Singh Ayush Kumar Satish || 2020133
Naman Kaushik || 2020088
Tarushi Gandhi || 2020579
Apoorva Arya || 2020032

Problem Statement

In the fast-paced lifestyles that we see in most of the metro and urban cities today, most people find it difficult to take out time to go shopping.
On the other hand, people in rural areas do not get access to a large number of brands in their neighborhoods.
Online retail stores provide a one-stop solution for both these issues by allowing consumers to buy their favorite products online without the hassle of commuting.

Scope of Project

We aim to provide a rich consumer experience for the online audience by providing a database to online retail stores, which they can use to increase their efficiency. We help Online Retail Stores to manage their operations by providing them with an efficient Database which keeps a record of the employees working for the organization, the suppliers, they get their products from and the type of products they receive. The database also stores all necessary information about registered customers, available Products with the finest details like Cost, Quantity, etc. We help them manage their supply chain by keeping record of the products that have been supplied and the products that are ordered by the end consumer. When a consumer orders a product, a delivery person is assigned to the order who can update the status of delivery. We aim to help Online Retail Stores to manage their operations comfortably by providing them with an efficient Database.
Our Database keeps records of all the necessary information required to run a retail store in an efficient manner such as :
1. Organizational Details(Employees, Departments, Delivery Locations).
2. Customer Details
3. Product Supplier Detail
4. Product Details, etc.

Stakeholders
<u>Consumers:</u>
Role - They are the people who will be ordering products from the retail store, adding payment information and registering themselves and providing their information such as an address, phone number etc.
● Registering themselves/logging in
● Adding items to the cart
● Placing an order
● Providing feedback
● Can raise a complaint against a product
● Can utilize coupons to afford discounts.

Inventory Managers:
Role - They are responsible for keeping track of the available products within the store.
- Adding items after buying from a supplier
- Reduction of the quantity of the products upon purchases
- Can update the cost and details of products supplied by supplier and add new suppliers.

Delivery Supervisor:
They are responsible for the timely delivery of goods to the consumers' door
- Can update the status of delivery
- Can check the delivery address and contact information of the consumers.

HR:
- Can hire a new employee into the Institution and update the Employee Table.

Advertisement:
- They can view all the data related to customers except their password for Advertisement purposes.

Customer Care:
- Can view, and update the complaints table.

General Employee:
- Can view, and update tables like Products and Coupons.

Employees:
They are responsible for maintaining the records and handling the consumer's complaints and feedback.
- Can handle complaints registered by consumers and resolve them on time.
- Can add and update the employee details.
- Can disable the coupons after a stipulated time.

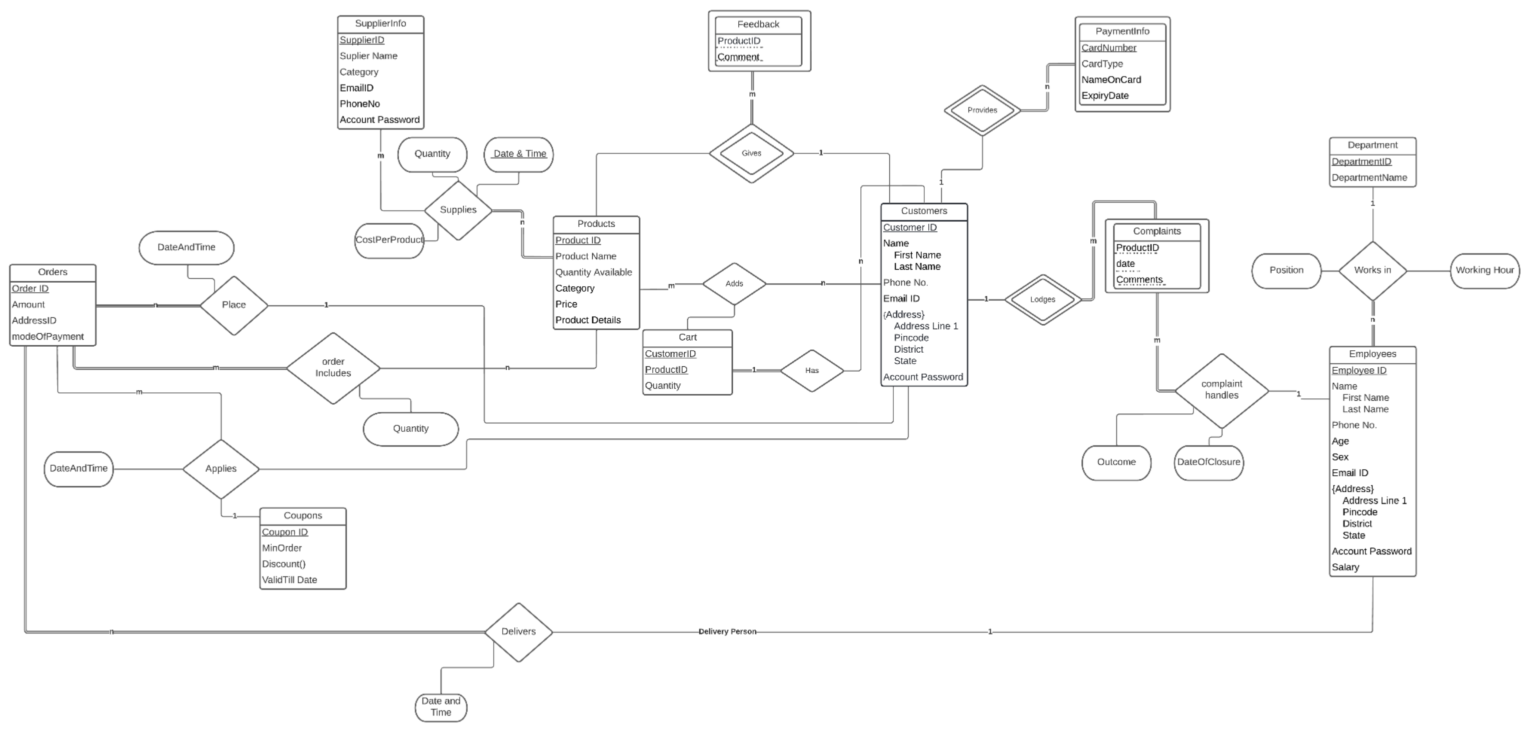
Relational Schema:

Customers(customerID, firstName, lastName, phoneNo, emailID, accountPassword)
Products(productID, productName, quantityAvailable, category, price, productDetails)
Cart(customerID, productID, quantity)
Orders(orderID, amount, addressID, modeOfPayment)
orderIncludes(orderID, productID, quantity)
Coupons(couponID, minOrder, validTillDate, discount)
Employees(employeeID, firstName, lastName,phoneNo, age, gender, emailID, accountPassword, Salary)
address(addressID, customerID, addressLine1, pincode)
employeeAddress(addressID, employeeID, addressLine1,pincode)
paymentInfo(cardNo, customerID, cardType, nameOnCard, expiryDate)
place(orderID, customerID, dateAndTime)

complaintHandles(customerID, productID, comments, complaintDate, employeeID, outcome, dateOfClosure)
delivers(orderID, employeeID, dateAndTime)
worksIn(employeeID, departmentID, position, workingHours)
applies(orderID, couponID)
departments(departmentID, departmentName)
supplierInfo(supplierID, supplierName, category, emailID, phoneNo, accountPassword)
supplies(supplierID, productID, dateAndTime, costPerProduct, quantity)
feedback(productID, customerID, comments)
Complaints(customerID, productID, comments, complaintDate)

Legend:
Primary Key
Foriegn Key

ER Diagram



Indexes

create index **productCategoryIndex** on Products (category);
create index **WorksInPostIndex** on worksIn(position);
create index **SupplierInfoCategoryIndex** on supplierInfo(category);
create index **modeOfPayIndex** on orders(modeOfPayment);
create index **productFeedbackIndex** on feedback(productID);

create index **employeeIndex** on complainthandles(employeeID);
create index **categoryIndex** on products (category);
create index **empName** on employees(firstName, lastName);
create index **customerLastNameIndex** on Customers (lastName);
create index **employeeLastName_AgeIndex** on Employees(lastName, age);
create index **employeeAddress_PinCode_District_StateIndex** on employeeAddress(pincode, district, state);
create index **address_PinCode_District_StateIndex** on address(pincode, district, state);
create index **OrderAddressIDIndex** on Orders(addressID);
create Index **PaymentInfoCaredType** on paymentInfo(cardType);

‘Unique’ constraint

Customers(customerID, phoneNo, emailID)
Employees(employeeID,phoneNo, emailID)
supplierInfo(supplierID, emailID, phoneNo)
Products(productID)
Cart(customerID, productID)
Orders(orderID)
Coupons(couponID)
address(addressID)
employeeAddress(addressID)
paymentInfo(cardNo, customerID)
place(orderID)
complaintHandles(customerID)
delivers(orderID)
worksin(employeeID)
applies(orderID)
departments(departmentID)
supplies(supplierID, productID, dateAndTime)
feedback(productID, customerID, comments)
Complaints(customerID, productID, comments, complaintDate)
orderIncludes(orderID, productID)

Check constraints
Customers(phoneNo) check(phoneNo>6000000000)
Products(category) check (category in ('Eatables', 'Apparels', 'Electronics', 'Furniture'))
Cart(quantity) check(quantity > 0)
orders(amount) check(amount > 0)
orders(modeOfPayment) check(modeOfPayment in ('COD', 'Online'))
coupons(minOrder) check(minOrder > 800)
coupons(discount) check (discount between 150 and 250)
employees(phoneNumber) check(phoneNumber>6000000000)
employees(gender) check (gender in ('Male', 'Female', 'Transgender', 'Non binary','Others'))
employees(salary) check(salary>10000)

address(pincode) check(pincode > 0)
employeeAddress(pincode) check(pincode > 0)
paymentInfo(cardNo) check(cardNo>=0)
delivers(dateAndTime) check(dateAndTime >= sysdate()))
worksIn(workinghours) check(workinghours > 0)
supplierInfo(category) check(category in ('Eatables', 'Aparrels', 'Electronics', 'Furniture'))
supplies(costPerProduct) check(costPerProduct > 0)
supplies(quantity) check(quantity > 0)
orderIncludes(quantity) check(quantity > 0)

Not null constraints

Customers(customerID, firstName, lastName, phoneNo, emailID, accountPassword)
Products(productID, productName, productDetails)
Cart(customerID, productID, quantity)
Orders(orderID, amount, addressID, modeOfPayment)
Coupons(couponID, minOrder, validTillDate, discount)
Employees(employeeID, firstName,phoneNo, age, gender, emailID, accountPassword, Salary)
address(addressID, addressLine1, pincode, district, state)
employeeAddress(addressID, addressLine1,pincode, district, state)
paymentInfo(cardNo, customerID, cardType, nameOnCard, expiryDate)
place(orderID)
complaintHandles(customerID)
delivers(orderID)
worksin(employeeID, position, workingHours)
applies(orderID)
departments(departmentID, departmentName)
supplierInfo(supplierID, supplierName, category, emailID, phoneNo, accountPassword)
supplies(supplierID, productID, dateAndTime, costPerProduct, quantity)
feedback(productID, customerID, comments)
Complaints(customerID, productID, comments, complaintDate)
orderIncludes(orderID, productID, quantity)

GRANTS/AUTHORIZATIONS

<u>Authorization Customer:</u>

create user 'ayush_gmail.com'@'localhost' identified by 'ayush';
grant update on customers to 'ayush_gmail.com'@'localhost';
grant select on products to 'ayush_gmail.com'@'localhost';
grant insert, select, delete on cart to 'ayush_gmail.com'@'localhost';
grant insert on feedback to 'ayush_gmail.com'@'localhost';
grant insert on complaints to 'ayush_gmail.com'@'localhost';
grant insert, delete, select on address to 'ayush_gmail.com'@'localhost';
grant insert, delete, select on paymentinfo to 'ayush_gmail.com'@'localhost';

```
grant insert, select on orders to 'ayush_gmail.com'@'localhost';
grant insert, select on orderincludes to 'ayush_gmail.com'@'localhost';
grant select on coupons to 'ayush_gmail.com'@'localhost';
grant execute on procedure apply_Coupon to 'ayush_gmail.com'@'localhost';
grant execute on procedure getAddressList to 'ayush_gmail.com'@'localhost';
grant execute on procedure getOrderList to 'ayush_gmail.com'@'localhost';
grant select, insert on place to 'ayush_gmail.com'@'localhost';
grant execute on function getOrderAmount to 'ayush_gmail.com'@'localhost';
```

Authorization Supplier:

```
create user 'tarushi_gmail.com'@'localhost' identified by 'tarushi';
grant select, insert on products to 'tarushi_gmail.com'@'localhost';
grant insert on supplies to 'tarushi_gmail.com'@'localhost';
grant execute on procedure categoryProductList to 'tarushi_gmail.com'@'localhost';
```

General Employee/CustomerCare:

```
create user 'naman_gmail.com'@'localhost' identified by 'naman';
create role 'generalEmployee';
grant ‘generalEmployee’ to 'naman_gmail.com'@'localhost';
grant select on customerg_employee_view to 'generalEmployee';
grant select on supplierg_employee_view to 'generalEmployee';
grant select on customerg_employee_view to 'generalEmployee';
grant select on complaints to 'generalEmployee';
grant insert on complaintHandles to 'generalEmployee';
grant select on feedback to 'generalEmployee';
```

HR:

```
create user 'apoorva_gmail.com'@'localhost' identified by 'apoorva';


grant select on customerg_employee_view to 'apoorva_gmail.com'@'localhost';
grant select on supplierg_employee_view to 'apoorva_gmail.com'@'localhost';
grant select on customerg_employee_view to 'apoorva_gmail.com'@'localhost';

grant select on complaints to 'apoorva_gmail.com'@'localhost';
grant insert on complaintHandles to 'apoorva_gmail.com'@'localhost';
grant select on feedback to 'apoorva_gmail.com'@'localhost';
grant select on place to 'apoorva_gmail.com'@'localhost';
grant select on orders to 'apoorva_gmail.com'@'localhost';
grant select on supplies to 'apoorva_gmail.com'@'localhost';
grant select on products to 'apoorva_gmail.com'@'localhost';
```

grant select on orderincludes to 'apoorva_gmail.com'@'localhost';
grant select on applies to 'apoorva_gmail.com'@'localhost';
grant select on coupons to 'apoorva_gmail.com'@'localhost';
grant select on customers to 'apoorva_gmail.com'@'localhost';
grant insert,delete, update on employees to 'apoorva_gmail.com'@'localhost';
grant insert,delete on worksin to 'apoorva_gmail.com'@'localhost';
grant select on customerg_employee_view to 'apoorva_gmail.com'@'localhost';
grant select on supplierg_employee_view to 'apoorva_gmail.com'@'localhost';

<u>Delivery:</u>
create user 'vaibhav'@'localhost' identified by 'vaibhav';
create role 'deliveryPerson';
grant 'deliverPerson' to 'vaibhav'@'localhost';
grant select, update on delivers to 'deliveryPerson';
grant select on orders to 'deliveryPerson';

VIEWS

1. create view **CustomerG_Employee_View** (customerID, firstName, lastName, phoneNo, emailID) as
        select customerID, firstName, lastName, phoneNo, emailID
    from Customers;

2. create view **EmployeeG_Employee_View** (employeeID, firstName, lastName, phoneNo, age, gender, emailID) as
        select employeeID, firstName, lastName, phoneNumber, age, gender, emailID
    from Employees;

3. create view **SupplierG_Employee_View** (supplierID, supplierName, category, emailID, phoneNo) as
        select supplierID, supplierName, category, emailID, phoneNo
    from supplierInfo;

TRIGGERS

1. Update Product Quantity after getting Supplies
delimiter //
create trigger update_product_quantity

```
        after insert on supplies for each row
   begin
   update products
   set quantityAvailable = quantityAvailable + new.quantity
   where productID = new.productID;
   end//

delimiter ;
```

2. To add into place table after order has been placed

```
delimiter //
    create trigger add_into_place
                after insert on orders for each row
                begin
                declare currentUser varchar(50);
      declare customerID int;
      select substring_index(user(), '@', 1) into currentUser;
      select customers.customerID into customerID
      from customers
      where customers.emailID = currentUser;
      insert into place(orderID, dateAndTime, customerID)
      values
      (new.orderID, sysdate(), customerID);
      end//

      delimiter ;
```

3. Update Product Quantity after Order:

```
delimiter //
create trigger update_Quantity
        before delete on cart for each row
   begin
                update products
                set quantity = quantity - old.quantity
      where products.productID = old.productID;
        end //
delimiter ;
```

4. Add to Complaint Table

```
delimiter //
        create trigger add_to_complaint_handles
   after insert on complaints for each row
```

```
    begin
    insert into complaintHandles (customerID, productID, comments, complaintDate) values
(new.customerID, new.productID, new.comments, new.complaintDate);
    end //
delimiter ;
```

5. Add to Delivers Table:

```
delimiter //
create trigger add_to_delivers
after insert on orders for each row
        begin
                insert into delivers(orderID) values (new.orderID);
        end //
delimiter ;
```

QUERIES

1. **List all unused Coupons:**

```
select couponID, minOrder, validTillDate, discount
from coupons natural left outer join applies
where orderID is null and validTillDate >= curdate();
```

2. **List all Customers who have placed an order at least once but haven't bought anything from a particular Category:**

```
create procedure get_category_negative_customers(category varchar(20))
begin
select customerID, firstName, lastName
from customers
where not exists(select productID
                        from products left join (select orderincludes.productID
                                                from orderincludes natural join place
                                                where place.customerID = customers.customerID)
                        as productsBought(productID) using (productID)
                        where products.category = category and productsBought.productID = null);
end//
```

3. **Verify Coupon and if Valid return Discount:**

```
create procedure apply_Coupon(in couponID int, in totalAmount int, in currentDate date, out isValid
int , out orderDiscount float)
        begin
   declare min_Order float;
   declare validTill_Date date;
   select count(couponID), minOrder, validTillDate, discount into isValid, min_Order, validTill_Date,
orderDiscount
   from coupons
   where coupons.couponID = couponID;
   if isValid = 1 and totalAmount < min_Order or currentDate > validTill_Date then
                set orderDiscount = 0;
        end if;
   end
```

4. **List all Coupons which were used at least twice.**

```
select couponID
from coupons
where 2 <= (select count(couponID)
                from applies
                where coupons.couponID = applies.couponID);
```

5. **Rank Employees based on Complaints Resolved**

```
select employeeID, rank() over (order by count(dateOfClosure)) as employeeRank,
count(dateOfClosure) as complaintsHandle
from complaintshandles
where dateOfClosure is not null
group by employeeID;
```

6. **Average spending Across each OrderID**

```
select orderID as OrderNo, avg(amount)
over (order by orderID rows unbounded preceding) as AvgSale
from orders;
```

7. **Get product with most Number of Complaints**

```
create procedure mostComplaint_product(category varchar(20))
begin
        select complaints.productID, products.category
        from complaints natural join products
        where products.category = category
        group by complaints.productID
        order by count(*) desc
        limit 1;
end
```

8. **Customers who have used both COD and Card as mode of payment.**

```
select P.customerID
from place P, orders O
where P.orderID = O.orderID AND O.modeOfPayment = ‘COD’ AND P.customerID IN (
select P.customerID
from place P, orders O
where P.orderID = O.orderID AND O.modeOfPayment = ‘Online’);
```

9. **Rank Employees based on Complaints Resolved**

```
select employeeID, rank() over (order by count(dateOfClosure)) as employeeRank,
count(dateOfClosure) as complaintsHandle
from complaintshandles
where dateOfClosure is not null
group by employeeID;
```

10. **Average spending Across each OrderID**

```
select orderID as OrderNo, avg(amount) as Average
over (order by orderID rows unbounded preceding) as AvgSale
from orders;
```

Embedded Queries:

1. **customer wants to see cart total:**

```
with product_list (productID, price) as
         (select productID, price
         from products)
select sum(price * quantity) into orderAmount
from orderincludes join product_list using (productID);
```

Get Category Wise Profit

```
with category_purchase (category, purchaseAmount) as
        (select category, sum(supplies.quantity*costPerProduct)
        from supplies natural join products
   group by products.category
   )
select category, category_sold.sellingAmount - category_purchase.purchaseAmount as profit
from (select category, sum(orderincludes.quantity*price)
        from orderincludes natural join products
```

```
    group by products.category) as category_sold(category, sellingAmount) natural join
category_purchase;
```

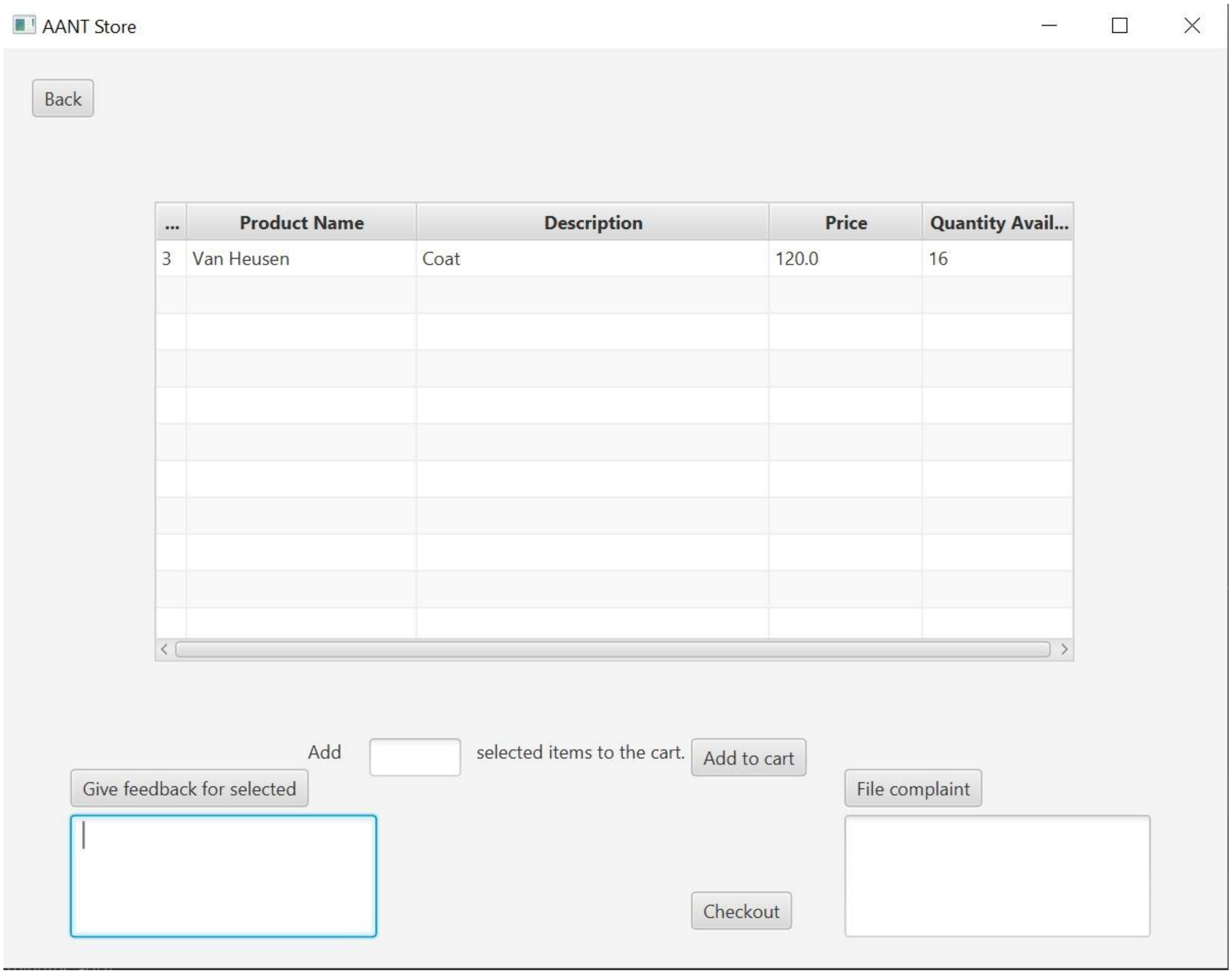
List all Coupons which were used at least twice.

```
select couponID
from coupons
where 2 <= (select count(couponID)
                    from applies
                    where coupons.couponID = applies.couponID);
```

List all unused Coupons:

```
select couponID, minOrder, validTillDate, discount
from coupons natural left outer join applies
where orderID is null and validTillDate >= curdate();
```

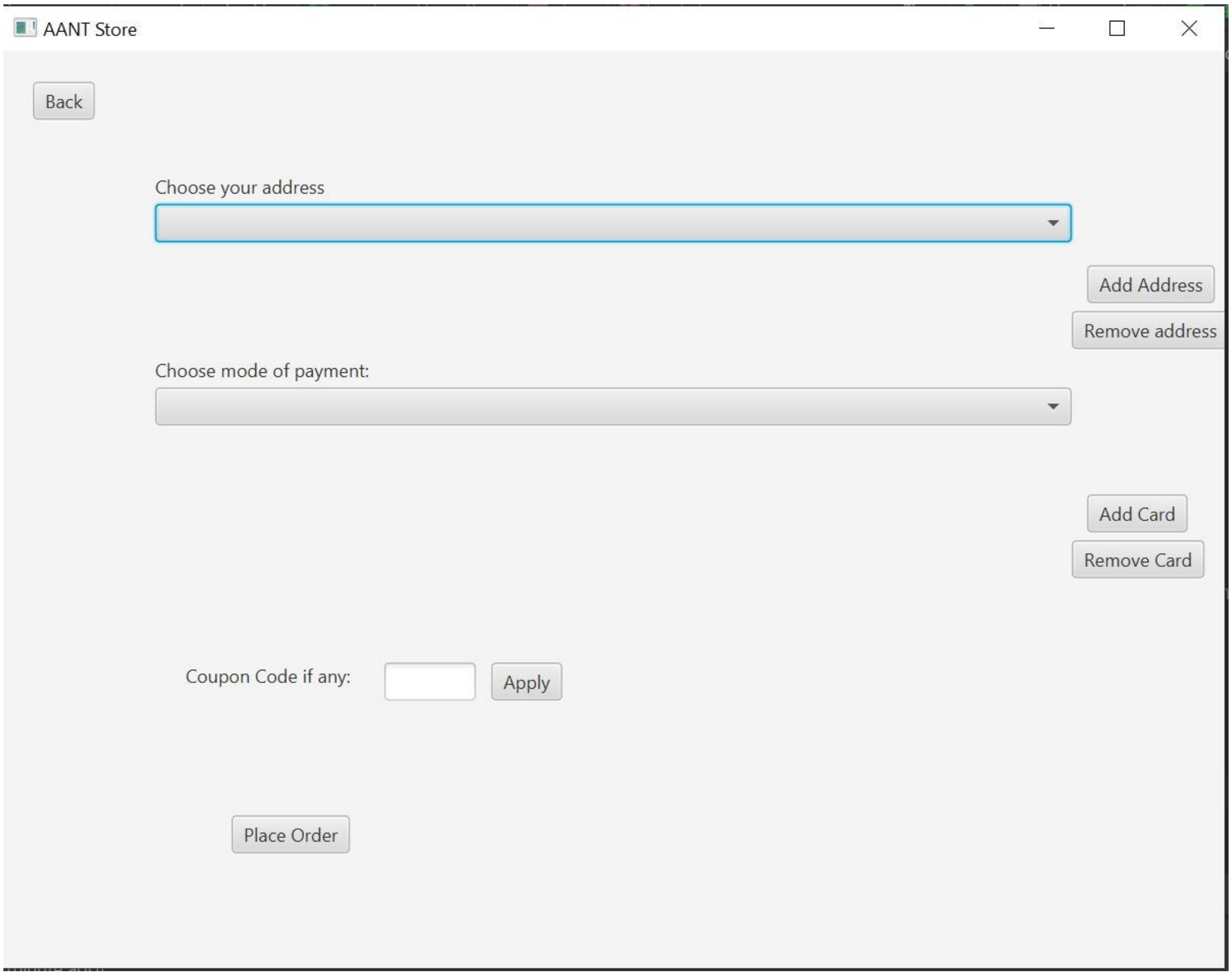
Screen Shots:

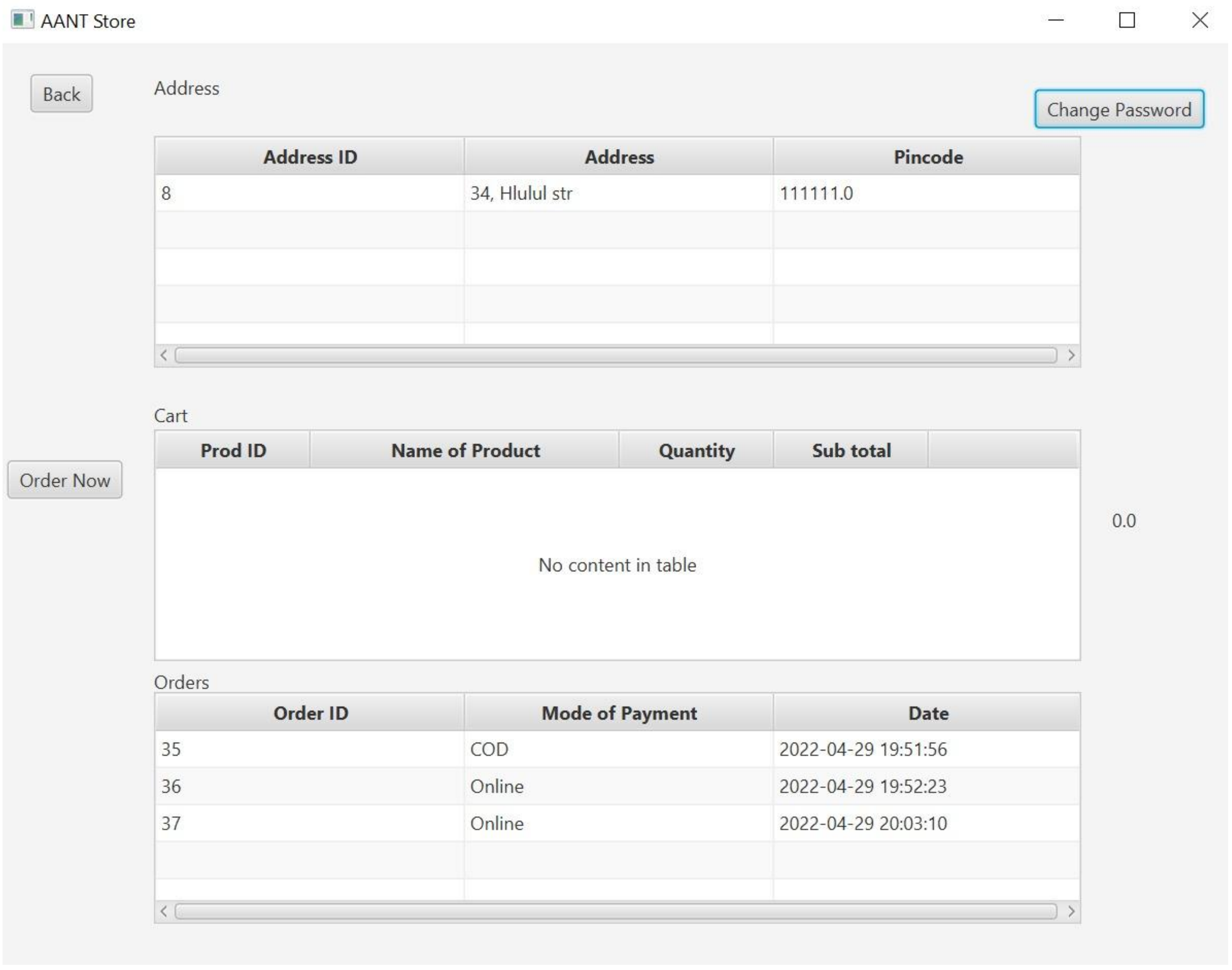

AANT Store

Back

...	Product Name	Description	Price	Quantity Avail...
3	Van Heusen	Coat	120.0	16

Add selected items to the cart.


AANT Store
Back
Choose your address
Add Address
Remove address
Choose mode of payment:
Add Card
Remove Card
Coupon Code if any:
Apply
Place Order


AANT Store

Back

Order Now

Change Password

Address

Address ID	Address	Pincode
8	34, Hlulul str	111111.0

Cart

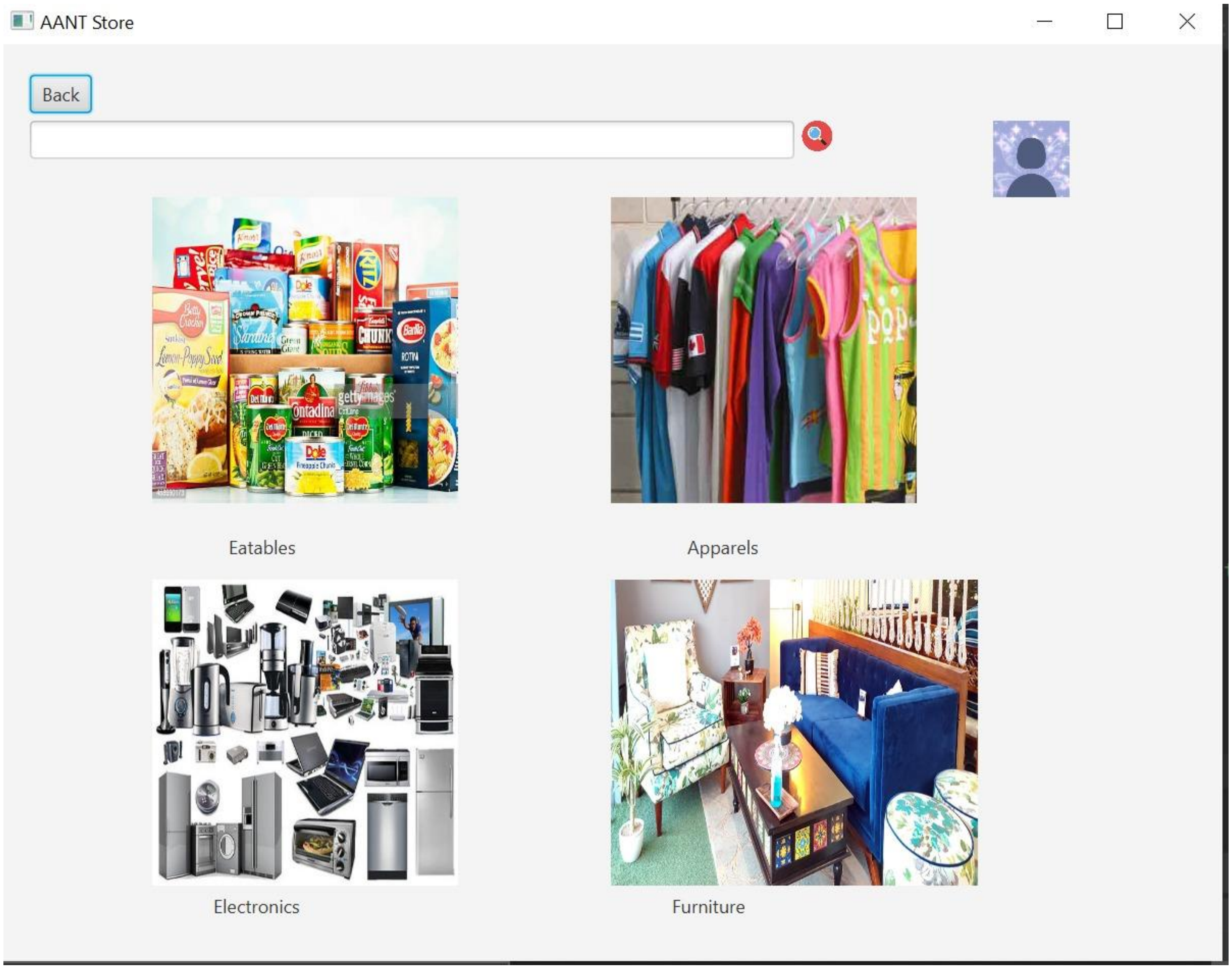
Prod ID	Name of Product	Quantity	Sub total	

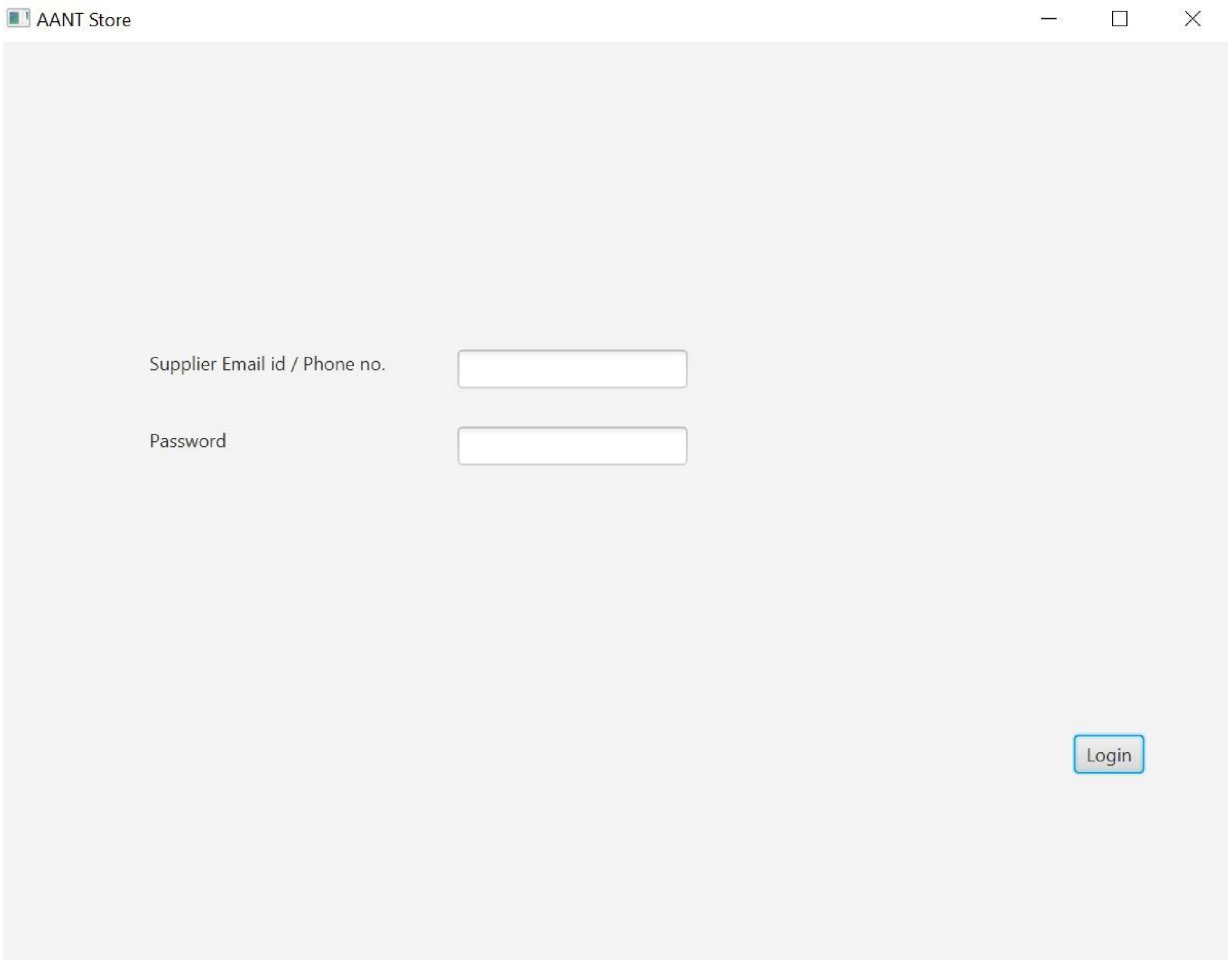
No content in table

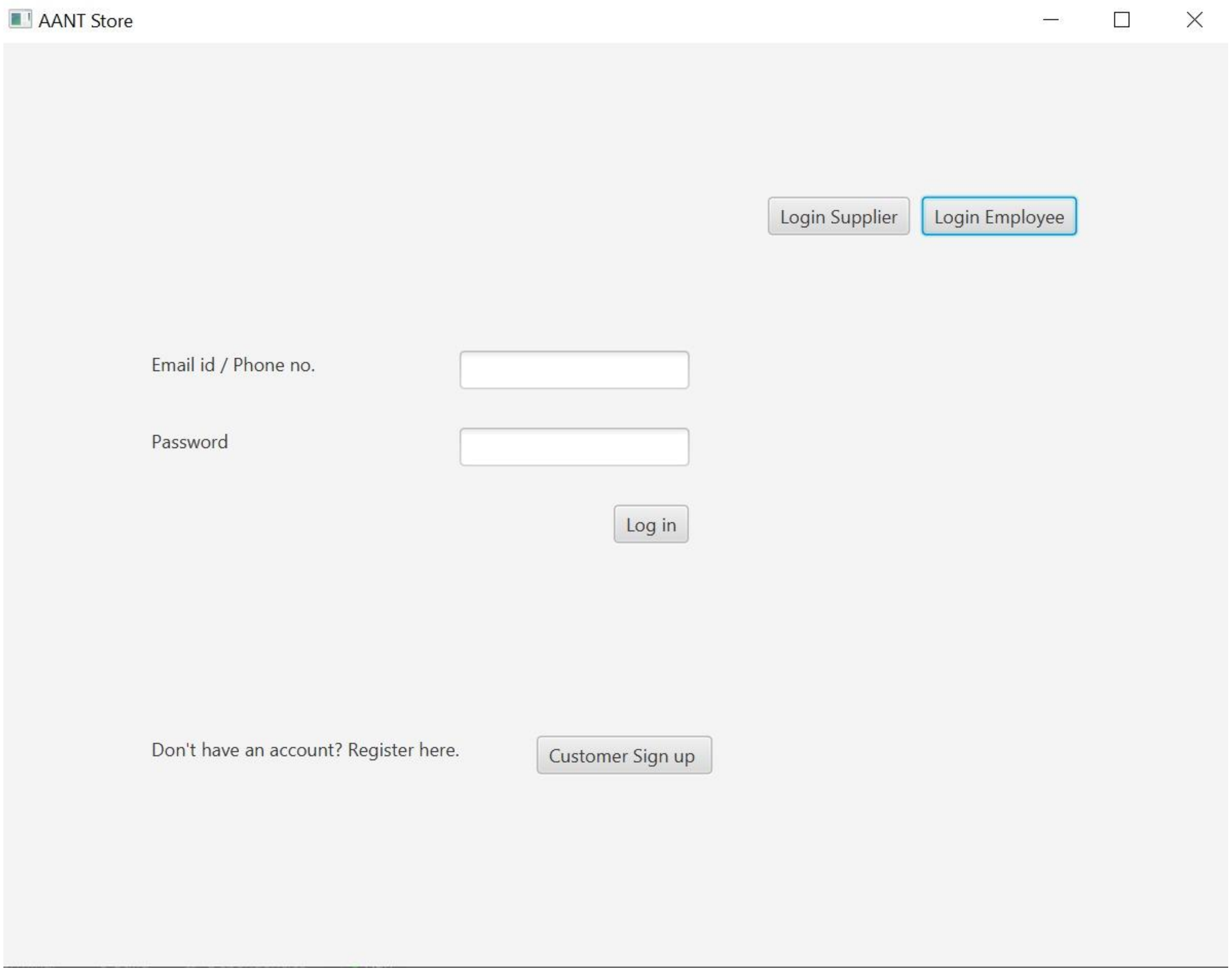
0.0

Orders

Order ID	Mode of Payment	Date
35	COD	2022-04-29 19:51:56
36	Online	2022-04-29 19:52:23
37	Online	2022-04-29 20:03:10


AANT Store
Back
Eatables
Apparels
Electronics
Furniture


AANT Store
Supplier Email id / Phone no.
Password
Login


AANT Store
Login Supplier
Login Employee
Email id / Phone no.
Password
Log in
Don't have an account? Register here.
Customer Sign up