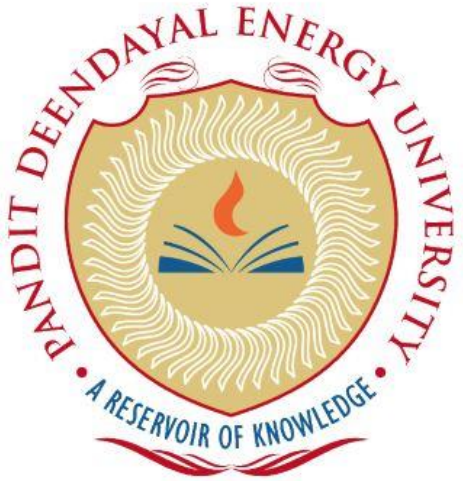
PANDIT DEENDAYAL ENERGY UNIVERSITY

SCHOOL OF TECHNOLOGY



SUPERMARKET MANAGEMENT SYSTEM

20CP209P

Database Management Systems Lab

B.Tech. (Computer Science and Engineering)

Semester 4

Submitted To:	**Submitted By:**	**Roll Numbers:**
Dr. Hargeet Kaur	Harsh Shah	21BCP359
	Vraj Patel	21BCP362
	Tanish Suthar	21BCP366

INDEX

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude and appreciation to all those who have contributed to the successful completion of our supermarket management system project. First and foremost, we would like to extend our sincere thanks to our instructor, Dr. Hargeet Kaur, for her continuous guidance, support, and encouragement throughout the project. Her valuable insights and feedback have played a crucial role in shaping our ideas and enhancing the overall quality of our work.

We are grateful to our team members, Harsh Shah, Vraj Patel, Tanish Suthar for their unwavering dedication, hard work, and exceptional teamwork in completing this project successfully. Each member has brought their unique skills and expertise to the table, allowing us to create a comprehensive and efficient supermarket management system that we are truly proud of.

Furthermore, we would like to acknowledge the resources and references that have been instrumental in enriching our understanding of the subject matter. Online tutorials, research papers, and other relevant sources have provided invaluable guidance and knowledge throughout the project.

In conclusion, we extend our heartfelt thanks to everyone who has supported and assisted us in completing this project. We are excited to have developed a supermarket management system that will streamline operations and improve the overall shopping experience for customers.

SUPERMART MANAGEMENT SYSTEM

Objective:

The Objective of our supermarket management system is to create an efficient database that streamlines inventory, sales, customer and employee management, and order processing. The project aims to improve and optimize supermarket operations, and enable better decision-making for management.

Overview:

The supermarket management system project is an innovative solution that simplifies the management of a supermarket's daily operations. By consolidating various aspects such as inventory, sales, orders, and staff information into a single system, it provides a cohesive platform for better organization and increased efficiency within the supermarket.

Description:

The supermarket management system is a relational database designed to store and manage all pertinent data for a supermarket, with a focus on facilitating effective communication and collaboration among employees and the owner. The database comprises several tables, including CUSTOMER, CATEGORY, PRODUCT, EMPLOYEE, ORDERS, ORDER_ITEM, PAYMENT, SALES, EMPLOYEE_MOBILE, and EMPLOYEE_ORDER. Each table represents a specific aspect of the supermarket's operations and is linked through primary and foreign keys, ensuring data integrity and allowing for complex queries.

DATABASE SCHEMA

- **CUSTOMER** – Store's customer information like name, mobile number, and a unique ID.

```
CREATE TABLE CUSTOMER (
    C_ID VARCHAR(10) PRIMARY KEY,
    C_NAME VARCHAR(40),
    C_MOBILE VARCHAR(10),
    CONSTRAINT CHECK_PHONE CHECK (C_MOBILE REGEXP '^[0-9]{10}$')
);
```

- **CATEGORY** - Stores product categories with a unique ID and category name.

```
CREATE TABLE CATEGORY (
    CATEGORY_ID VARCHAR(20) PRIMARY KEY,
    CATEGORY_NAME VARCHAR(20)
);
```

- **PRODUCT** - Stores product details like name, category, cost price, selling price, and stock.

```
CREATE TABLE PRODUCT (
    PRODUCT_ID VARCHAR(10) PRIMARY KEY, PRODUCT_NAME,
            VARCHAR(40),
    CATEGORY_ID VARCHAR(20),
    COST_PRICE DECIMAL(10,2),
    SELLING_PRICE DECIMAL(10,2),
    STOCK INTEGER,
    CONSTRAINT CHECK_STOCK CHECK (STOCK >= 0),
    FOREIGN KEY (CATEGORY_ID) REFERENCES CATEGORY
            (CATEGORY_ID)
);
```

- **EMPLOYEE** - Stores employee information like name, gender, salary, and a unique ID.

```
CREATE TABLE EMPLOYEE (
    EMP_ID VARCHAR(10) PRIMARY KEY,
    FIRST_NAME VARCHAR(15),
    LAST_NAME VARCHAR(15),
    EMP_GENDER ENUM('MALE', 'FEMALE'),
    EMP_SALARY DECIMAL(10,2)
);
```

- **ORDERS** - Stores order details like customer ID, amount, and order date.

```
CREATE TABLE ORDERS (
    ORDER_ID VARCHAR(10) PRIMARY KEY,
    CUSTOMER_ID VARCHAR(10),
    AMOUNT DECIMAL(10,2),
    ORDER_DATE DATE,
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(C_ID)
);
```

- **ORDER_ITEM** - Stores details of items in an order, including product ID and quantity.

```
CREATE TABLE ORDER_ITEM (
    ORDER_ID VARCHAR(10),
    PRODUCT_ID VARCHAR(10),
    QUANTITY INTEGER,
    PRIMARY KEY (ORDER_ID, PRODUCT_ID),
    FOREIGN KEY (ORDER_ID) REFERENCES ORDERS(ORDER_ID),
    FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT(PRODUCT_ID)
);
```

- **PAYMENT** - Stores payment details like order ID, payment mode, amount, and payment date.

```
CREATE TABLE PAYMENT (
    PAY_ID VARCHAR(10) PRIMARY KEY,
    ORDER_ID VARCHAR(10),
    PAY_MODE ENUM('CASH', 'UPI', 'CARD'),
    AMOUNT DECIMAL(10,2),
    PAY_DATE TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ORDER_ID) REFERENCES ORDERS(ORDER_ID)
);
```

- **SALES** - Stores sales information like product ID, quantity sold, and profit.

```
CREATE TABLE SALES (
    PRODUCT_ID VARCHAR(10),
    QUANTITY_SOLD INTEGER,
    PROFIT DECIMAL(10,2),
    FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT(PRODUCT_ID)
);
```

- **EMPLOYEE_MOBILE** - Stores employee mobile numbers, allowing multiple numbers for one employee.

```
CREATE TABLE EMPLOYEE_MOBILE (
    EMP_ID VARCHAR(10),
    EMP_MOBILE VARCHAR(10),
    PRIMARY KEY (EMP_ID, EMP_MOBILE),FOREIGN KEY (EMP_ID)
REFERENCES EMPLOYEE(EMP_ID)
);
```

- **EMPLOYEE_ORDER** - Stores the relationship between employees and the orders they process.

```
CREATE TABLE EMPLOYEE_ORDER (
        EMP_ID VARCHAR(10),
        ORDER_ID VARCHAR(10),
        PRIMARY KEY (EMP_ID, ORDER_ID),
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE(EMP_ID),
        FOREIGN KEY (ORDER_ID) REFERENCES ORDERS(ORDER_ID)
);
```

INSERTED VALUES

```
INSERT INTO CUSTOMER VALUES ('C001', 'RAJESH KUMAR', 9876543210);
INSERT INTO CUSTOMER VALUES ('C002', 'SONIA GUPTA', 9812345678);
INSERT INTO CUSTOMER VALUES ('C003', 'NEHA SHARMA', 9810987654);
INSERT INTO CUSTOMER VALUES ('C004', 'AYUSH PATEL', 6587541232);

INSERT INTO CATEGORY VALUES('C101','BAKERY');
INSERT INTO CATEGORY VALUES('C102','BEVERAGES');
INSERT INTO CATEGORY VALUES('C103','DAIRY');
INSERT INTO CATEGORY VALUES('C104','GROCERY');
INSERT INTO CATEGORY VALUES('C105','PRODUCE');
INSERT INTO CATEGORY VALUES('C106','CEREALS');
INSERT INTO CATEGORY VALUES('C107','SNACKS');
INSERT INTO CATEGORY VALUES('C108','HOUSEHOLD ITEMS');
INSERT INTO CATEGORY VALUES('C109','BEAUTY');

INSERT INTO ORDER_ITEM VALUES ('O001', 'P101', 5);
INSERT INTO ORDER_ITEM VALUES ('O001', 'P102', 3);
INSERT INTO ORDER_ITEM VALUES ('O002', 'P103', 2);
```

```
INSERT INTO PRODUCT VALUES('P101','OATS', 'C106',150,200,89);
INSERT INTO PRODUCT VALUES('P102','MILK', 'C103',45,50,157);
INSERT INTO PRODUCT VALUES('P103','KETCHUP', 'C104',90,100,50);
INSERT INTO PRODUCT VALUES('P104','RICE', 'C106',500,550,26);
INSERT INTO PRODUCT VALUES('P105','BANANA', 'C105',80,100,69);
INSERT INTO PRODUCT VALUES('P106','COKE', 'C102',100,125,46);
INSERT INTO PRODUCT VALUES('P107','HAIR OIL', 'C109',80,85,72);
INSERT INTO PRODUCT VALUES('P108','RICE', 'C101', 350, 400, 100);

INSERT INTO ORDERS VALUES ('O001', 'C004', 4500, '2022-02-23');
INSERT INTO ORDERS VALUES ('O002', 'C002', 3200, '2023-03-15');
INSERT INTO ORDERS VALUES ('O003', 'C001', 1200, '2022-01-08');
INSERT INTO ORDERS VALUES ('O004', 'C003', 1200, '2022-06-11');

INSERT INTO PAYMENT VALUES ('P001', 'O001', 'UPI', 4500, '2022-03-14');
INSERT INTO PAYMENT VALUES ('P002', 'O002', 'CARD', 3200, '2022-03-15');
INSERT INTO PAYMENT VALUES ('P003', 'O003', 'CASH', 1200, '2022-03-15');

INSERT INTO EMPLOYEE VALUES ('E001', 'SURESH', 'KUMAR', 'MALE',
30000);
INSERT INTO EMPLOYEE VALUES ('E002', 'RENU', 'SINGH', 'FEMALE', 25000);
INSERT INTO EMPLOYEE VALUES ('E003', 'ANIL', 'VERMA', 'MALE', 28000);

INSERT INTO EMPLOYEE_MOBILE VALUES ('E001', 9879654218);
INSERT INTO EMPLOYEE_MOBILE VALUES ('E002', 7845321975);
INSERT INTO EMPLOYEE_MOBILE VALUES ('E003', 6548724387);

INSERT INTO EMPLOYEE_ORDER VALUES ('E001', 'O001');
INSERT INTO EMPLOYEE_ORDER VALUES ('E002', 'O002');
INSERT INTO EMPLOYEE_ORDER VALUES ('E003', 'O003');
```

```
INSERT INTO SALES VALUES ('P101', 15, 5000);
INSERT INTO SALES VALUES ('P102', 10, 300);
INSERT INTO SALES VALUES ('P103', 5, 500);
```

FUNCTIONAL DEPENDENCIES

1. **CUSTOMER:**

   C_ID -> C_NAME, C_MOBILE

2. **CATEGORY:**

   CATEGORY_ID -> CATEGORY_NAME

3. **PRODUCT:**

   PRODUCT_ID -> PRODUCT_NAME, CATEGORY_ID, COST_PRICE, SELLING_PRICE, STOCK

4. **EMPLOYEE:**

   EMP_ID -> FIRST_NAME, LAST_NAME, EMP_GENDER, EMP_SALARY, EMP_MOBILE

5. **ORDERS:**

   ORDER_ID -> CUSTOMER_ID, AMOUNT, ORDER_DATE, PRODUCTS

6. **ORDER_ITEM:**

   (ORDER_ID, PRODUCT_ID) -> QUANTITY

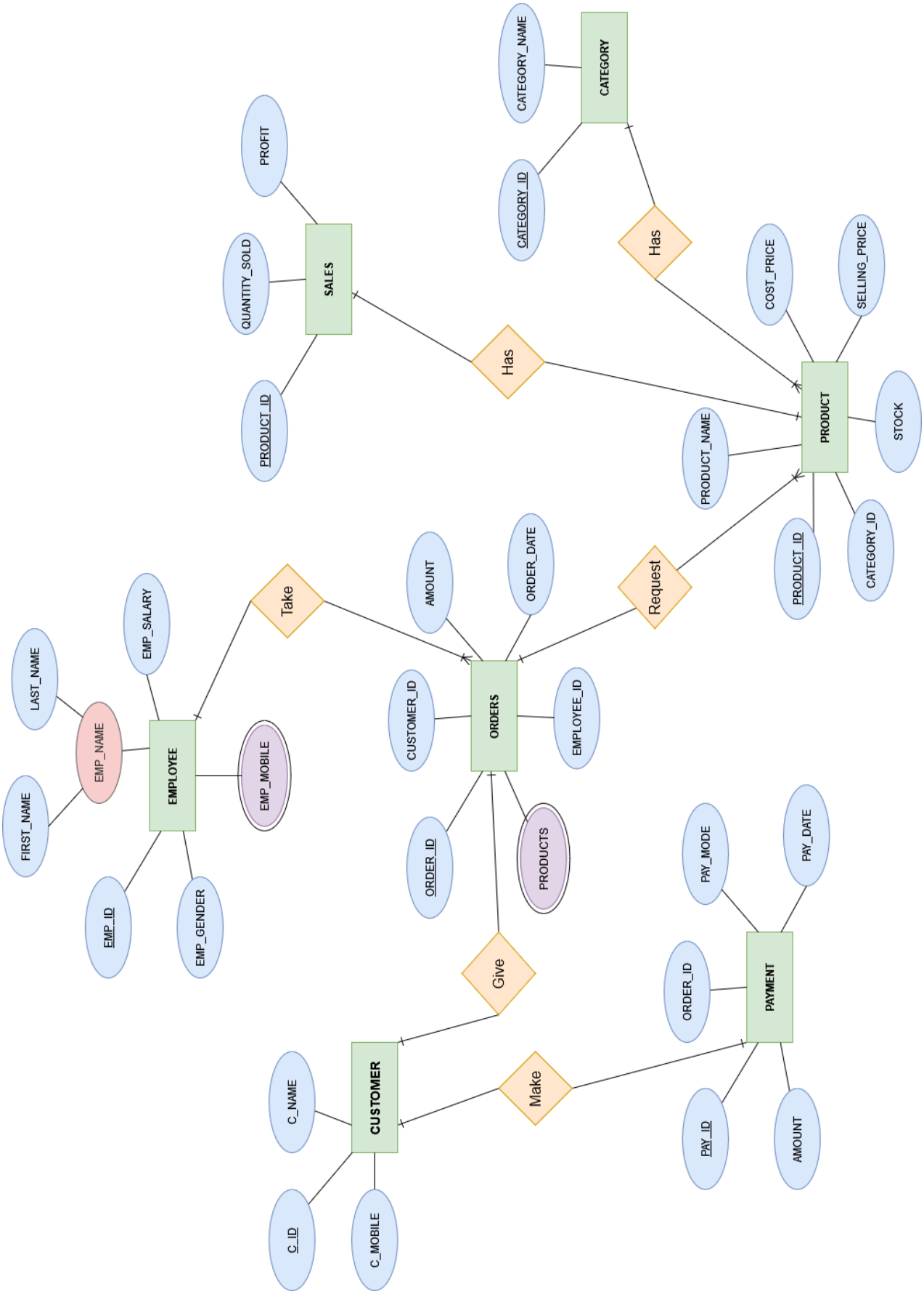
7. **PAYMENT:**

   PAY_ID -> ORDER_ID, PAY_MODE, AMOUNT, PAY_DATE

8. **SALES:**

   PRODUCT_ID -> QUANTITY_SOLD, PROFIT

ENTITY RELATIONSHIP DIAGRAM



NORMALIZATION

1st Normal Form

First Normal Form (**1NF**) requires that each column in a table should contain atomic values, meaning that a column should only contain one value for each record, and that value should not be divisible further.

Our Schema contained two **multivalued attributes** EMPLOYEE (EMP_MOBILE) & ORDRS (PRODUCTS). So, we created two more relations EMP_MOBILE & ORDER_ITEMS.

Our Schema contained a **composite attributes** EMPLOYEE (EMP_NAME). So, we divided that into two attributes FIRST_NAME, LAST_NAME.

2nd Normal Form

2nd Normal Form (**2NF**) is a database normalization stage that ensures all non-prime attributes (attributes not part of the primary key) are fully functionally dependent on the primary key and no partial dependencies exist.

Our Schema is already in 2NF, because all tables have a single attribute primary key or a composite primary key, ensuring each row is uniquely identifiable.

Non-prime attributes in each table are fully functionally dependent on the primary key, and no partial dependencies are present. For example, in the PRODUCT table, all **non-prime attributes** (PRODUCT_NAME, CATEGORY_ID, COST_PRICE, SELLING_PRICE, STOCK) depend on the primary key (PRODUCT_ID) and not just a part of it.

3rd Normal Form

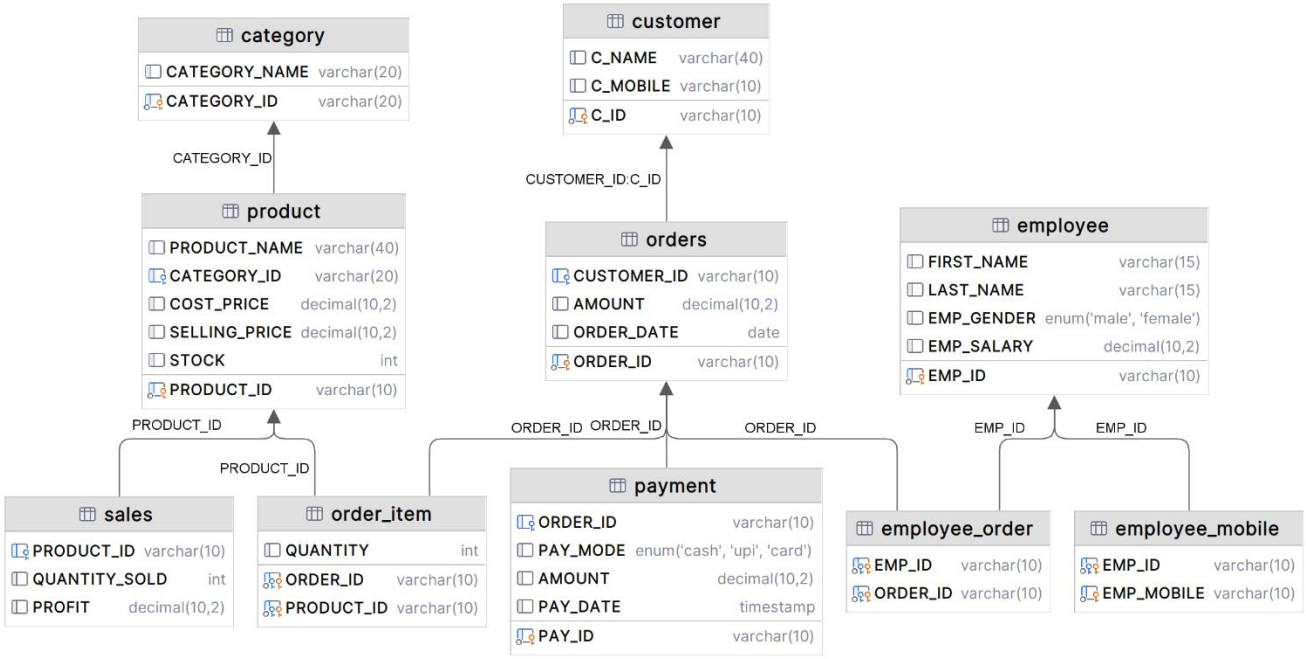
3rd Normal Form (**3NF**) is a database normalization stage that ensures a table is in 2NF and, additionally, all non-prime attributes are non-transitively dependent on the primary key. In other words, there should be no functional dependencies between non-prime attributes.

Our Schema is already in 3NF, because all tables are in 2NF, as previously mentioned.

Non-prime attributes in each table are directly dependent on the primary key, and there are **no transitive dependencies** between non-prime attributes. For example, in the PRODUCT table, non-prime attributes like COST_PRICE and SELLING_PRICE depend only on the primary key (PRODUCT_ID) and not on any other non-prime attribute.

By satisfying these conditions, the schema adheres to the principles of 3rd Normal Form, further reducing redundancy and ensuring that the data is free of insertion, update, and deletion anomalies.

NORMALIZED RELATIONAL MODEL



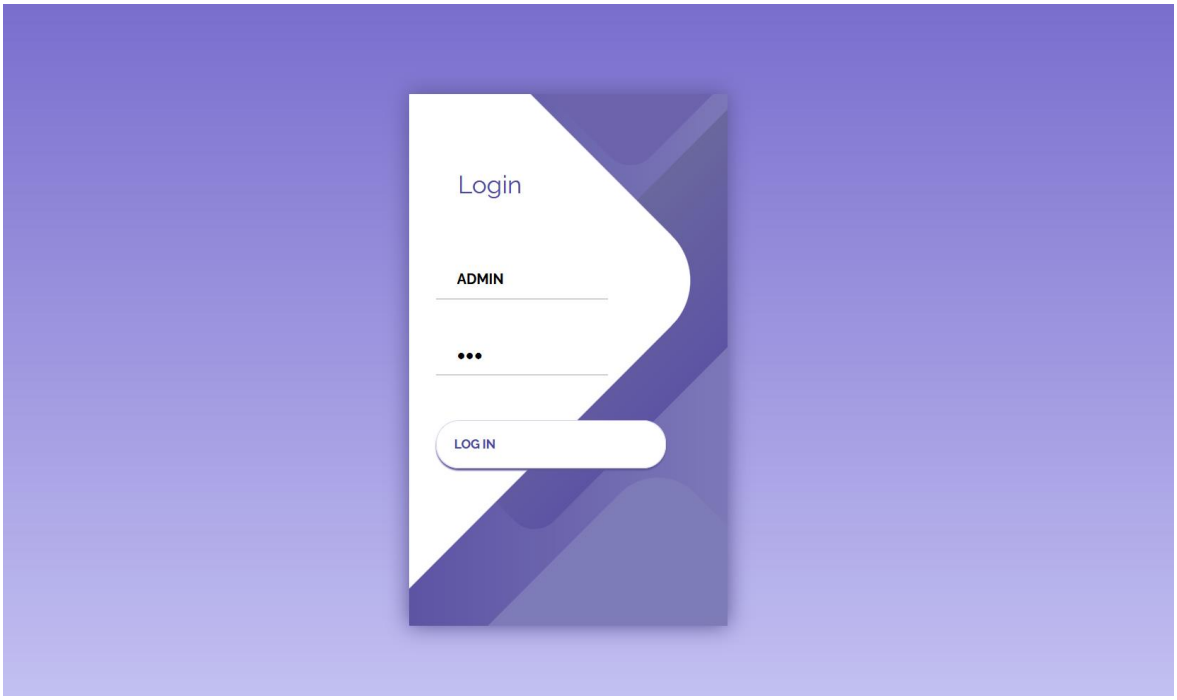
- **Customer** (C_Id, C_Name, C_Mobile)
- **Category** (Category_Id, Category_Name)
- **Product** (Product_Id, Product_Name, Category_Id, Cost_Price, Selling_Price, Stock)
- **Employee** (Emp_Id, First_Name, Last_Name, Emp_Gender, Emp_Salary)
- **Orders** (Order_Id, Customer_Id, Amount, Order_Date)
- **Order_Item** (Orser_Id, Product_Id, Quantity)
- **Payment** (Pay_Id, Order_Id, Pay_Mode, Amount, Pay_Date)
- **Sales** (Product_Id, Quantity_Sold, Profit)
- **Employee_Mobile** (Emp_Id, Emp_Mobile)
- **Employee_Order** (Emp_Id, Order_Id)

TECH STACKS USED

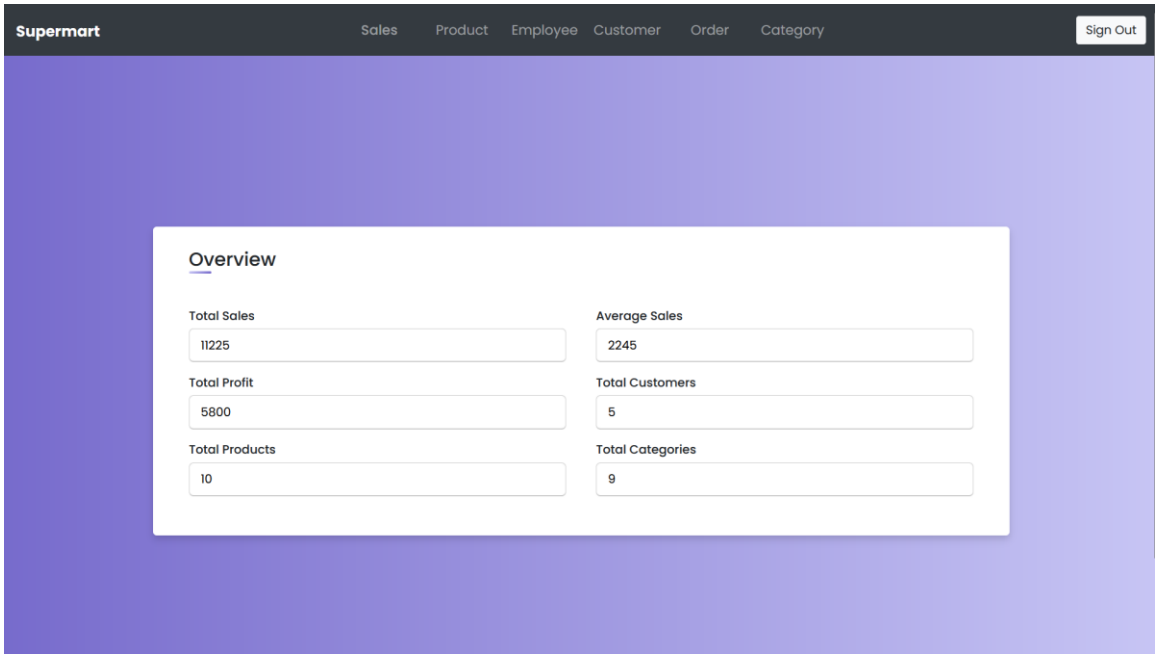
- MySQL
- NodeJS
- EJS
- CSS
- Nodemon
- Express
- Body Parser
- Visual Studio Code
- MySQL Workbench 8.0 CE

FRONT END INTERFACE

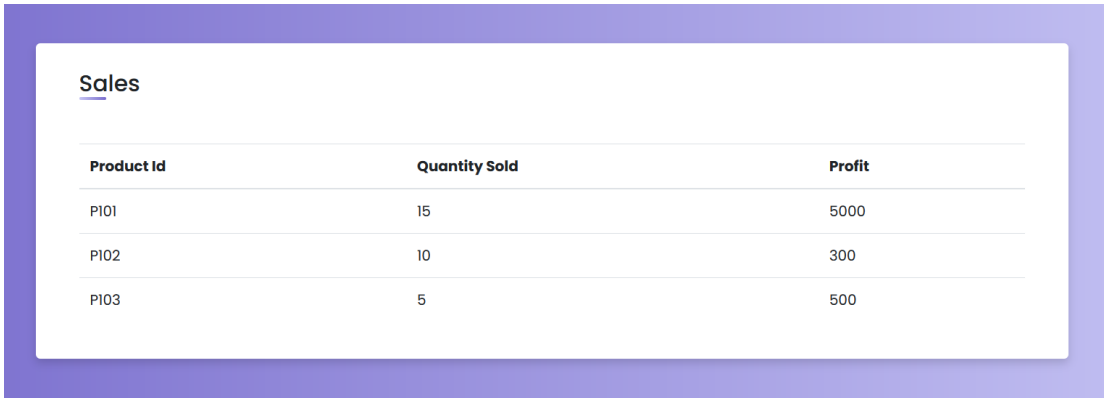
➢ Login Page



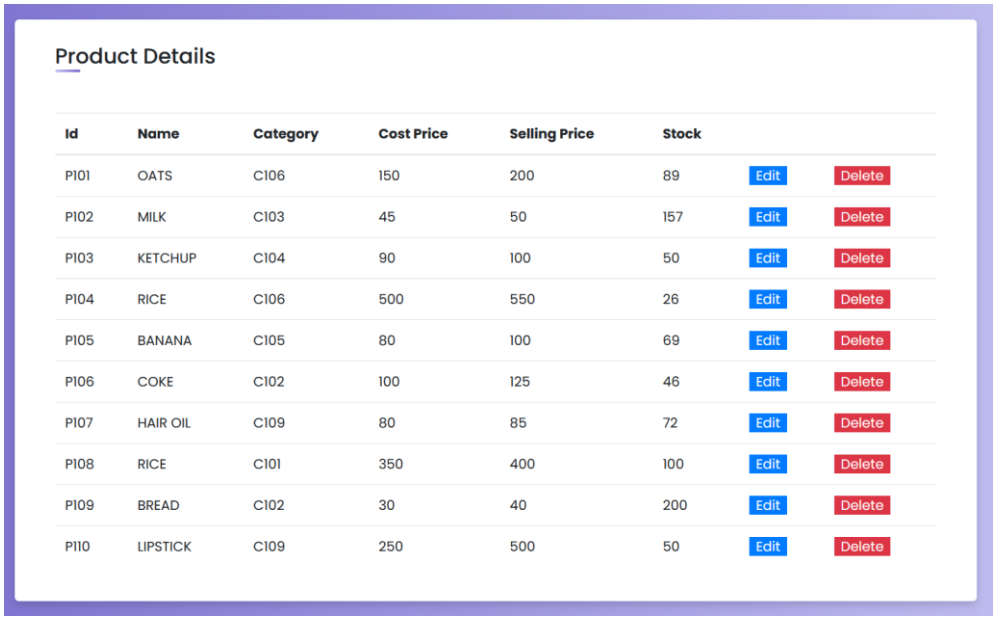
➢ Dashboard



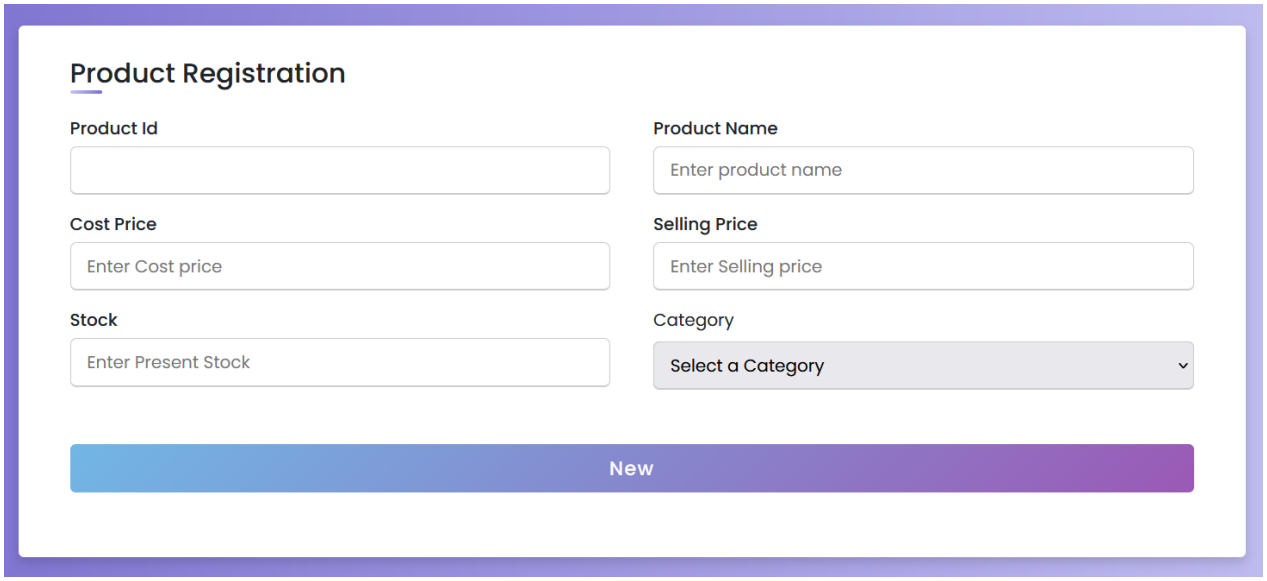
➢ Sales



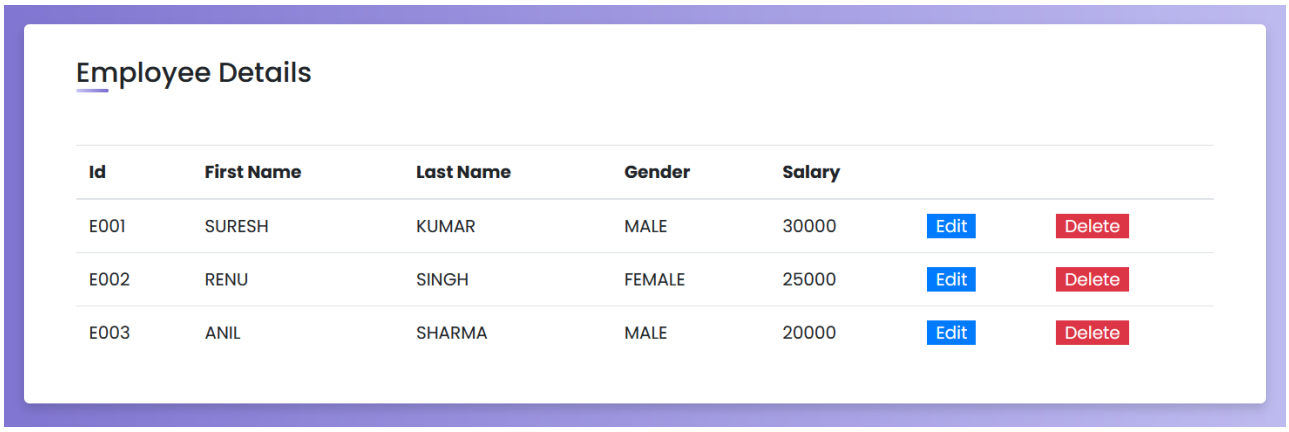
➢ Products



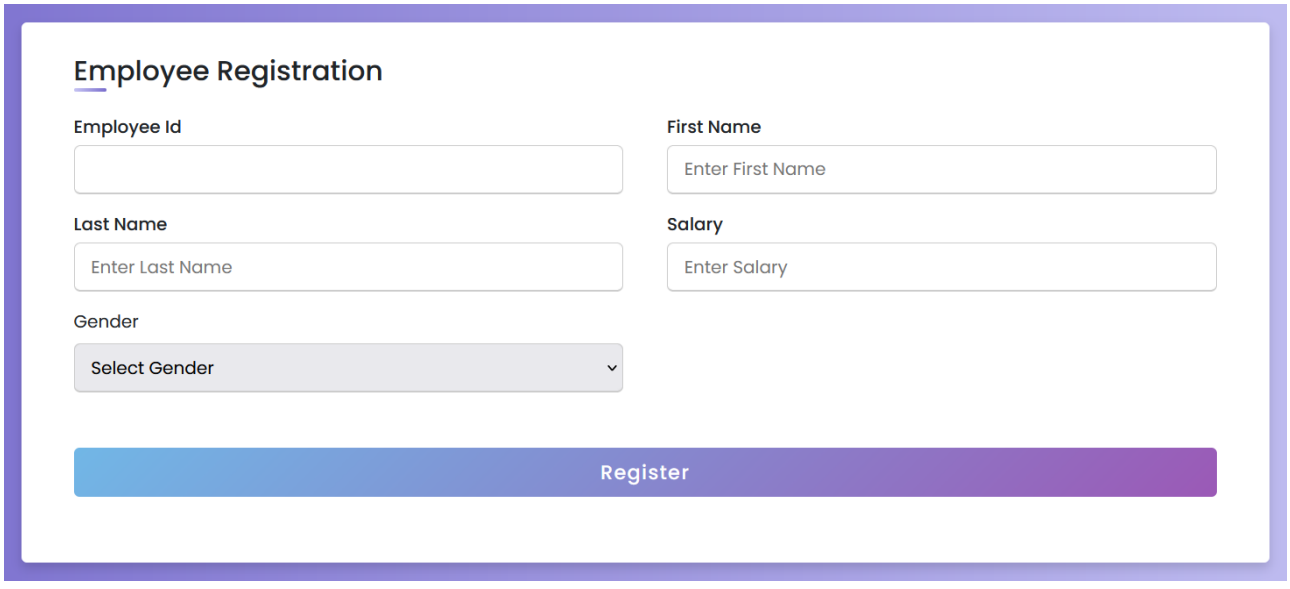
➢ New Product



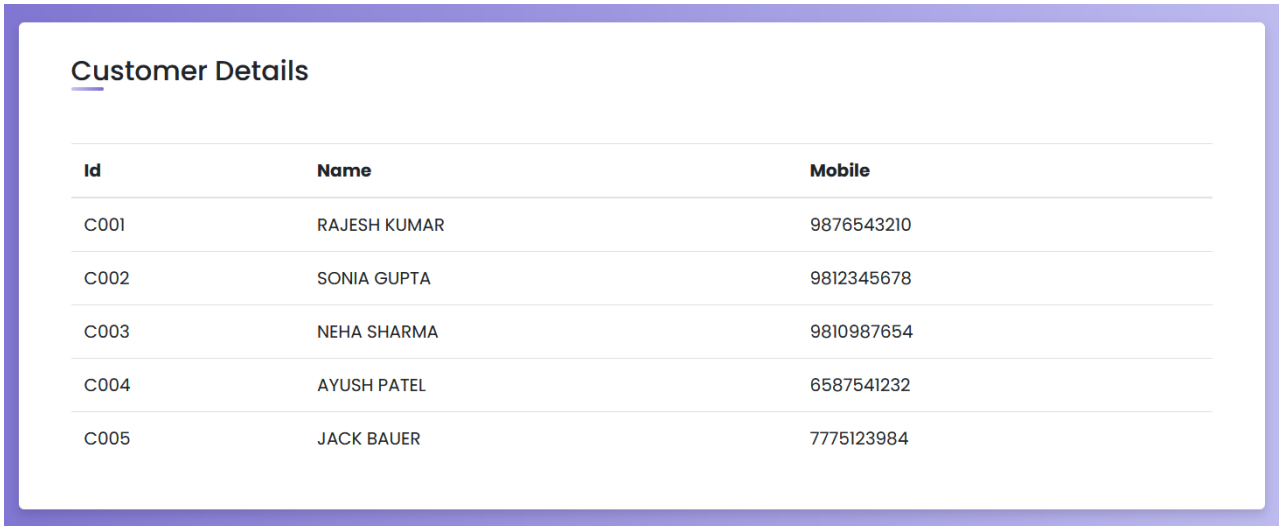
➢ Employees



➢ New Employee



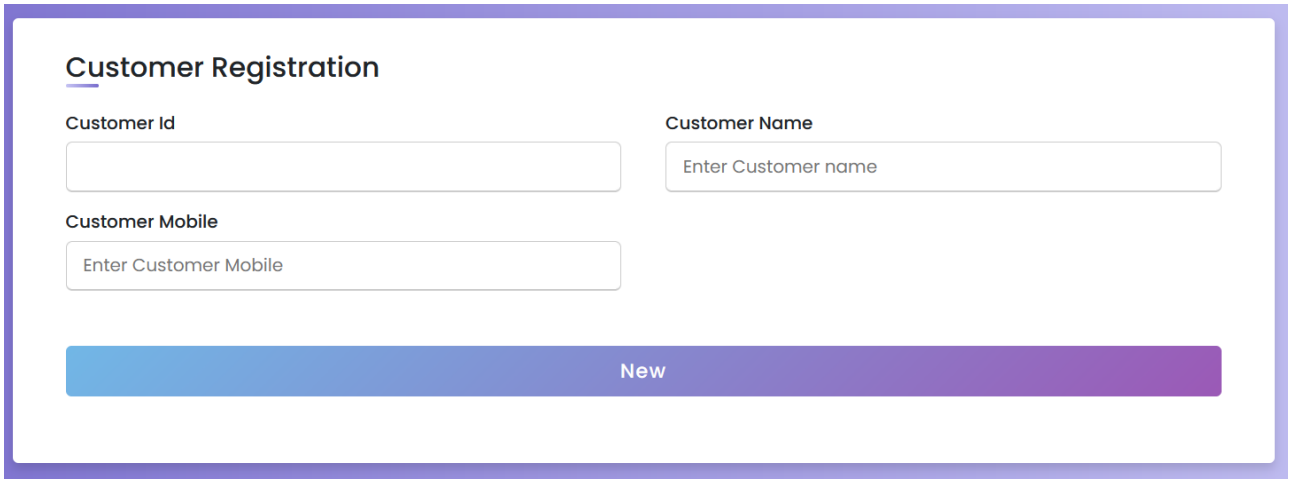
➢ Customers



Customer Details

Id	Name	Mobile
C001	RAJESH KUMAR	9876543210
C002	SONIA GUPTA	9812345678
C003	NEHA SHARMA	9810987654
C004	AYUSH PATEL	6587541232
C005	JACK BAUER	7775123984

➢ New Customer



Customer Registration

Customer Id

Customer Name

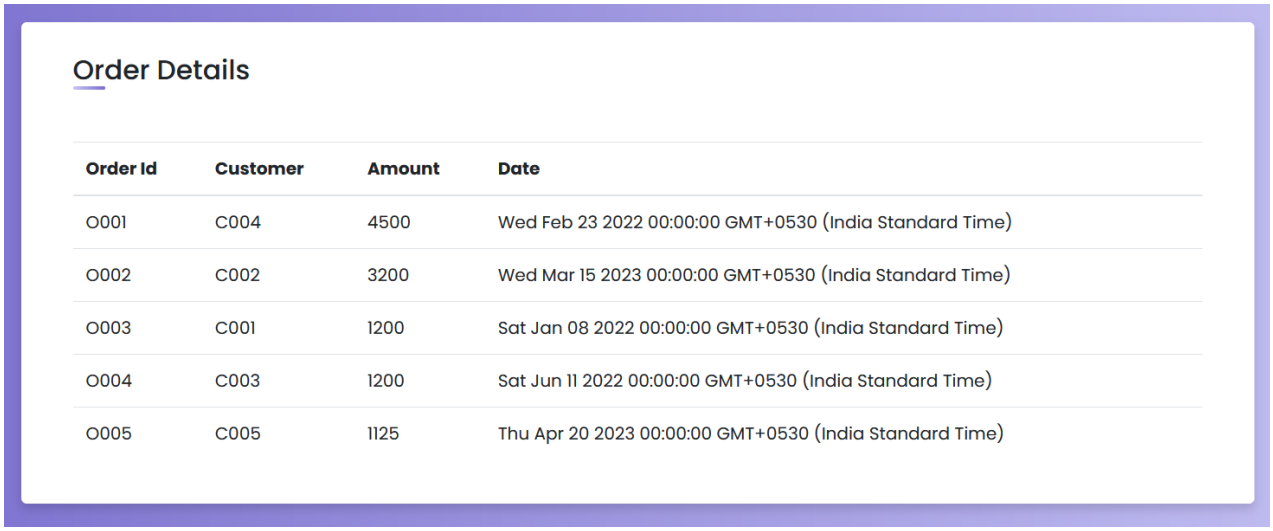
Enter Customer name

Customer Mobile

Enter Customer Mobile

New

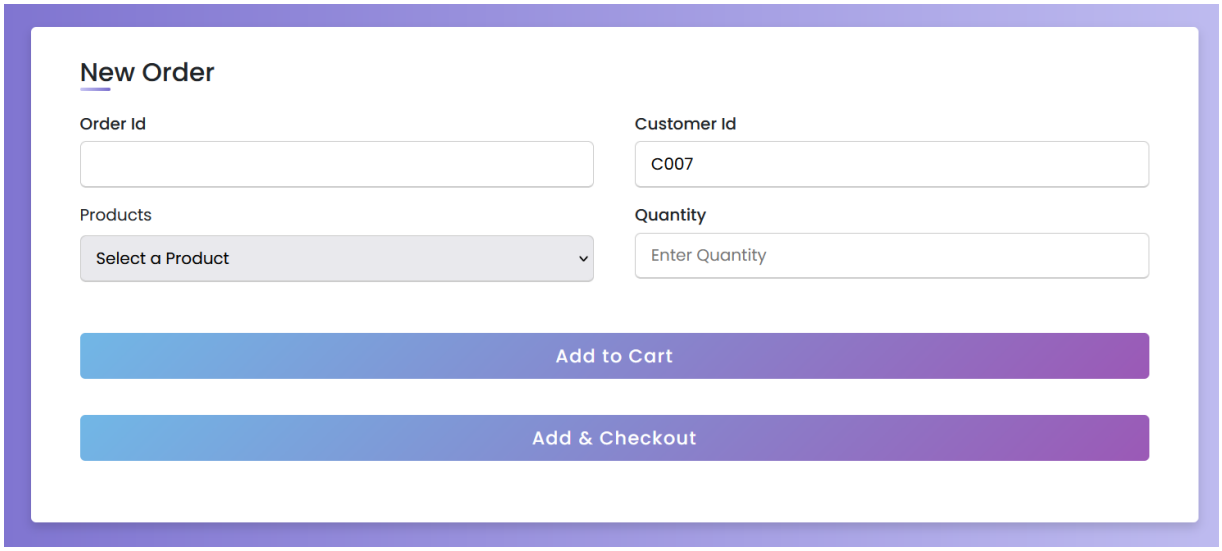
➢ Orders



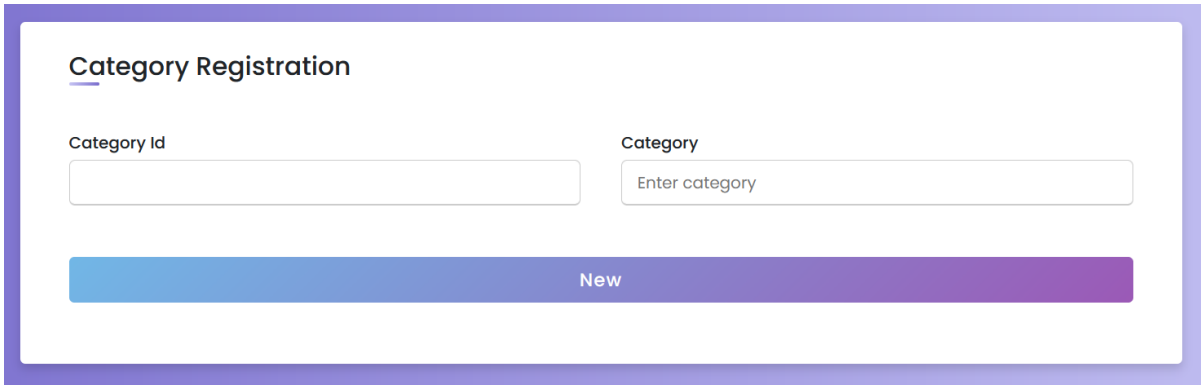
Order Details

Order Id	Customer	Amount	Date
O001	C004	4500	Wed Feb 23 2022 00:00:00 GMT+0530 (India Standard Time)
O002	C002	3200	Wed Mar 15 2023 00:00:00 GMT+0530 (India Standard Time)
O003	C001	1200	Sat Jan 08 2022 00:00:00 GMT+0530 (India Standard Time)
O004	C003	1200	Sat Jun 11 2022 00:00:00 GMT+0530 (India Standard Time)
O005	C005	1125	Thu Apr 20 2023 00:00:00 GMT+0530 (India Standard Time)

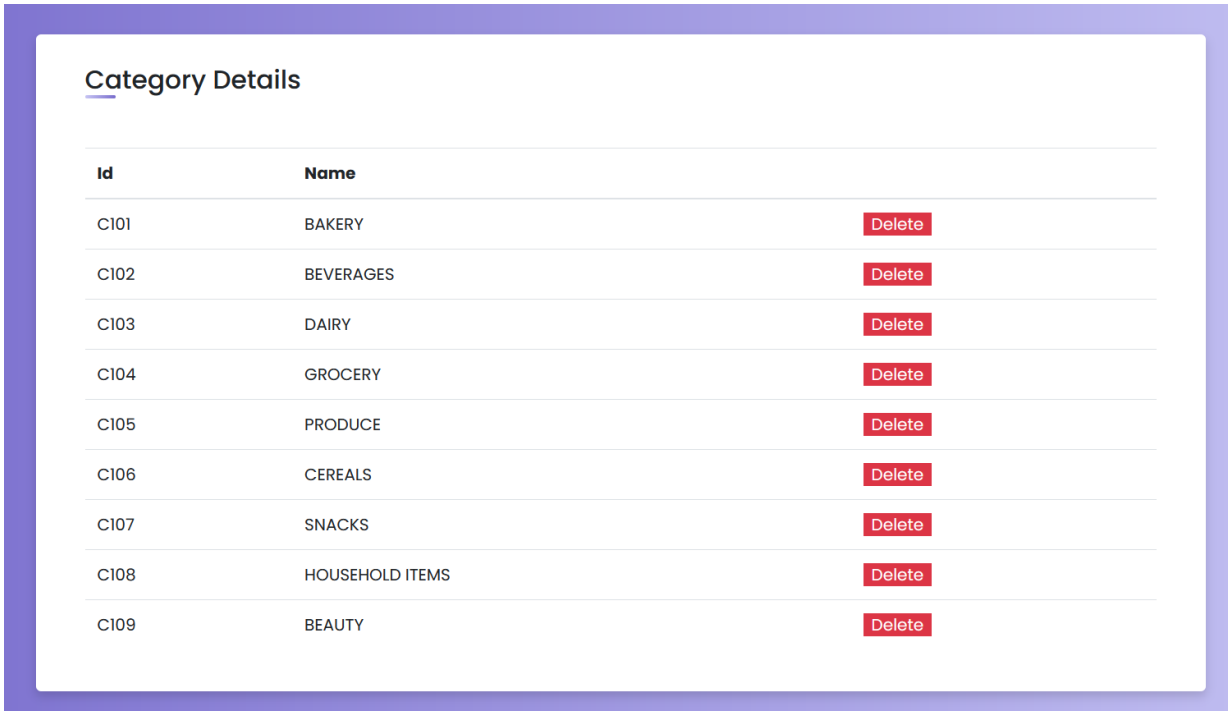
➢ New Order



➢ New Categories



➢ Categories



Category Details

Id	Name	
C101	BAKERY	Delete
C102	BEVERAGES	Delete
C103	DAIRY	Delete
C104	GROCERY	Delete
C105	PRODUCE	Delete
C106	CEREALS	Delete
C107	SNACKS	Delete
C108	HOUSEHOLD ITEMS	Delete
C109	BEAUTY	Delete

BACK-END CONNECTIVITY

```
var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "pavilion",
  database: "SUPERMART",
  insecureAuth: true,
});

connection.connect(function (err) {
  if (err) throw err;
  console.log("MySQL Connected!");
});
```

SAMPLE MYSQL QUE RIES IN NODEJS

1) INSERT

```
var sql =
    "INSERT INTO PRODUCT VALUES (?, ?, ? , " +
    productCostPrice +
    ", " +
    productSellingPrice +
    ", " +
    productStock +
    ");";
  connection.query(
    sql,
    [id, productName, productCategory],
```

```
  function (err, data) {
    if (err) throw err;
    console.log(data.affectedRows + " record(s) updated");
    res.redirect("/products");
  }
);
```

2) SELECT

```
var sql = "SELECT * FROM product ORDER BY PRODUCT_ID";
 connection.query(sql, function (err, productData, fields) {
  if (err) throw err;
  res.render("products", { title: "Products", productData: productData });
 });
```

3) UPDATE

```
var sql =
  "UPDATE PRODUCT SET PRODUCT_NAME = ?, CATEGORY_ID = ?,
COST_PRICE = " +
  editedCostPrice +
  " ,SELLING_PRICE = " +
  editedSellingPrice +
  " ,STOCK = " +
  editedStock +
  " WHERE PRODUCT_ID = ?;";
 connection.query(sql, [editedName, editedCategory, id], function (err, data) {
  if (err) throw err;
  console.log(data.affectedRows + " record(s) updated");
  res.redirect("/products");
 });
```

4) DELETE

```
var sql = "DELETE FROM product WHERE PRODUCT_ID = ?";
  connection.query(sql, [req.params.deleteId], function (err, data) {
    if (err) throw err;
    console.log(data.affectedRows + " record(s) deleted");
    res.redirect("/products");
  });
```

CONCLUSION

In conclusion, the development of a supermarket management system is a significant step towards improving the efficiency and productivity of supermarkets. The project aimed to design and implement a software system that would automate various supermarket operations, such as inventory management, sales tracking, and customer management.

Throughout the project, we identified the key requirements of the system, designed the architecture, implemented the functionality, and tested the system thoroughly to ensure its reliability and accuracy. We also ensured that the system could adapt to the changing needs of the supermarket and its customers.

The successful implementation of the supermarket management system offers several benefits, including improved inventory management, accurate sales tracking, better customer experience, and increased profitability. It also eliminates the need for manual processes, reducing the chances of errors and saving time.

Overall, the supermarket management system represents a significant technological advancement for supermarkets, providing them with a competitive edge in the market. The project has been a great learning experience, allowing us to apply our knowledge and skills in software development, and we are confident that it will bring tangible benefits to supermarket owners, employees, and customers.