Order Management System

Team Drop Table;

BTech/III Year CSE/V Semester

15CSE302/Database Management Systems

Roll No.	Name
CB.EN.U4CSE18331	KARTHIKEYAN.RV
CB.EN.U4CSE18332	B. KIRTHI SAGAR
CB.EN.U4CSE18339	MIGHIL DATH
CB.EN.U4CSE18362	TANMAAY KANKARIA

Amrita School of Engineering, Coimbatore

Department of Computer Science and Engineering

2020 -2021 Odd Semester

Table of Contents

1. Project Preview 3
2. Project Analysis 4
3. Project Design 5
4. Normalization 10
5. Back-End Design 21
6. Front-End Design 28
7. Database Connectivity 30
8. Sample UI 31
9. Conclusion 35
10. References 36

1. Project Preview

Abstract

Our Order Management System is for a company that sells construction materials to customers who buy different products. Each customer can have more than one address and is assigned to a default salesperson who is a liaison between the customer and the company. The company sells multiple products. When a customer orders more than one product at a time, all order items are logically grouped in an order line. The OMS is implemented as an easy to use website where client-side scripting is facilitated through HTML, CSS, JavaScript with the data being stored and processed in the server-side script using Node.js.

Business Rules

(1) One shipment contains one order.

(2) To have three different users – Salesperson, Sales department clerks, and Warehouse supervisors.

(3) In the website, each user has access only to his/her functionality.

Need and motivation

Our Order Management System (OMS) can help a company selling construction materials to customers. It can perform multiple activities like tracking the stock of each product, viewing the list of customers and the products bought and so on. This will reduce costs and save time for the company and at the same time will facilitate easy retrieval and modification of data.

Tools used

Front-end: HTML, CSS, JavaScript, React js

Runtime: Node.js

Back-end Connection: Express, Database: PostgreSQL

Text-editor: Visual Studio Code

2. Project Analysis

Modules:

1. Customer Module:

The company must have at least one customer. Each customer in the database is assigned at least one address, contact number, and a default salesperson.

2. Product Module:

Each product has a price, a description, and some other characteristics. Orders can be placed for one or more product at a time.

3. Order Module:

The invoice number of the order is populated automatically in the database and cannot be changed by users. Each order has a status assigned to it: complete, shipped, invoiced, and so on. Usually, one shipment contains one order, but the database is designed in such a way that one order can be distributed between more than one shipment, as well as one shipment can contain more than one order.

4. User Module:

The OMS has three different user groups:

1. The salesperson is a liaison between the customer and the company. They have their products and customers assigned to them.

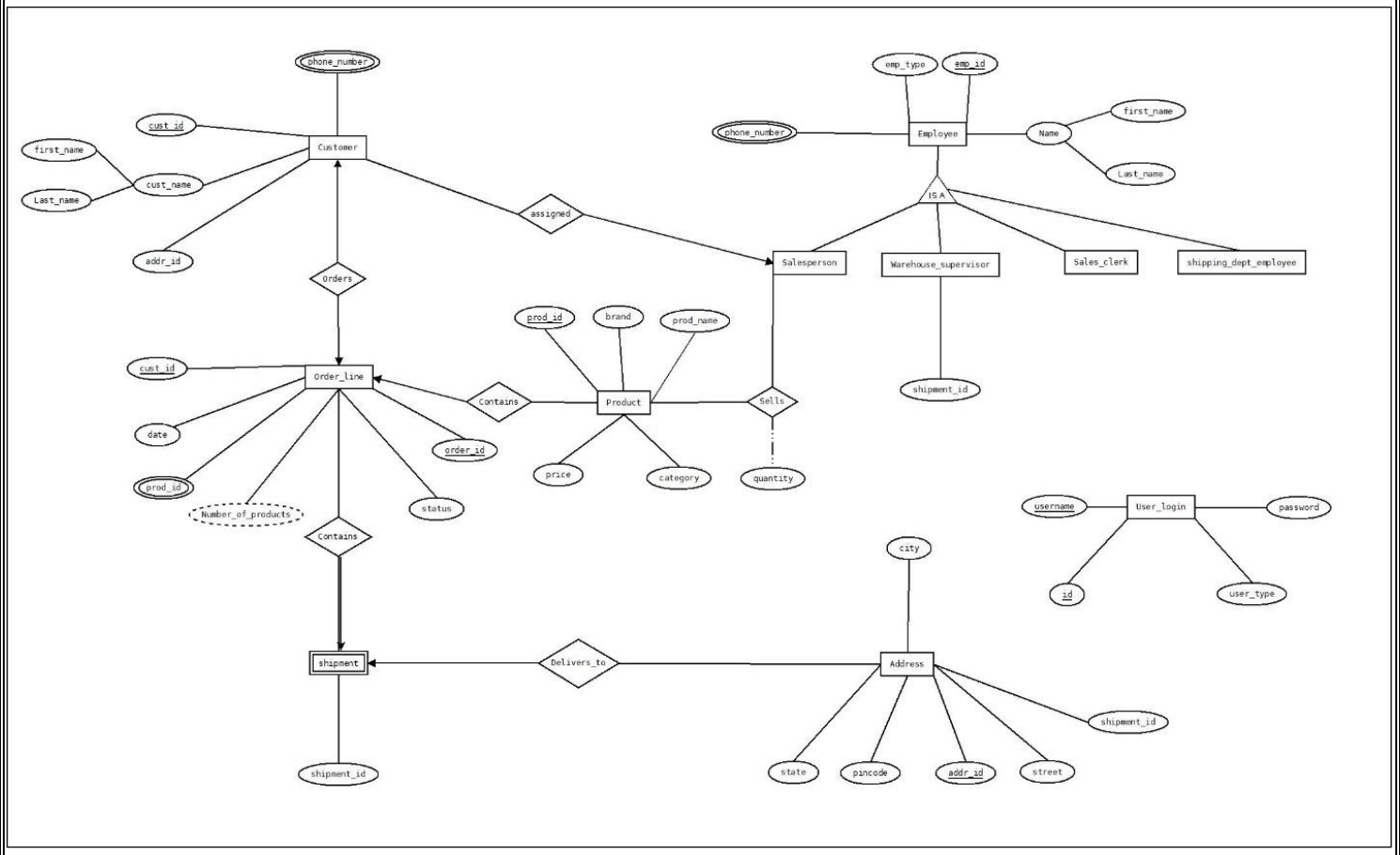
2. The sales department clerks' function is to enter or modify order and customer information.

3. Warehouse supervisors maintain the whole inventory and they can also add a new product.

Also, all three user groups view diverse database information under different angles, using reports and ad-hoc queries.

3. Project Design

ER Diagram



ER Diagram - Entities

Customers

Employees

Salesperson

Warehouse Supervisor

Sales Clerk

Shipping Dept. Employees

Order

Product

Shipment

Address

User Login

ER Diagram - Attributes

phone_number - Multivalued
cust_id
cust_name
addr_id

cust_id
date
prod_id
order_id
Number_of_products - Derived attribute

emp_type
emp_id
phone_number - Multivalued
name

brand
prod_name
price
quantity - Descriptive

category

shipment_id
state
city
street
pincode

username
password
id
user_type

ER Diagram - Relationships

Customer assigned to Salesperson

Salesperson is a Employee

Warehouse Supervisor is a Employee

Sales Clerk is a Employee

Shipping Dept. Employees is a Employee

Customer places Order

Order contains Product

Salesperson sells Product

Shipment contains Order

Shipment delivered to Address

ER to Relational Schema Mapping

Customer (cust_id, cust_name, addr_id, phone_number)

Employee (phone_number, emp_type, emp_id, Name, first_name, last_name)

Order (date, prod_id, cust id, number_of_products, status, order_id)

Product (prod_id, brand, prod_name, price, category)

Sells (quantity, prod_id, emp_id)

Salesperson (phone_number, emp_type, emp_id, Name, first_name, last_name)

Warehouse_supervisor (shipment_id, phone_number, emp_type, emp_id, Name, first_name, last_name)

Sales_clerk (phone_number, emp_type, emp_id, Name, first_name, last_name)

shipping_dept_employee (phone_number, emp_type, emp_id, Name, first_name, last_name)

Shipment (shipment_id, order_id)

Address (city, state, pincode, addr_id, street, shipment_id)

User_login (username, id, user_type, password)

Phone_number (user_type, user_id, phone_number)

Database Schema (before Normalization)

TABLE{Cust_id, First_name, Last_name, addr_id, Phone_number, cust_id, date ,number_of_products , status, order_id, status, shipment_id ,prod_id ,brand , prod_name, price, category, emp_type, emp_id ,shipment_id ,First_name, Last_name ,username, password, id ,user_type, city, state, pincode, addr_id, street, shipment_id}

Now, we decompose the table into four main tables

1)Each customer has cust_id, first_name, last_name, addr_id, phone_number

Thus, they form Table A.

2)Each order has order_id, addr_id, date, status, product_id, shipment_id

and each product has prod_id, brand, prod_name, price, stock_left, category

and each address has addr_id, pincode, state, city, street.

Thus, they form Table B.

3)Each employee has emp_id, emp_type, first_name, last_name, phone_number.

Thus, they form Table C.

4)Each user has an ID, username, password, user_type.

Thus, they form Table D.

4. Normalization

First Normal Form

Condition: For a table to be in first normal form, it must not have any multi-valued attributes.

TABLE-A: First Normal Form:

By applying the first normal form for Table A,

In table A, Phone_number is a multivalued attribute.

So, applying 1st normal form rule, we decompose the table A into

Table A1.1{cust_id, first_name, last_name, addr_id}

Table A1.2{cust_id, phone_number}

TABLE-B: First Normal Form:

In Table B, one order_id can have multiple product_id thus it is a multi-valued attribute.

So, applying 1st normal form rule: we decompose the table B into

Table B1.1{order_id, addr_id, date, status, shipment_id, pincode, state, city, street}

Table B1.2{product_id, brand, prod_name, price, stock_left, category}

TABLE-C: First Normal Form:

In table C, Phone_number is a multivalued attribute.

So, applying 1st normal form rule, we decompose the table C into

Table C1.1 {emp_id,emp_type,first_name,last_name}

Table C1.2{emp_id, Phone_number}

TABLE-D: First Normal Form:

In table D we have no multivalued attributes.

So, it is already in 1NF:

Table D1.1{ID, username, password, user_type}

Tables obtained from First normal form are,

Table A1.1{cust_id, first_name, last_name, addr_id}

Table A1.2{cust_id, phone_number}

Table B1.1{order_id, addr_id, date, status, shipment_id, pincode, state, city, street}

Table B1.2{product_id, brand, prod_name, price, stock_left, category}

Table C1.1 {emp_id, emp_type, first_name, last_name}

Table C1.2{emp_id, Phone_number}

Table D1.1{ID, username, password, user_type}

Second Normal Form

Condition: For a table should be in First normal form and should not have any partial Dependencies.

Table A1.1-Second Normal Form:

Table A1.1{cust_id, first_name, last_name, addr_id}

Primary key-cust_id

Functional Dependencies:

cust_id-->first_name

cust_id-->Last_name

cust_id-->addr_id

cust_id-->phone_number

No partial Dependencies and the table are already in 1NF so the table is in 2NF.

Thus, the resultant table is,

Table A2.1{cust_id, first_name, last_name, addr_id}

Table A1.2: Second Normal Form:

Table A1.2{cust_id, phone_number}

Primary key-cust_id

Functional Dependencies:

cust_id-->Phone_number

No partial Dependencies and the table are in 1NF so the table is in 2NF.

Thus, the table is,

Table A2.2{cust_id, phone_number}

Table B1.1-Second Normal Form

Table B1.1{order_id, addr_id, date, status, shipment_id, pincode, state, city, street}

Primary key-order_id, addr_id

Functional Dependencies:

order_id-->date

order_id-->status

order_id-->shipment_id

addr_id-->pincode

addr_id-->state

addr_id-->street

addr_id-->city

Here Order_id and addr_id has to be the primary key and as there are partial dependencies,

applying the 2NF rule, the table can be decomposed into

TableB2.1{order_id, status, date}

Table B2.2{addr_id, pincode, state, city, street}

Table B1.2: Second Normal Form

Table B1.2{prod_id, brand, prod_name, price, stock_left, category}

Primary key-prod_id

Functional Dependencies:

prod_id-->brand

prod_id-->prod_name

prod_id-->price

prod_id-->stock_left

Prod_id-->category

Here the Table is in 1NF and it has no partial dependencies.

Thus, the table is

TableB2.2.1{product_id, brand, prod_name, price, stock_left, category}

Table B1.3: Second Normal Form

Table B1.3{order_id, shipment_id}

Primary key-order_id

Functional Dependencies:

Order_id-->shipment_id

Here the Table is in 1NF and it has no partial dependencies.

Thus, the table is

Table B2.3{order_id, shipment_id}

Table C1.1: Second Normal Form

Table C1.1 {emp_id, emp_type, first_name, last_name}

Primary key-emp_id

Functional Dependencies:

emp_id-->emp_type

emp_id-->first_name

emp_id-->last_name

emp_id-->phone_number

The Table is in 1NF and there are no partial dependencies so it is in 2NF

Thus, the table is

Table C2.1 {emp_id, emp_type, first_name, last_name}

Table C1.2: Second Normal Form

Table C1.2{emp_id, Phone_number}

Primary key-phone_number

Functional Dependencies:

phone_number-->emp_id

The table is in 1NF and it has no partial dependencies

Thus, the table is

Table C2.2{emp_id, Phone_number}

Table D1.1: Second Normal Form

Table D1.1{ID, username, password, user_type}

Primary key- username, ID

Functional Dependencies:

Username-->user_type

ID-->user_type

ID-->username

Username-->ID

The table is in 1NF and it has no partial dependencies

Thus, the table is

Table D2.1{ID, username, password, user_type}

Tables after the Second Normal Form:

Table A2.1.1{cust_id, first_name, last_name, addr_id}

Table A2.2.2{cust_id, phone_number}

TableB2.1.1{order_id, status, date}

Table B2.1.2{addr_id, pincode, state, city, street}

TableB2.2.1{product_id, brand, prod_name, price, stock_left, category}

Table B2.3.1{order_id, shipment_id}

Table C2.1.1 {emp_id, emp_type, first_name, last_name}

Table C2.2.1emp_id, Phone_number}

Table D2.1{ID, username, password, user_type}

Third Normal Form

Condition: Table should be in 2NF and should not have transitive dependencies.

Table A2.1.1: Third Normal Form-

Table A2.1.1{cust_id, first_name, last_name, addr_id}

Functional Dependencies:

cust_id-->first_name

cust_id-->Last_name

cust_id-->addr_id

cust_id-->phone_number

The table is in 2NF and it has no transitive dependencies.

Thus, the table is

A3.1.1{cust_id, first_name, last_name, addr_id}

Table A2.2.2: Third Normal Form-

Table A2.2.2{cust_id, phone_number}

Functional Dependencies:

cust_id-->Phone_number

The table is in 2NF and it has no transitive dependencies

Thus, the table is

Table A3.2.2{cust_id, phone_number}

TableB2.1.1: Third Normal Form-

TableB2.1.1{order_id, status, date}

Functional Dependencies:

order_id-->date

order_id-->status

order_id-->shipment_id

The table is in 2NF and it has no transitive dependencies

Thus, the table is

TableB3.1.1{order_id, status, date}

Table B2.1.2: Third Normal Form

Table B2.1.2{addr_id, pincode, state, city, street}

Functional Dependencies:

addr_id-->pincode

addr_id-->state

addr_id-->street

addr_id-->city

The table is in 2NF and it has no transitive dependencies

Thus, the table is

Table B3.1.2{addr_id, pincode, state, city, street}

TableB2.2.1: Third Normal Form

TableB2.2.1{product_id, brand, prod_name, price, stock_left, category}

Functional Dependencies:

prod_id-->brand

prod_id-->prod_name

prod_id-->price

prod_id-->stock_left

Prod_id-->category

The table is in 2NF and it has no transitive dependencies

Thus, the table is

TableB3.2.1{product_id, brand, prod_name, price, stock_left, category}

Table C2.1.1: Third Normal Form

Table C2.1.1 {emp_id, emp_type, first_name, last_name}

Functional Dependencies:

emp_id-->emp_type

emp_id-->first_name

emp_id-->last-named

The table is in 2NF and it has no transitive dependencies

Thus, the table is

Table C3.1.1 {emp_id, emp_type, first_name, last_name}

Table C2.2.1: Third Normal Form

Table C2.2.1{emp_id, Phone_number}

Functional Dependencies:

emp_id-->phone_number

The table is in 2NF and it has no transitive dependencies

Thus, the table is

Table C3.2.1{emp_id, Phone_number}

Table D2.1: Third Normal Form

Table D2.1{ID, username, password, user_type}

Functional Dependencies:

Username-->user_type

ID-->user_type

ID-->username

Username-->ID

The table is in 2NF and it has no transitive dependencies

Thus, the table is

Table D3.1.1{ID, username, password, user_type}

Tables after Third Normal Form

A3.1.1{cust_id, first_name, last_name, addr_id}

A3.2.2{cust_id, phone_number}

B3.1.2{addr_id, pincode, state, city, street}

B3.1.1{order_id, status, date}

B3.2.1{product_id, brand, prod_name, price, stock_left, category}

B3.3.1{order_id, shipment_id}

C3.1.1 {emp_id, emp_type, first_name, last_name}

C3.2.1{emp_id, Phone_number}

D3.1.1{ID, username, password, user_type}

BCNF

Applying BCNF rules to the schema, it turns out that the relations are already in BCNF.

So, the schema remains unchanged and thus no more decomposition.

Database schema after Normalization

Customer {cust_id, first_name, last_name, addr_id}

CustomerPhoneNumber {cust_id, phone_number}

Address {addr_id, pincode, state, city, street}

OrderLine {order_id, cust_id}

OrderInfo {order_id, status, date}

Shipment {order_id, shipment_id}

Products {prod_id, brand, prod_name, price, stock_left, category}

Employee {emp_id, emp_type, first_name, last_name}

EmployeePhoneNumber {emp_id, Phone_number}

UserLogin {ID, username, password, user_type}

Considering the Specialization of the employee class by Four other entities, we get four more tables as follows.

Salesperson {emp_id, emp_type, first_name, last_name}

WarehouseSupervisor {emp_id, emp_type, first_name, last_name}

SalesClerk {emp_id, emp_type, first_name, last_name}

ShippingDepartmentEmployee {emp_id, emp_type, first_name, last_name}

5. Back-End Design

Table Creation Queries:

```
create table customer (
    cust_id NUMERIC(10) NOT NULL PRIMARY KEY,
    first_name varchar(50) NOT NULL ,
    last_name varchar(50),
    addr_id BIGINT NOT NULL REFERENCES addr(addr_id)
);


create table cust_phone (
    cust_id BIGINT NOT NULL REFERENCES customer(cust_id),
    ph_no VARCHAR(15) NOT NULL
);


create table order_line (
    cust_id BIGINT NOT NULL REFERENCES customer(cust_id),
    order_id BIGSERIAL NOT NULL PRIMARY KEY
);


create table order_info (
    order_id BIGINT NOT NULL REFERENCES order_line(order_id),
    date_ DATE NOT NULL DEFAULT CURRENT_DATE,
    order_status VARCHAR(20) NOT NULL
);


create table products (
    prod_id BIGSERIAL NOT NULL PRIMARY KEY,
    brand VARCHAR(50) NOT NULL,
    prod_name VARCHAR(50) NOT NULL,
    price NUMERIC(10, 3) NOT NULL,
    category VARCHAR(20) NOT NULL,
    stock NUMERIC(5) NOT NULL,
);

create table order_products (
    order_id BIGINT NOT NULL REFERENCES order_line(order_id),
    prod_id BIGINT NOT NULL REFERENCES products(prod_id)

);
```

```sql
create table user_login (
    id BIGSERIAL NOT NULL PRIMARY KEY,
    user_name VARCHAR(50) NOT NULL,
    passwd VARCHAR(50) NOT NULL,
    user_type VARCHAR(10) NOT NULL
);

create table employee(
    emp_id NUMERIC(10) NOT NULL PRIMARY KEY,
    emp_type VARCHAR(15) NOT NULL
);

create table salesperson(
    emp_id NUMERIC(10) NOT NULL PRIMARY KEY,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NOT NULL
);

create table WareHouseSupervisor (
    emp_id NUMERIC(10) NOT NULL PRIMARY KEY,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NULL
);

create table SalesClerk (
    emp_id NUMERIC(10) NOT NULL PRIMARY KEY,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NULL
);

create table addr (
    addr_id BIGSERIAL NOT NULL PRIMARY KEY,
    state_  VARCHAR(50) NOT NULL,
    city VARCHAR(50) NOT NULL,
    street VARCHAR(50) NOT NULL,
    pincocde NUMERIC(10) NOT NULL
);

create table ShippingDepartmentEmployee(
    emp_id NUMERIC(10) NOT NULL PRIMARY KEY,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NULL
);
```

```
create table shipment (
    order_id BIGINT NOT NULL REFERENCES order_line(order_id),
    shipment_id NUMERIC() NOT NULL
);

create table emp_ph (
    emp_id BIGINT NOT NULL REFERENCES employee(emp_id),
    ph_no VARCHAR(15) NOT NULL
);

create table sp_prod (
    emp_id BIGINT NOT NULL REFERENCES employee(emp_id),
    prod_id BIGINT NOT NULL REFERENCES products(prod_id)
);

create table sales_cust (
    cust_id NUMERIC(10) NOT NULL REFERENCES customer(cust_id),
    emp_id NUMERIC(10) NOT NULL REFERENCES employee(emp_id)
);
```

Tables in the Database

	addr_id [PK] bigint	state_ character varying (50)	city character varying (50)	street character varying (50)	pincode numeric (10)
1	1	Tamil Nadu	Trichy	Baker Street	620015
2	2	Tamil Nadu	Chennai	Amrita Salai	928015
3	3	Tamil Nadu	Trichy	ABC Street	620015
4	4	Kerala	Palakad	DEF Street	123456
5	5	Kerala	Palakad	Street Street	123456

	cust_id bigint	ph_no character varying (15)
1	1	9512924133
2	2	9525125200
3	3	9887562143
4	4	7075687420
5	5	9856743210

	cust_id [PK] numeric (10)	first_name character varying (50)	last_name character varying (50)	addr_id bigint
1	1	Rick	Morty	1
2	2	James	King	2
3	3	Antony	Davis	3
4	4	Hannibal	Lector	4
5	5	Will	Graham	5

	cust_id bigint	order_id [PK] bigint
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

	emp_id bigint	ph_no character varying (15)
1	6	7085463210
2	7	9884756410
3	8	8056789490
4	9	7103698745
5	10	9856789420
6	11	9863254177
7	12	7078423585
8	13	9887564123
9	14	7234241231
10	15	8567352444
11	16	7674578458
12	17	3779656345
13	18	4945463246
14	19	9756346345
15	20	6734532463
16	21	8655634623
17	22	5435235235
18	23	9754363453
19	24	7545635634

	order_id bigint	prod_id bigint
1	1	1
2	1	2
3	1	4
4	2	6
5	2	7
6	3	3
7	3	5
8	3	7
9	4	1
10	5	2

	order_id bigint	date_ date	order_status character varying (20)
1	1	2020-05-...	COMPLETE
2	2	2020-05-...	SHIPPED
3	3	2020-06-...	INVOICED
4	5	2020-09-...	INVOICED
5	4	2020-10-...	COMPLETE

	order_id bigint	shipment_id bigint
1	1	123
2	2	124
3	3	123
4	4	125
5	5	125

	prod_id [PK] bigint	brand character varying (50)	prod_name character varying (50)	price numeric (10,3)	category character varying (20)	stock numeric (5)
1	1	Havels	Fan	5000.000	Electrical	25
2	2	Philips	LED Lamp	870.000	Electrical	15
3	3	AsianPaints	Black Paint	6750.000	Building Materials	3
4	4	Company1	PVC Pipe	3200.000	Pipes and Fittings	13
5	5	Company1	1000 litre Water Tank	45000.000	Pipes and Fittings	10
6	6	AsianPaints	Tractor Emulsion	5000.000	Building Materials	7
7	7	Havels	Wire	5000.000	Electrical	11

	emp_id [PK] numeric (10)	first_name character varying (20)	last_name character varying (20)
1	6	Tim	Ren
2	7	Ben	Ten
3	8	Gwen	Ten
4	9	Kevin	Joe
5	10	Jackie	Jhon

	emp_id bigint	prod_id bigint
1	6	1
2	6	4
3	6	5
4	6	6
5	7	2
6	7	3
7	7	4
8	8	1
9	8	2
10	8	4
11	8	5
12	8	5
13	9	3
14	9	7
15	9	6
16	10	2
17	10	5
18	10	6

	id [PK] bigint	user_name character varying (50)	passwd character varying (50)	user_type character varying (10)
1	1	customer1	customer1	CM
2	2	customer2	customer2	CM
3	3	customer3	customer3	CM
4	4	customer4	customer4	CM
5	5	customer5	customer5	CM
6	6	salesperson1	salesperson1	SP
7	7	salesperson2	salesperson2	SP
8	8	salesperson3	salesperson3	SP
9	9	salesperson4	salesperson4	SP
10	10	salesperson5	salesperson5	SP
11	11	warehouse1	warehouse1	WS
12	12	warehouse2	warehouse2	WS
13	13	warehouse3	warehouse3	WS
14	14	warehouse4	warehouse4	WS
15	15	warehouse5	warehouse5	WS
16	16	salesclerk1	salesclerk1	SC
17	17	salesclerk2	salesclerk2	SC
18	18	salesclerk3	salesclerk3	SC
19	19	salesclerk4	salesclerk4	SC
20	20	salesclerk5	salesclerk5	SC
21	21	shipping1	shipping1	SE
22	22	shipping2	shipping2	SE
23	23	shipping3	shipping3	SE
24	24	shipping4	shipping4	SE
25	25	shipping5	shipping5	SE

6. Front-End Design

Framework used: React js

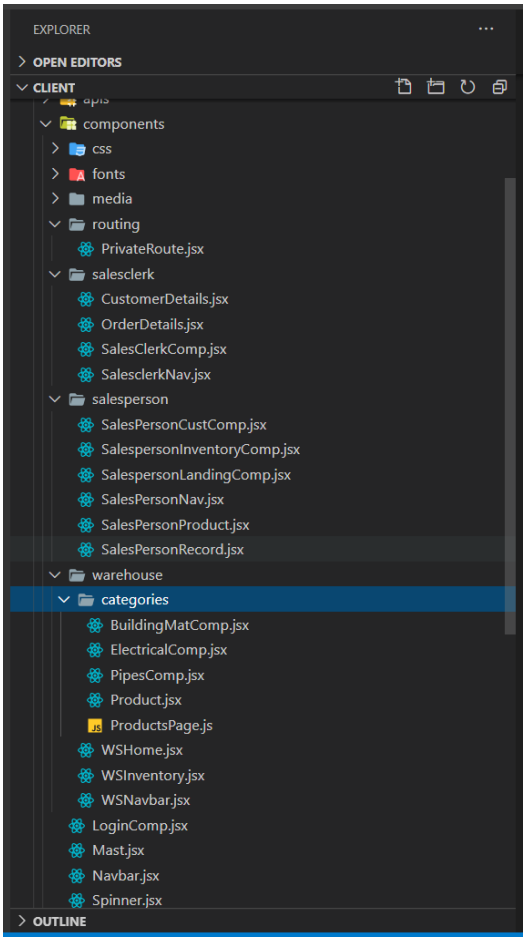
Link: https://reactjs.org/

Language: JavaScript

React does a great job in making the process of building user interfaces or UI components easier.

How does it work?

- The HTML elements are wrapped into different UI components in the react.
- The data is grabbed from the UI and passed to the respective components to load the dynamically updated data.
- The use of states is the main key for dynamic data.



These are the list of UI components used for our project's front-end design.

Sample screenshot of a UI component in React,

SalesPersonNav.jsx

src > components > salesperson > SalesPersonNav.jsx > ...

```jsx
import React from 'react';
import { connect } from 'react-redux';
import PropTypes from 'prop-types';
import { logout } from '../../actions/auth';

const SalesPersonNav = ({ logout }) => {
    return (
        <div className='SP'>
            <header>
                <nav
                className="SpNav">
                    <a href="#" class="Home">DropTable</a>
                    <section>
                        <a href="/salesperson/inventory" class="Inv">Inventory</a>
                        <a href="/salesperson/customer" class="Cust">Customers</a>
                        <a onClick={logout} href="#!" class="Sout">Sign out</a>
                    </section>
                </nav>
            </header>
        </div>
    )
}

SalesPersonNav.propTypes = {
    logout: PropTypes.func.isRequired,
};


export default connect(null, { logout })(SalesPersonNav)

```

7. Database Connectivity

We have used PostgreSQL as the database management system for our project. PostgreSQL is a popular SQL database. It has been in active development for the last 30+ years and is considered to be one of the most advanced relational databases out there. PostgreSQL is also easy to learn and setup compared to other relational databases available. Because of its free and open-source nature, this is a popular choice among startups.

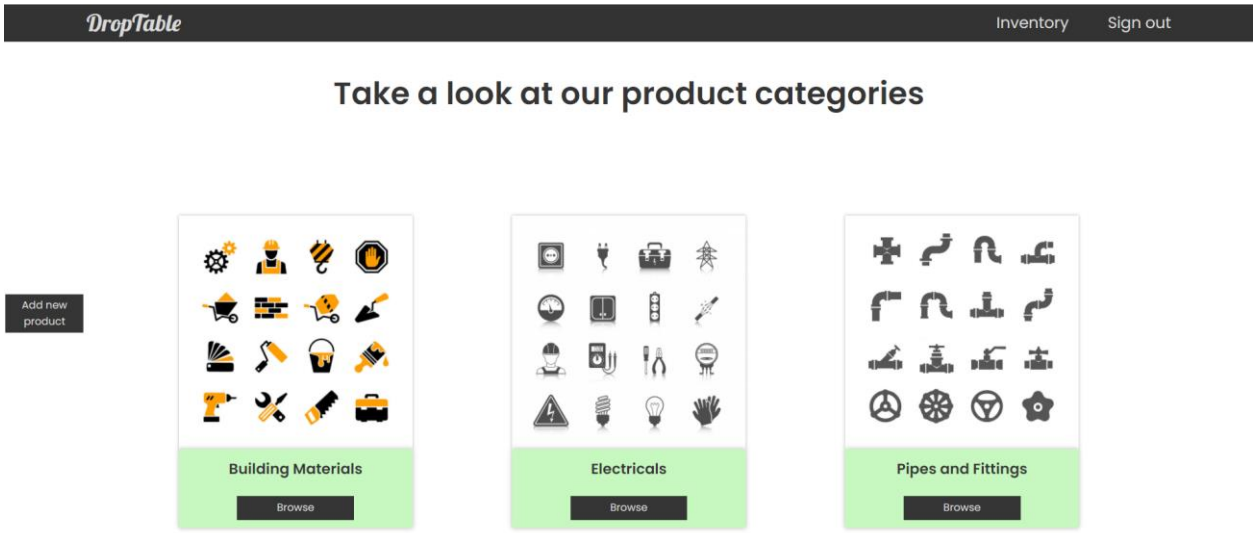
We have used node-Postgres to connect our application to the PostgreSQL database. Node-Postgres is a collection of node.js modules that provides an easy way to integrate Node.js with PostgreSQL.

Connectivity Code:

```
const { Pool } = require('pg');
require("dotenv").config();

const devConfig = {
    user: process.env.PG_USER,
    password: process.env.PG_PASSWORD,
    host: process.env.PGHOST,
    databse: process.env.PGDATABASE,
    port: process.env.PGPORT
}

const proConfig = {
    connectionString: process.env.DATABSE_URL
}

const pool = new Pool(process.env.NODE_ENV === "production" ? proConfig : dev
Config)

module.exports = {
    query: (text, params) => pool.query(text, params),
}
```

8. Sample UI and Code

Warehouse Supervisor Categories



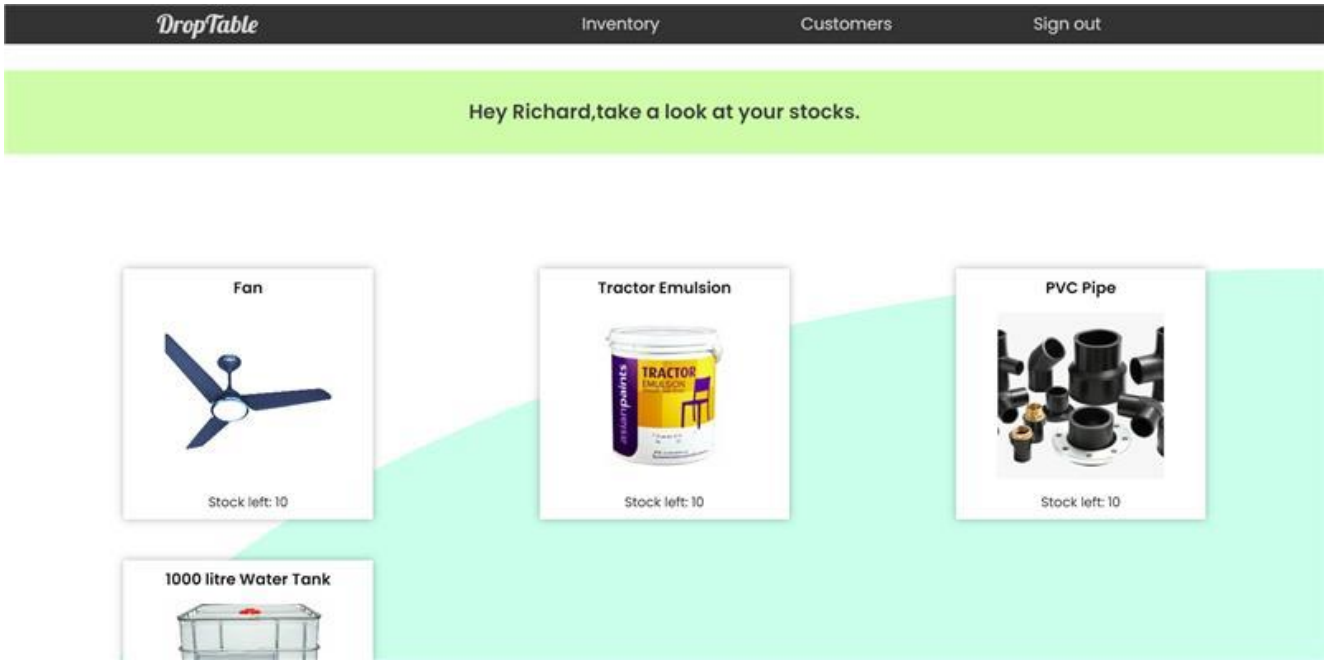

```
ent > src > components > warehouse > categories > BuildingMatComp.jsx > ...
24              }
25          })();
26
27      }, [])
28
29      if(user)
30      {
31          if(user.user_type !== "WS")
32              return <Redirect to='/login' />
33      }
34
35      return(
36          <div className="WS">
37              <div className="Products" id="Pipe">
38                  <h2>Take a look at our stocks.
39                  </h2>
40
41                  <div className="Products-Container">
42                      { buildingmat && buildingmat.map(element => {
43
44                          return (
45                              <Product
46                                  key={element.prod_id}
47                                  prod_name={element.prod_name}
48                                  id={element.prod_id}
49                                  price={element.price}
50                                  brand={element.brand}
51                                  stock={element.stock}
52                                  image={element.image}
53                              />
54                          )
55                      })}
56                  </div>
57              </div>
58          </div>
59      );
60  }
61
62  BuildingMatComp.propTypes = {
```

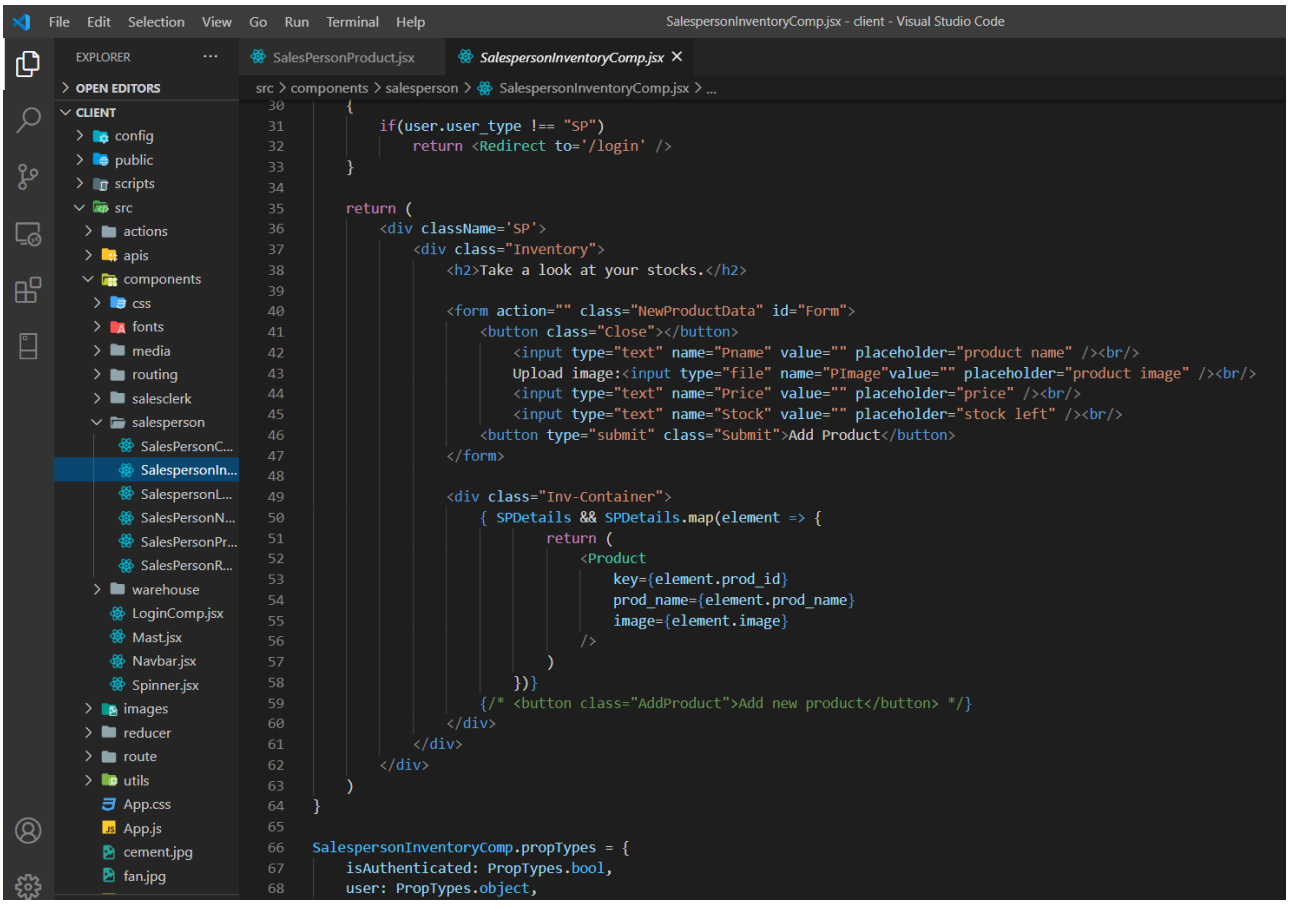
ElectricalComp.jsx

client > src > components > warehouse > categories > ElectricalComp.jsx > ...

```
        if(user.user_type !== "WS")
            return <Redirect to='/login' />
    }

    return(
        <div className="WS">
            <div class="Products" id="Pipe">
                <h2>These are the electrical products.
                </h2>


                <form action="" class="NewProductData" id="Form">
                    <button class="Close"></button>
                    <input type="text" name="Pname" value="" placeholder="product name"/><br/>
                    Upload image:<input type="file" name="PImage"value="" placeholder="product image"/><br/>
                    <input type="text" name="Price" value="" placeholder="price"/><br/>
                    <input type="text" name="Stock" value="" placeholder="stock left"/><br/>
                    <button type="submit" class="Submit">Add Product</button>
                </form>

                <div class="Products-Container">
                    { electrical && electrical.map(element => {


                        return (
                            <Product
                                key={element.prod_id}
                                id={element.prod_id}
                                prod_name={element.prod_name}
                                price={element.price}
                                stock={element.stock}
                                brand={element.brand}
                                image={element.image}
                            />
                        )
                    })}
                </div>
            </div>
        </div>
    );
```

Salesperson Inventory





Warehouse Supervisor Inventory

WSInventory.jsx

client > src > components > warehouse > WSInventory.jsx > ...

```
58
59      }
60
61      return(
62          <div className="WS">
63
64              <form
65                  action=""
66                  className="NewProductData"
67                  id="Form"
68                  style={{display: showForm ? "inherit": "none" }}
69                  >
70                  <button onClick={e => setShowForm(false)} className="Close"></button>
71                  <input onChange={e => onChange(e) } type="text" name="prod_name" value={formData.prod_name} placeholder="product name"/><br/>
72                  <input onChange={e => onChange(e) } type="text" name="brand" value={formData.brand} placeholder="product brand"/><br/>
73                  Upload image:<input onChange={e => onChange(e) } type="file" name="form_image" value={formData.form_image} placeholder="product imag
74                  <input onChange={e => onChange(e) } type="text" name="price" value={formData.price} placeholder="price"/><br/>
75                  <section className="category">
76                      <select value={formData.category} onChange={e => onChange(e)} name="category" id="category">
77                          <option value="electrical">Electrical</option>
78                          <option  (JSX attribute) React.OptionHTMLAttributes<HTMLOptionElement>.value?: string | number | readonly string[]
79                          <option value="pipes_fitting">Pipes and Fittings</option>
80                      </select>
81                  </section>
82                  <input onChange={e => onChange(e) } type="text" name="stock" value={formData.stock} placeholder="stock left"/><br/>
83                  <button onClick={e => onSubmit(e)} type="submit" className="Submit">Add Product</button>
84              </form>
85
86              <div className="Inventory">
87                  <h1
88                  >Take a look at our product categories</h1>
89                  <div
90
91                  className="Container">
92
93                      <section className="Category">
94
95                          <div className="Imageholder">
96                              <img src={require("../media/images/categories/buildingmat.jpg")}/>
```

9. Conclusion

With our Order Management System, the construction company can efficiently manage and process the orders placed by a customer through the company website. An order goes through various phases till it finally reaches the customer. At any point of time, the customer can check the status of the order online with the order id received while placing the order.

On the website, company employees can perform multiple activities like add a new product, track and update the stock of each product, view the list of customers and the products bought and so on. This will not only save time and reduce costs for the company, but it will also facilitate easy retrieval and modification of data.

Lastly, through this project we successfully implemented the following DBMS Concepts:

Schema

ER Model

Normalization

Database connectivity

10. References

Websites references:

GeeksForGeeks - https://www.geeksforgeeks.org

Stack Overflow - https://stackoverflow.com