List<Customer> getAllCustomer();

}
```

DealerRepo.java

```
package com.ims.repository;

import java.util.List;
```

```
import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.ims.entity.Dealer;

public interface DealerRepo extends JpaRepository<Dealer,
Integer> {

@Query("from Dealer order by dealerName")
public List<Dealer> getAllDealer();

}
```

ProductRepo.java

```
package com.ims.repository;

import java.util.List;

import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.ims.entity.Product;

public interface ProductRepo extends
JpaRepository<Product, Integer> {

@Query("from Product order by productName")
public List<Product> getAllProduct();

}
```

Com.ims.controller

CustomerController.java

```
package com.ims.contoller;

import java.util.List;
```

```java
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.CrossOrigin;
import
org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import
org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

import com.ims.entity.Customer;
import com.ims.repository.CustomerRepo;

@RestController
@CrossOrigin("http://localhost:3030")
public class CustomerController {

@Autowired
CustomerRepo repo;

@PostMapping("/createcustomer")
public String createCustomer(@RequestBody Customer c) {
String msg = "";

try {
repo.saveAndFlush(c);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}
```

```
return msg;
}

@GetMapping("/readallcustomer")
public List<Customer> readAllCustomer() {
return repo.getAllCustomer();
}

@GetMapping("/readcustomer")
public Customer readCustomer(@RequestParam("id")int id) {
return repo.findById(id).get();
}

@PutMapping("/updatecustomer")
public String updateCustomer(@RequestBody Customer c) {

String msg = "";

try {
repo.saveAndFlush(c);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}

return msg;
}

@DeleteMapping("/deletecustomer")
public String deleteCustomer(@RequestParam("id")int id) {
String msg = "";

try {
repo.deleteById(id);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}
```

```
return msg;
}

}
```

DealerController.java

```
package com.ims.contoller;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.CrossOrigin;
import
org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import
org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

import com.ims.entity.Dealer;
import com.ims.repository.DealerRepo;

@RestController
@CrossOrigin("http://localhost:3030")
public class DealerController {

@Autowired
DealerRepo repo;

@PostMapping("/createdealer")
public String createDealer(@RequestBody Dealer d) {
```

```
String msg = "";

try {
repo.saveAndFlush(d);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}

return msg;
}

@GetMapping("/readalldealer")
public List<Dealer> readAllDealer(){
return repo.getAllDealer();
}

@GetMapping("/readdealer")
public Dealer readDealer(@RequestParam("id")int id) {
return repo.findById(id).get();
}

@PutMapping("/updatedealer")
public String updateDealer(@RequestBody Dealer d) {
String msg = "";

try {
repo.saveAndFlush(d);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}

return msg;
}

@DeleteMapping("/deletedealer")
public String deleteDealer(@RequestParam("id")int id) {
```

```
String msg = "";

try {
repo.deleteById(id);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}

return msg;
}

}
```

ProductConroller.java

```
package com.ims.contoller;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.CrossOrigin;
import
org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import
org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

import com.ims.entity.Product;
import com.ims.repository.ProductRepo;
```

```java
@RestController
@CrossOrigin("http://localhost:3030")
public class ProductController {

@Autowired
ProductRepo repo;

@PostMapping("/createproduct")
public String createProduct(@RequestBody Product p) {
String msg = "";

try {
repo.saveAndFlush(p);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}

return msg;
}

@GetMapping("/readallproduct")
public List<Product> readAllProduct() {
return repo.getAllProduct();
}

@GetMapping("/readproduct")
public Product readProduct(@RequestParam("id") int id) {
return repo.findById(id).get();
}

@PutMapping("/updateproduct")
public String updateProduct(@RequestBody Product p) {
String msg = "";

try {
repo.saveAndFlush(p);
msg = "success";
} catch (Exception e) {
```

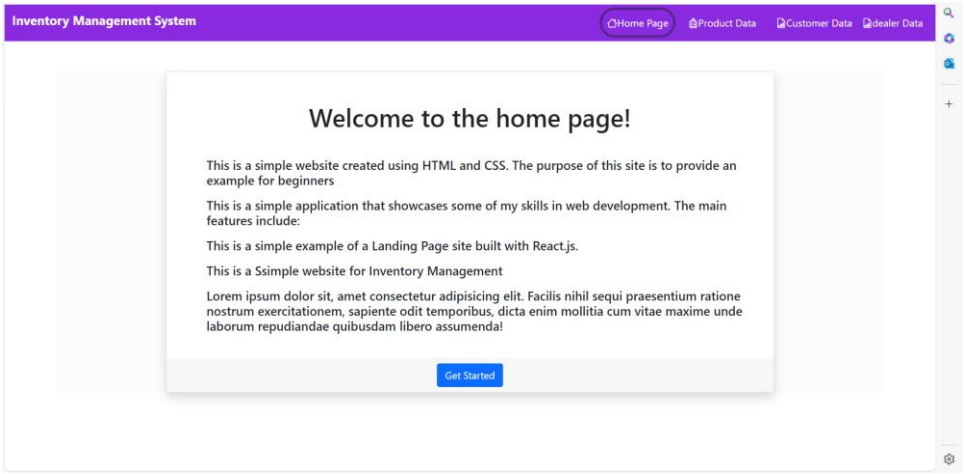
```
e.printStackTrace();
msg = "fail";
}

return msg;
}

@DeleteMapping("/deleteproduct")
public String deleteProduct(@RequestParam("id") int id) {
String msg = "";

try {
repo.deleteById(id);
msg = "success";
} catch (Exception e) {
e.printStackTrace();
msg = "fail";
}

return msg;
}

}
```

Application.properties

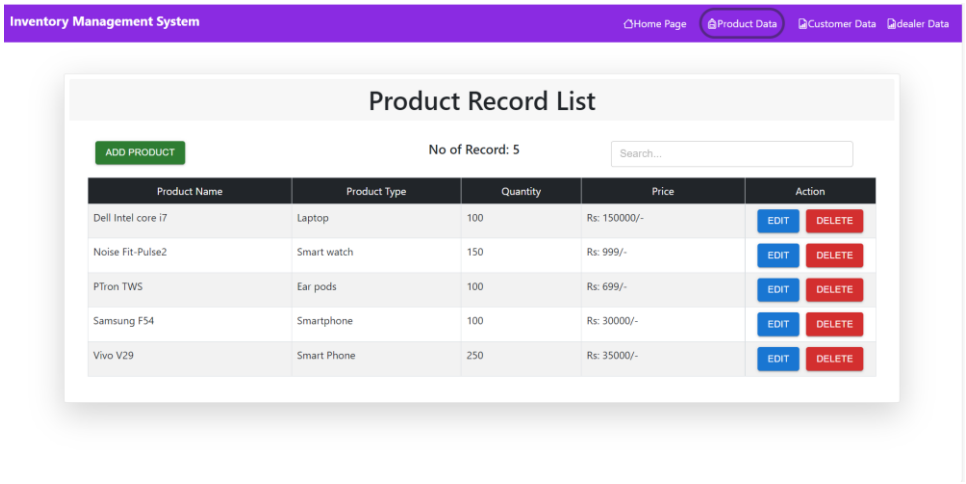
```
spring.application.name=InventoryManagementSystem
server.port=1260
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/ims_db
spring.datasource.username=root
spring.datasource.password=sam@6383587926
spring.datasource.driver-class-
name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql: true
```


Output:

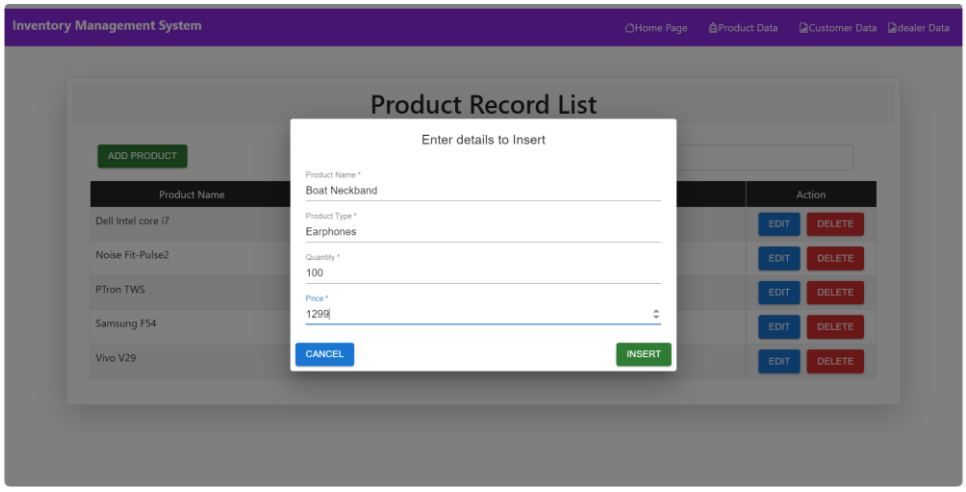
LandingPage:



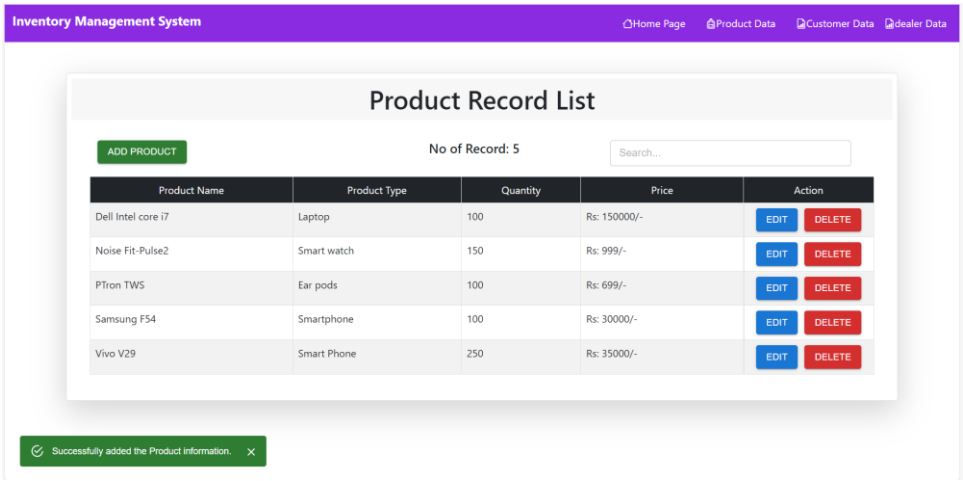
Product page:



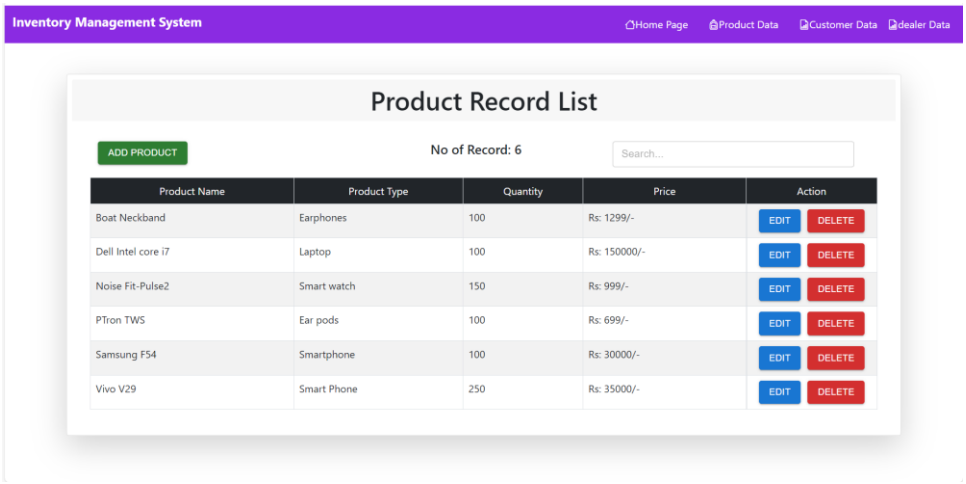
Add Product:



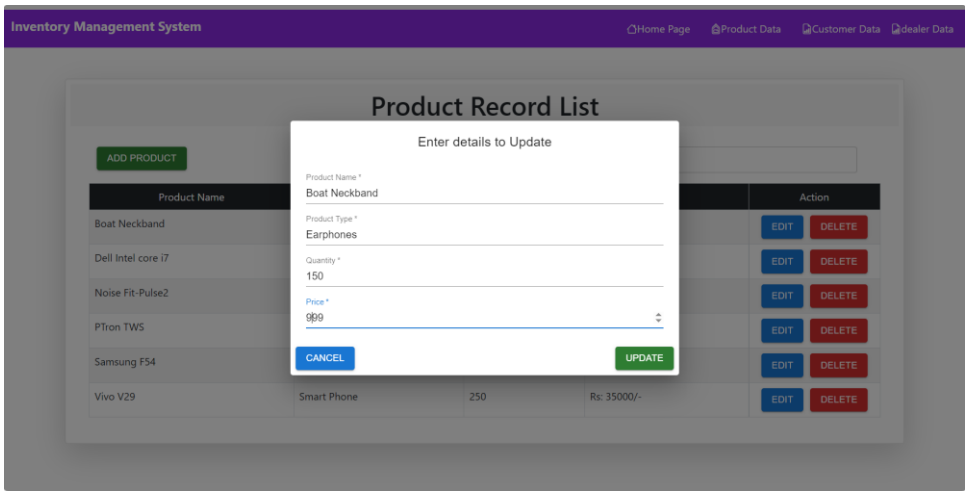
Add Product Success:



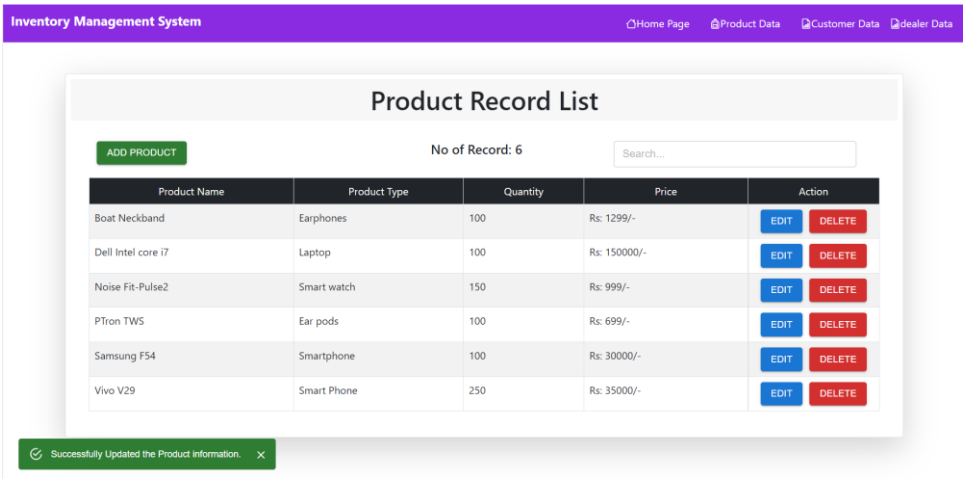
After insertion:



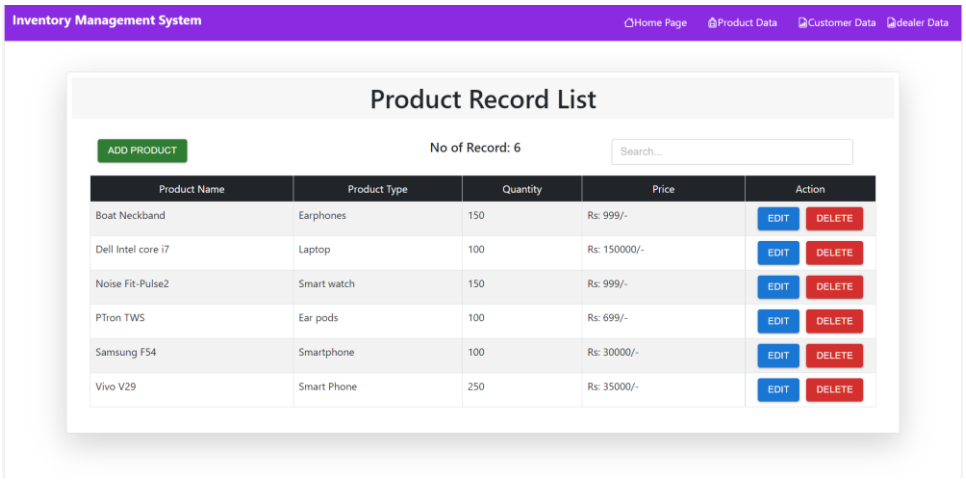
Edit Product:



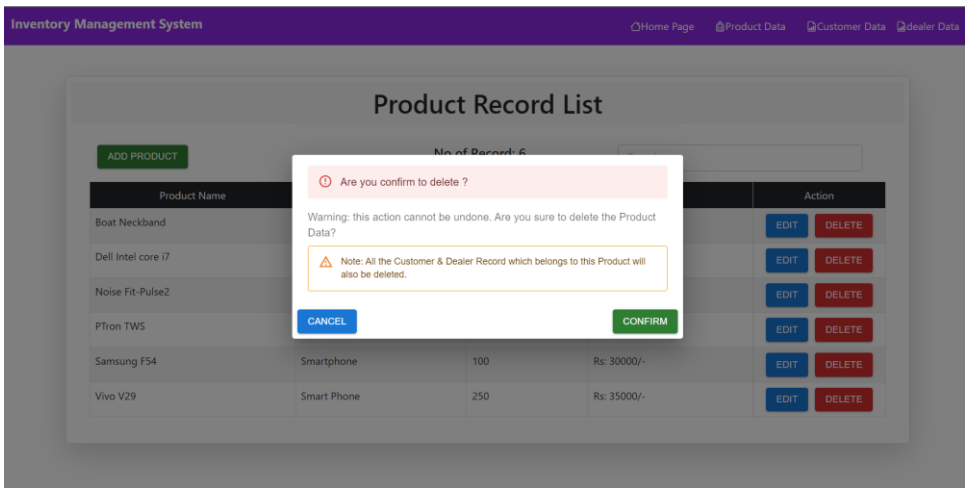
Edit Product Success



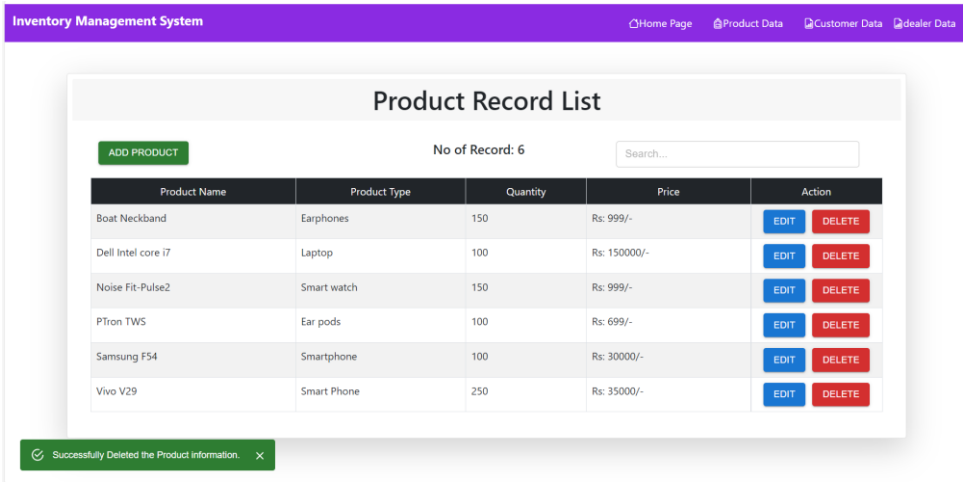
After Updating:



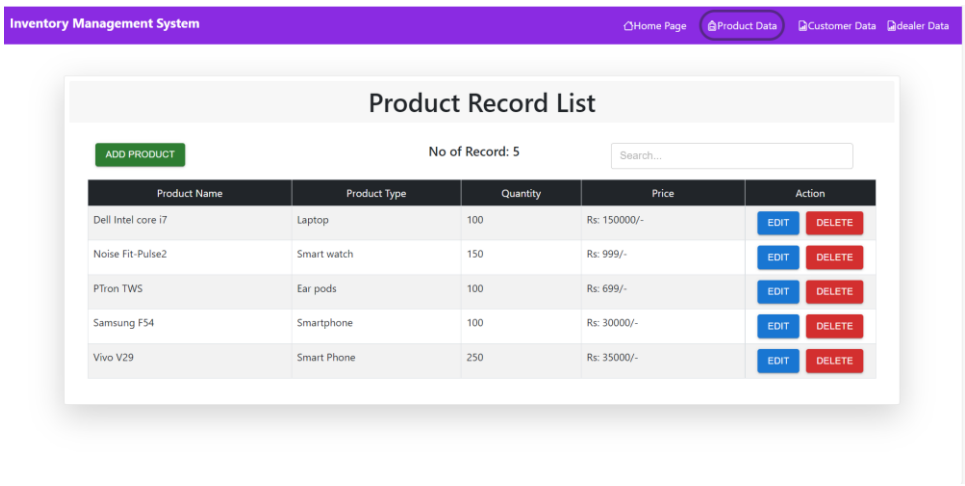
Delete product:



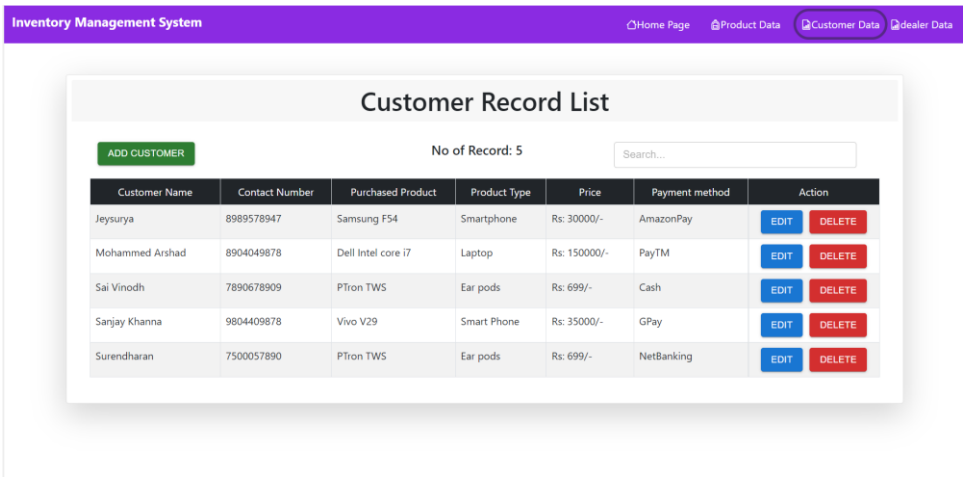
Delete Product Success:



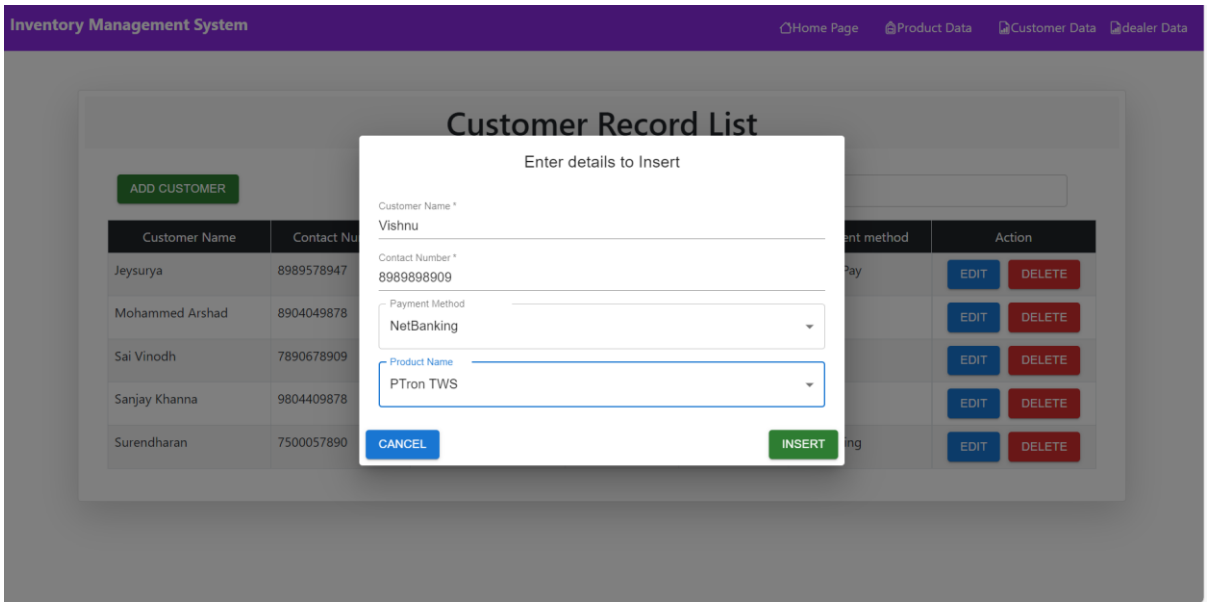
After Deletion:



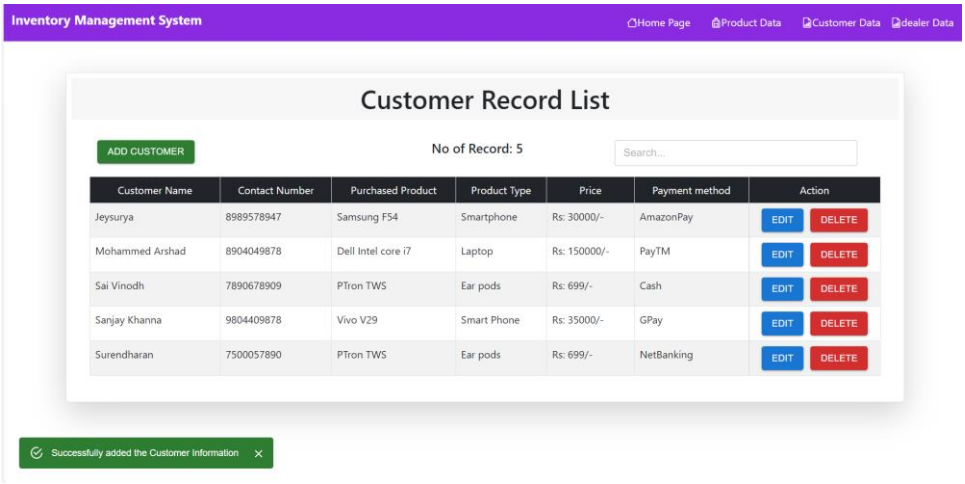
Customer Page:



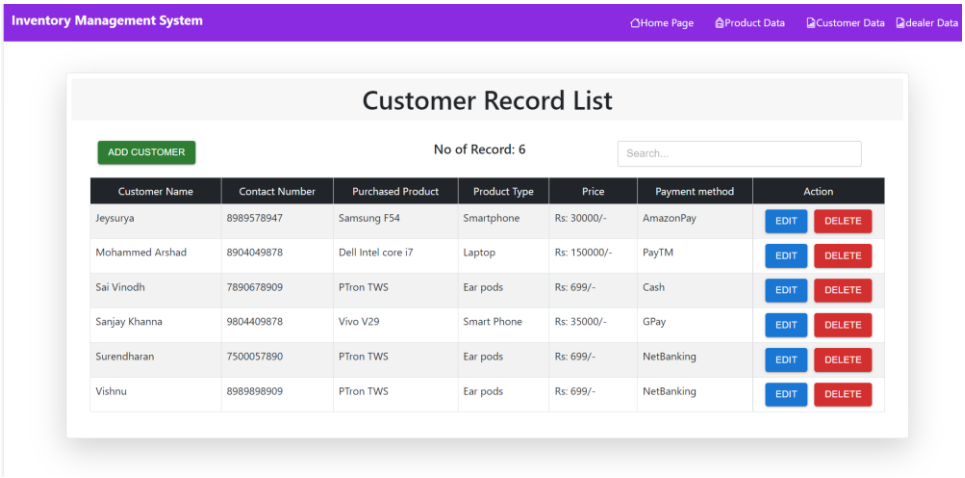
Add Customer



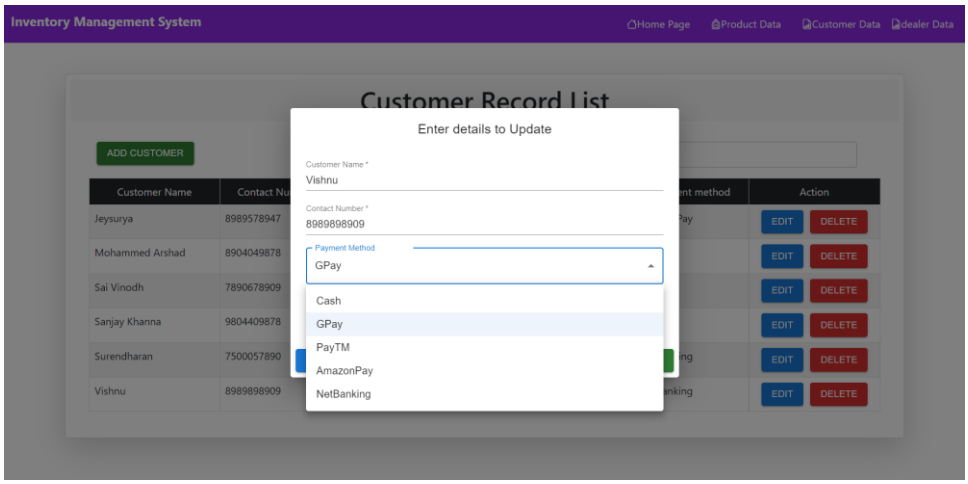
Add Customer Success:



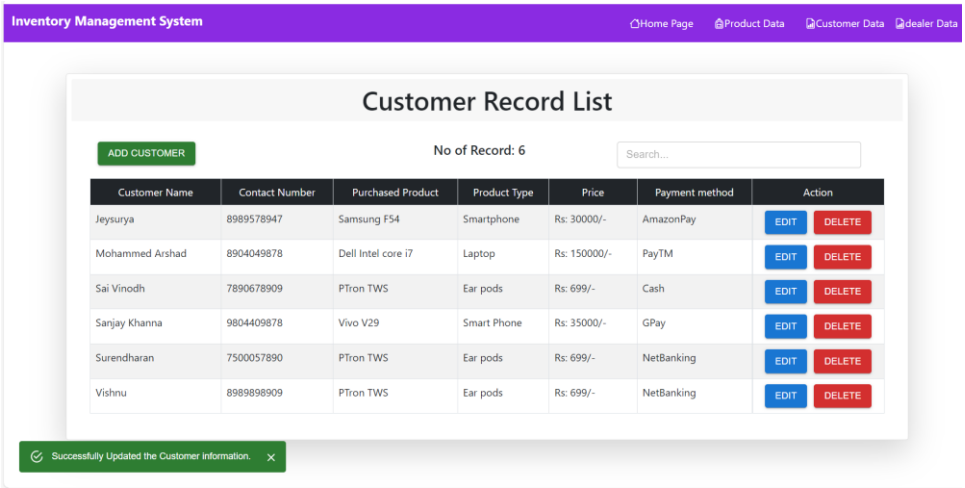
After insertion:



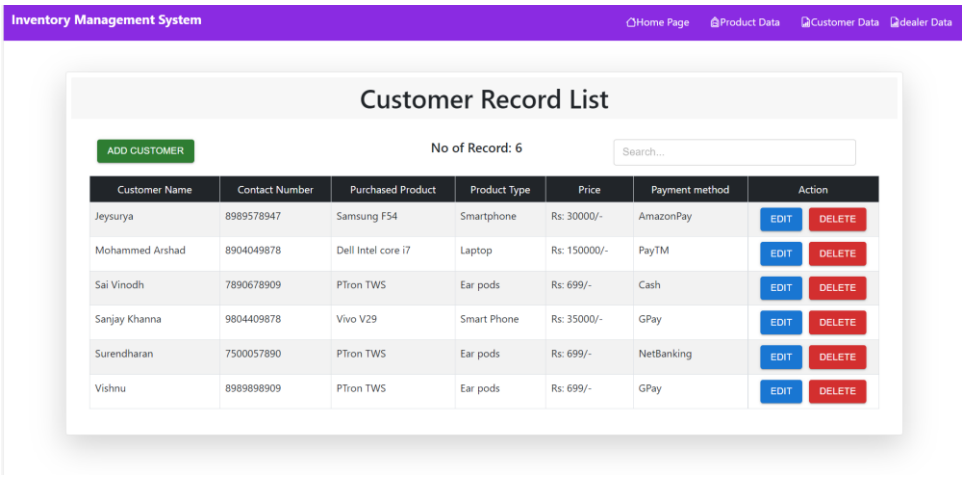
Edit Customer:



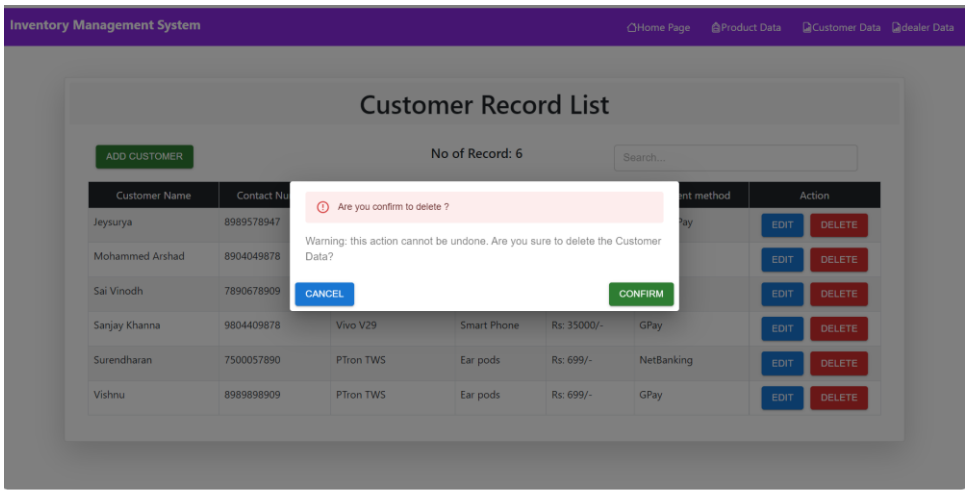
Edit Customer Success



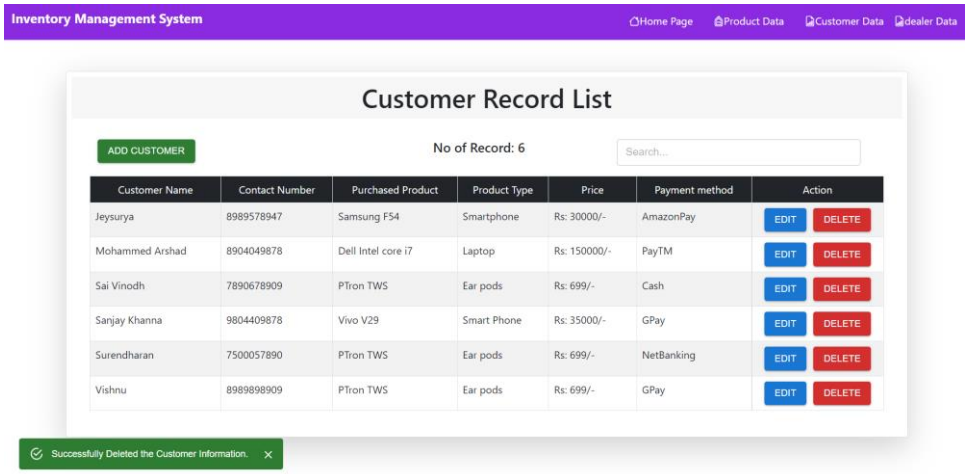
After Updating:



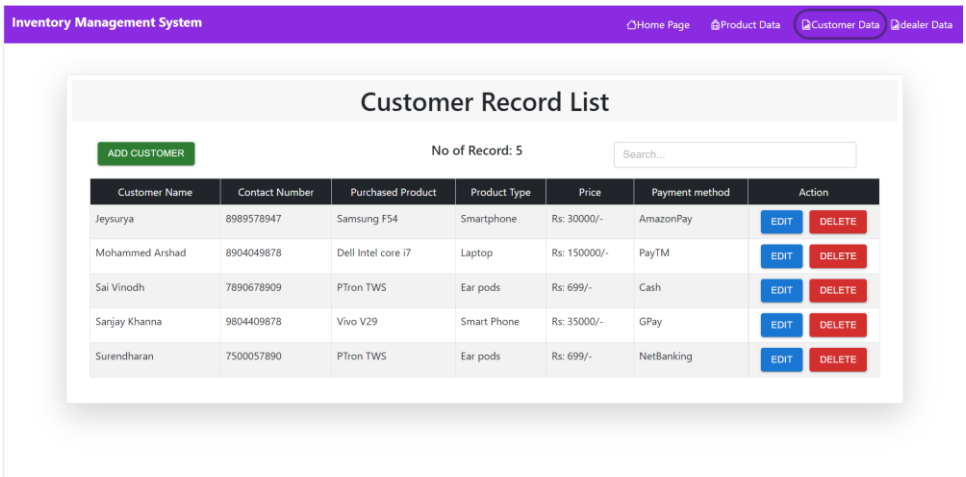
Delete Customer



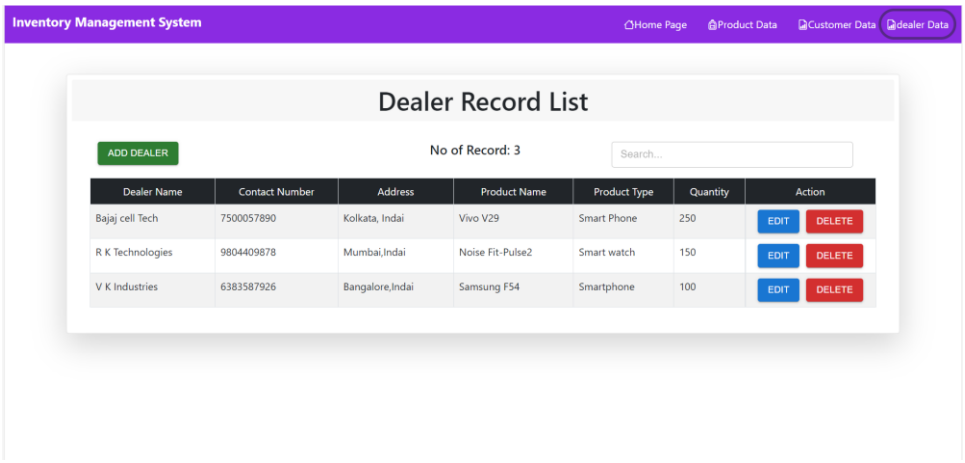
Delete Customer Success



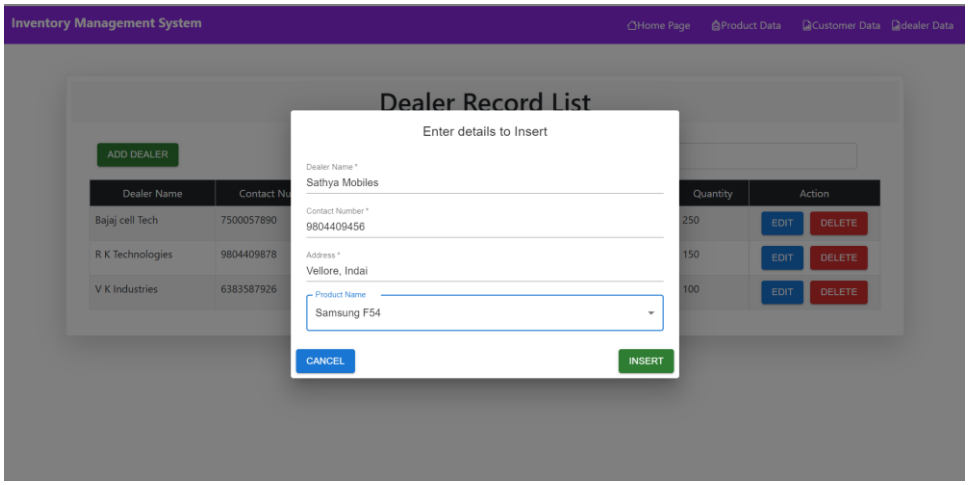
After Deletion



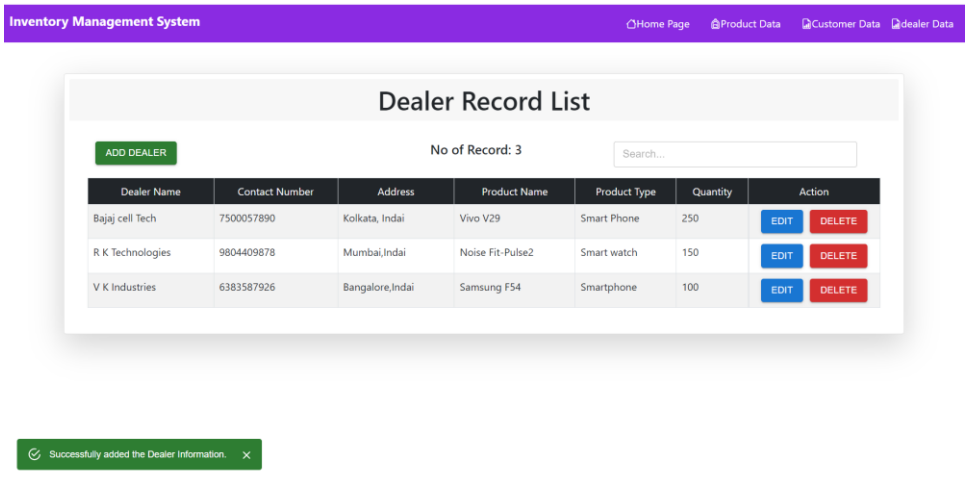
Dealer Page:



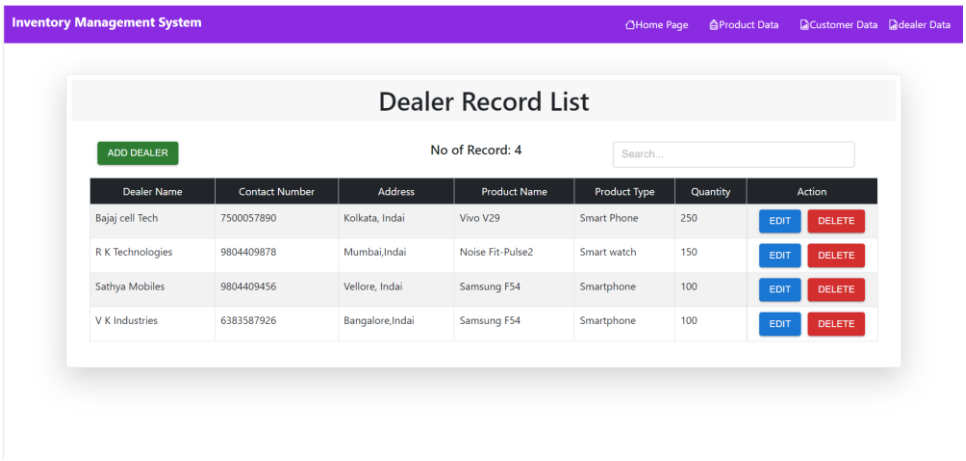
Add Dealer



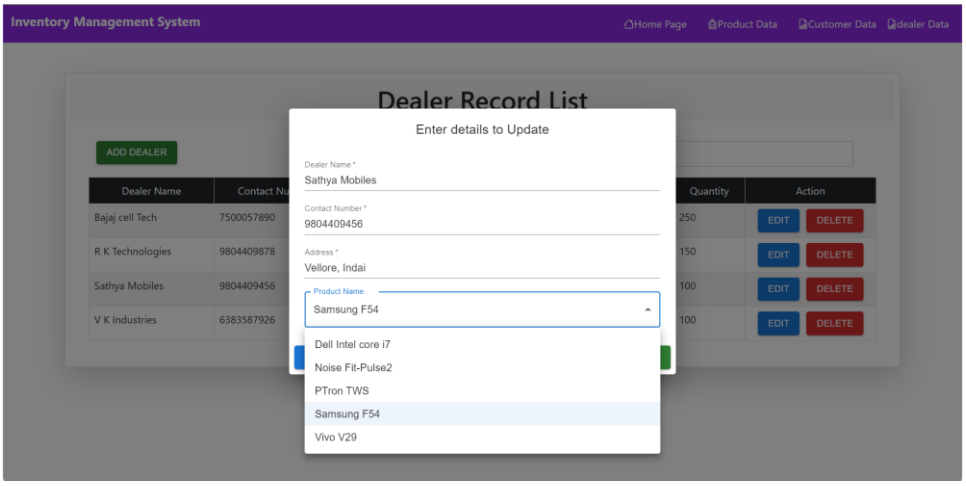
Add Dealer Success



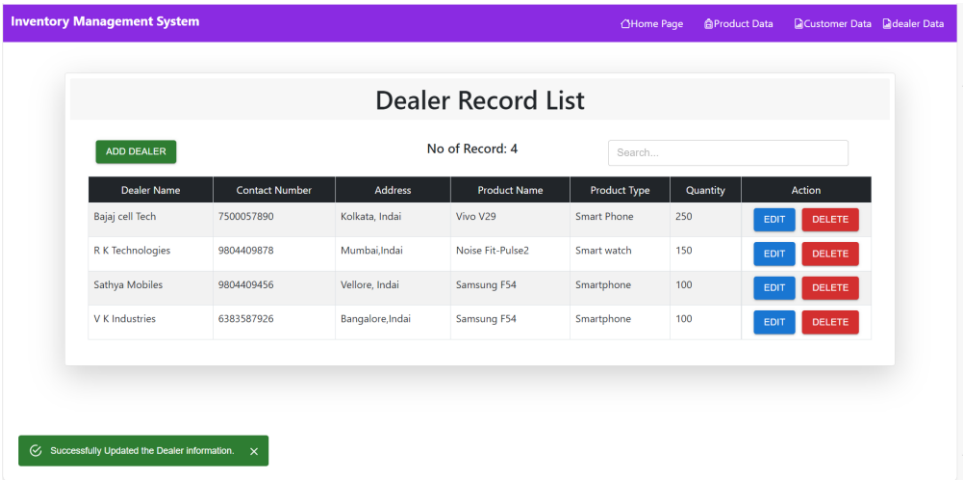
After Inserting



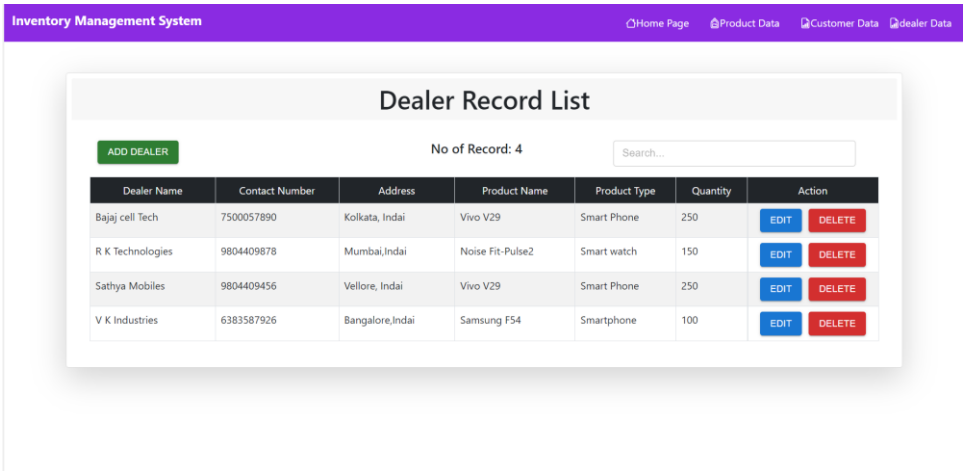
Edit Dealer:



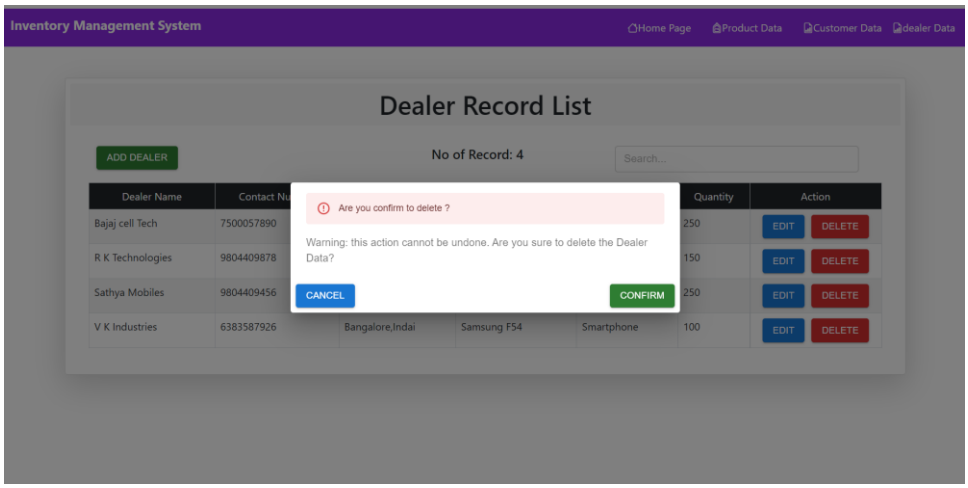
Edit Dealer Success



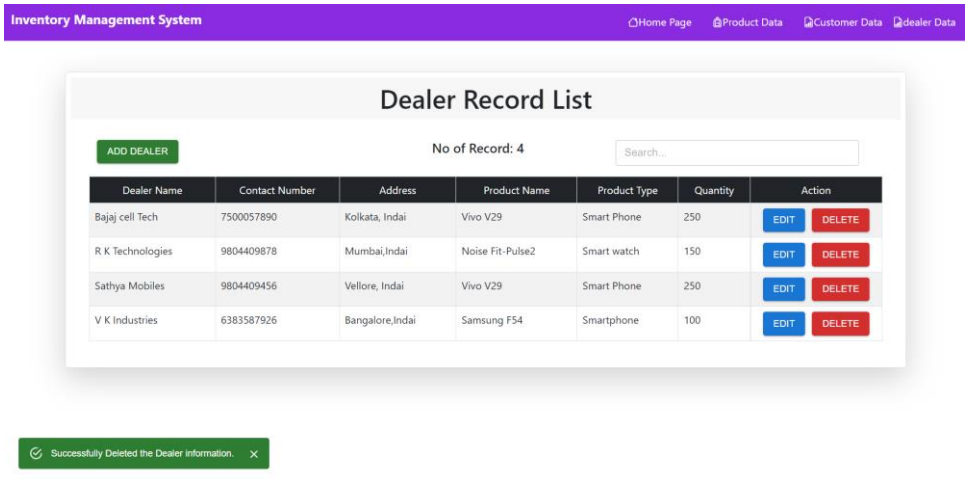
After Updating



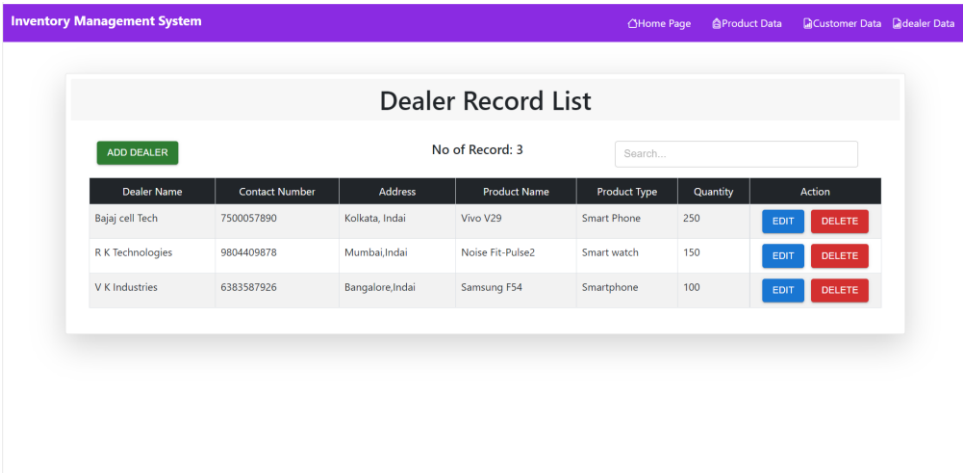
Delete Dealer



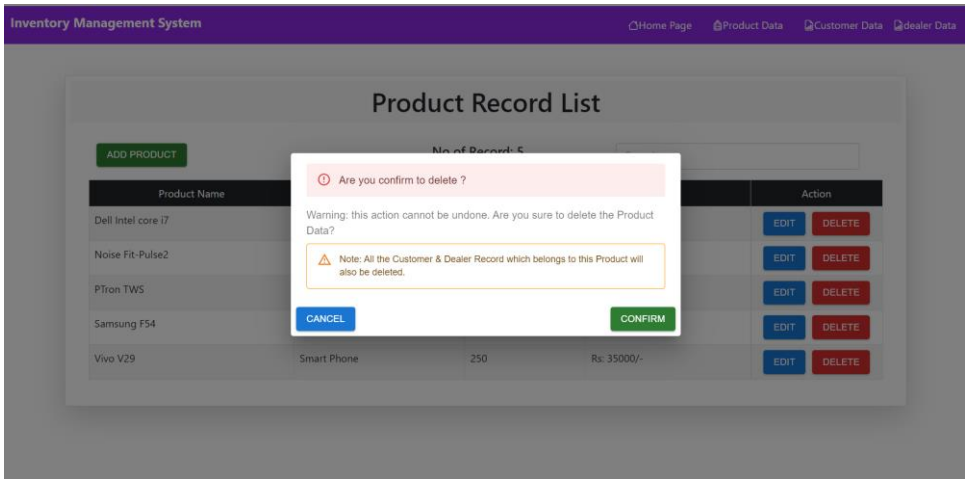
Delete Dealer Success



After Deletion



Effect Of Product delete:



Effect on customer record: