DBMS Project - Group 25
Shantanu Prakash 2021285
Shivesh Gulati   2021286

Online Retail Store

Project Scope :

The basic theme of this project is to create an online grocery shopping website/application that will deliver groceries to the users at their doorstep in minimal time.

Our Website will include products of several categories each stored as a separate table (relation), and will also provide users the ability to sort products based on their price-range and discounts within each category.

The scope of our website will not only include products and categories, but will also have 3 main types of users:-
1) The customer
2) The admin
3) The transporter
The data for all these 3 types of users along with their access permissions would also be stored.

Functional Requirements

1. Customer's Perspective

1) Users can either sign up by filling their details (name, address, phone number) or login by entering username and password.
i) If the user is an existing user, then they are directed to homepage
ii) If they are a new user, then they are directed to the profile page

2) After reaching the homepage, the customer can choose the products along with the quantities they wish to buy and then check their cart.

3) If the user satisfied, with the items they have added on the cart, they can proceed to checkout.

4) They can also cancel their order if it has not been delivered yet if they wish to.

2. Admin's Perspective

1) If the user logins as an admin, then they are redirected to the homepage to the product/category add/delete page, and review new transporter requests.

2) Admin can also set deals and discounts on products.

3. Transporter's Perspective

1) If the user wishes to apply as a transporter, then he/she can proceed by filling in their details, passing identity-checks and then will be granted authorization by the admin of the application, and can thereby proceed to work as an authorized transporter.

2) The Authorized user can login by entering their username and password, then on authentication will be redirected to reject where pending requests of users will be shown to them and they can accept any one of them.

Technical Requirements

Tech Stack: MYSQL, HTML, CSS, JS, DJANGO
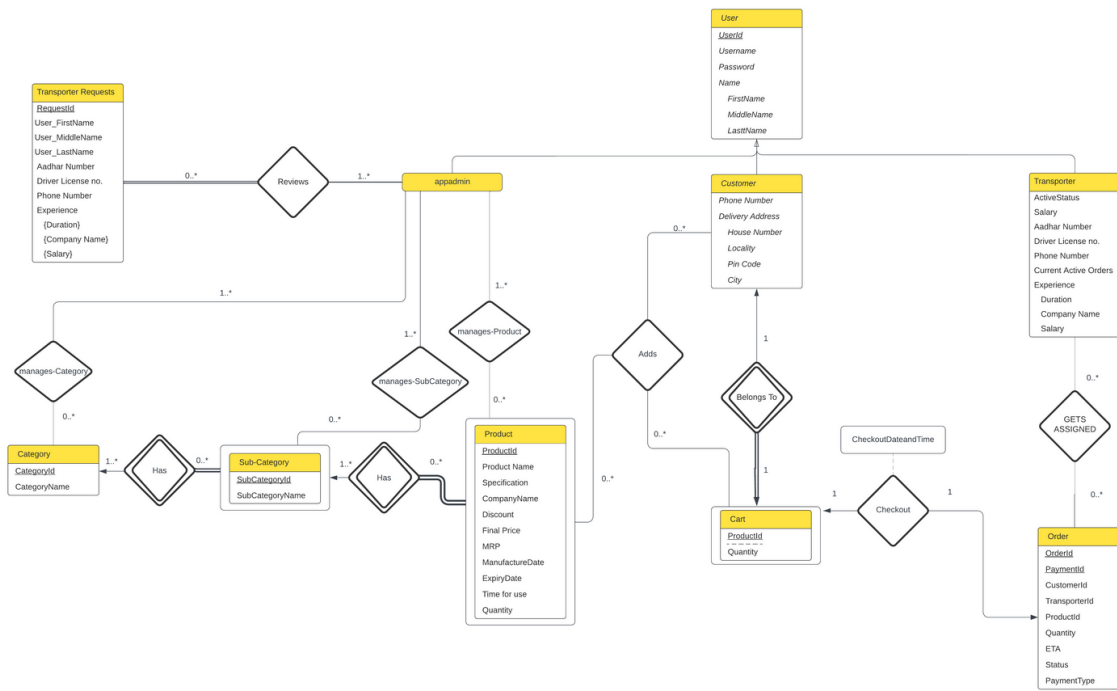
1) Database: MYSQL
2) Front-End: HTML, CSS, JS
3) Back-End + Routing: DJANGO

Entity Relationship Diagram

Here is the link to our Lucid chart we made of ER Diagram and picture corresponding to the same has been pasted to the document: ClickHereToGoTheLucidChart

ER Model

Entities, Attributes and Schemas

1. Customer Table

Fields	Constraints	Data Type (Domain Constraint)
CustomerId	Entity Integrity Constraint+Key Constraint = primary key	BIGINT
Username	Entity Integrity Constraint = unique value + not null	VARCHAR(50)
UserPassword	Entity Integrity Constraint = not null	VARCHAR(100)
User_FirstName	Entity Integrity Constraint= not null	VARCHAR(100)
User_MiddleName	-	VARCHAR(100)
User_LastName	-	VARCHAR(100)
PhoneNumber	Entity integrity constraint = unique val+not null	BIGINT (10dig)
Customer_HouseNumber	Entity Integrity constraint = not null	VARCHAR(100)
Customer_PinCode	Entity integrity constraint = not null	VARCHAR(6)

Fields	Constraints	Data Type (Domain Constraint)
Customer_City	Entity integrity constraint = not null	VARCHAR(100)
Customer_Locality	Entity integrity constraint = not null	VARCHAR(100)

Explanations for Constraints (Common for all Users Admin + Transporter + Customer)

    1) Customer Id and Username: Uniquely Identifies and is never NULL each Customer

    2) UserPassword and Firstname: Every User must have a password and firstname

    3) PhoneNumber: Each Customer must have a unique and NOT NULL phone number.

2. Transporter Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity Integrity Constraint+Key Constraint = primary key	BIGINT
Username	Entity Integrity Constraint = unique value + not null	VARCHAR(50)
UserPassword	Entity Integrity Constraint = not null	VARCHAR(100)
User_FirstName	Entity Integrity Constraint= not null	VARCHAR(100)
User_MiddleName	-	VARCHAR(100)
User_LastName	-	VARCHAR(100)
PhoneNumber	Entity integrity constraint = unique val+not null	BIGINT (10dig)
ActiveStatus	Entity Integrity constraint = not null	TINYINT (0/1/2)
AadharNumber	Entity integrity constraint = unique	BIGINT

Fields	Constraints	Data Type (Domain Constraint)
DrivingLicenseNumber	Entity integrity constraint = not null + unique	BIGINT
CurrentActiveOrders	Entity integrity constraint = not null	INT
PresentSalary	Entity integrity constraint = not null	FLOAT

Explanations for some of the Constraints

1) AadharNumber: Unique for all Drivers, but each driver may not have an AadharNumber, hence it may be null. It must always be a 10-digit number (Domain Constraint).

2) DrivingLicenseNumber Each driver must have a driving license number, which is unique for each driver and is a must and hence unique and not null.

3) Active Status : Each driver has an active status, 0 => unavailable, 1 => Available but no orders, 2 => Completing some order.

3. Transporter_Years Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity integrity constraint = not null Referential Integrity Constraint = foreign key	BIGINT
Years	Entity integrity constraint = not null	INT

4. Transporter_CompanyName Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity integrity constraint = not null Referential Integrity Constraint = foreign key	BIGINT
CompanyName	-	VARCHAR(100)

SOCIAL MEDIA REPORT

5. Transporter_Salary Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity integrity constraint = not null Referential Integrity Constraint = foreign key	BIGINT
Salary	-	FLOAT

Explanations for some of the Constraints

Each transporter may have previous work experience, which is a complex attribute (Composite + Multivalued), as it has fields like, company name, salary and years.

Hence, separate table has been created for each of them, each having the TransporterId as foreign key.

6. TransporterRequests Table

Fields	Constraints	Data Type (Domain Constraint)
RequestId	Entity Integrity Constraint+Key Constraint = primary key	BIGINT
User_FirstName	Entity Integrity Constraint= not null	VARCHAR(100)
User_MiddleName	-	VARCHAR(100)
User_LastName	-	VARCHAR(100)
PhoneNumber	Entity integrity constraint = unique val+not null	BIGINT (10dig)
Aadhar Number	Entity Integrity constraint = unique	BIGINT
Driving License Number	Entity integrity constraint = not null + unique	BIGINT

7. TransporterRequests_Years Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity integrity constraint = not null Referential Integrity Constraint = foreign key	BIGINT
Years	-	INT

8. TransporterRequests_CompanyName Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity integrity constraint = not null Referential Integrity Constraint = foreign key	BIGINT
CompanyName	-	VARCHAR(100)

9. TransporterRequests_Salary Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity integrity constraint = not null Referential Integrity Constraint = foreign key	BIGINT
Salary	-	FLOAT

10. Category Table

Fields	Constraints	Data Type (Domain Constraint)
CategoryId	Entity Integrity Constraint+Key Constraint = primary key	BIGINT
CategoryName	Entity Integrity Constraint= not null+ unique	VARCHAR(100)

11. SubCategory Table

Fields	Constraints	Data Type (Domain Constraint)
CategoryId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
SubCategoryId	Entity Integrity Constraint= not null+ unique	BIGINT
	Category Id + SubCategoryId Together Entity Integrity Constraint = primary key	
SubCategoryName	Entity Integrity Constraint= not null+ unique	VARCHAR(100)

12. Product Table

Fields	Constraints	Data Type (Domain Constraint)
CategoryId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
SubCategoryId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
ProductId	Entity Integrity Constraint = NOT NULL + UNIQUE	BIGINT

CategoryId+SubCategoryId+ProductId together	Entity Integrity Constraint = primary key	
ProductName	Entity Integrity Constraint = NOT NULL +.Unique	VARCHAR(100)
CompanyName	Entity Integrity Constraint = not null	VARCHAR(50)
Specification	Entity Integrity Constraint = not null	VARCHAR(100)
Discount	Entity Integrity Constraint= not null	FLOAT
MRP	Entity Integrity Constraint = not null	FLOAT
FinalPrice	-	FLOAT
ManufacturingDate	Entity Integrity Constraint = not null	DATE
ExpiryDate	Entity Integrity constraint = not null	DATE
TimeForUse	-	DAY

 Explanations for some of the Constraints

1) Category Id, SubCategory Id and Product Id : A subcategory cannot exist without a category, similarly a product cannot exist without a subcategory, and hence subcategory is a weak entity dependent on category and product a weak entity dependent on subcategory.
 Therefore Primary Key for SubCategory=> CategoryId + SubCategoryId
 Primary Key for Product => CategoryId + SubCategoryId + ProductId

2) CompanyName + ProductName+ ManufacturingDate : Miscellaneous attributes for each product, which describe the product and hence cannot be NULL.

13. ManagesCategory Table

Fields	Constraints	Data Type (Domain Constraint)
CategoryId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
AdminId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
CategoryId+AdminId	Entity Integrity Constraint + Key Constraint = Primary Key	

14. ManagesSubCategory Table

Fields	Constraints	Data Type (Domain Constraint)
CategoryId	Entity Integrity Constraint = NOT NULL	BIGINT
SubCategoryId	Entity Integrity Constraint = NOT NULL	BIGINT
AdminId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
CategoryId+SubCategoryId+AdminId	Entity Integrity Constraint + Key Constraint = Primary Key	

15. Adds Table

Fields	Constraints	Data Type (Domain Constraint)
CategoryId	Entity Integrity Constraint = NOT NULL	BIGINT
SubCategoryId	Entity Integrity Constraint = NOT NULL	BIGINT
ProductId	Entity Integrity Constraint = NOT NULL	BIGINT
CustomerId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
CategoryId+SubCategoryId+ProductId+CustomerId	Entity Integrity Constraint + Key Constraint = Primary Key	
CategoryId+SubCategoryId+ProductId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	

16. Reviews Table

Fields	Constraints	Data Type (Domain Constraint)
RequestId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
AdminId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT

17. ManagesProduct Table

Fields	Constraints	Data Type (Domain Constraint)
CategoryId	Entity Integrity Constraint = NOT NULL	BIGINT
SubCategoryId	Entity Integrity Constraint = NOT NULL	BIGINT
ProductId	Entity Integrity Constraint = NOT NULL	BIGINT
AdminId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
CategoryId+SubCategoryId+ProductId+AdminId	Entity Integrity Constraint + Key Constraint = Primary Key	
CategoryId+SubCategoryId+ProductId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	

18. GetsAssigned Table

Fields	Constraints	Data Type (Domain Constraint)
TransporterId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
OrderId	Entity Integrity Constraint = NOT NULL Referential Integrity Constraint = Foreign Key	BIGINT
OrderId+TransporterId	Entity Integrity Constraint + Key Constraint = Primary Key	

Explanations for some of the Constraints

All these tables denote the various tables for the various relationships between the different entities.

1) <u>Manages Category/Subcategory/Product Relation:</u> This relation, between admin and category/subcategory/product, where admin adds/deletes the categories/creates new categories. Hence it has relevant ids, (Product id/category id) as its foreign keys.

2) <u>Gets Assigned Relation:</u> This relation exists between order and transporter, where each transporter gets assigned one or more orders to be completed. Hence , here the orderId + TransporterId acts as primary key (unique + not null) and each of them individually as foreign keys.

19. Cart Table

Fields	Constraints	Data Type (Domain Constraint)
CustomerId	Entity Integrity Constrain = NOT NULL Referential Integrity Constraint = Foreign key	BIGINT
CategoryId	Entity Integrity Constraint= not null	BIGINT
SubCategoryId	Entity Integrity Constraint = not null	BIGINT
ProductId	Entity Integrity Constraint= not null	BIGINT
QuantityId	Entity Integrity Constraint= not null	BIGINT
CustomerId+CategoryId+SubCategoryId+ProductId	Entity Integrity Constraint+Key Constraint = primary key	
CategoryId+SubCategoryId+ProductId	Referential Integrity Constraint = foreign key	

Explanations for some of the Constraints

<u>Primary Key ;</u> Every customer has a cart, and hence, the cart table has a CustomerId, a productId, categoryId and a productId, jointly acting as a primary key, because every cart will be linked to a customer and every cart may have one or more products in it.

19. Delivery Table

Fields	Constraints	Data Type (Domain Constraint)
OrderId	Entity Integrity Constraint = Not NULL + Unique	BIGINT
PaymentId	Entity Integrity Constraint= not null + unique	BIGINT
CustomerId	Entity Integrity Constrain = NOT NULL Referential Integrity Constraint = Foreign key	BIGINT
TransporterId	Entity Integrity Constrain = NOT NULL Referential Integrity Constraint = Foreign key	BIGINT
CategoryId	Entity Integrity Constraint= not null	BIGINT
SubCategoryId	Entity Integrity Constraint = not null	BIGINT
ProductId	Entity Integrity Constraint= not null	BIGINT
QuantityId	Entity Integrity Constraint= not null	BIGINT
CheckOutDateAndTime	Entity Integrity Constraint= not null	DATETIME
ETA	-	TIME
OrderId+PaymentId	Entity Integrity Constraint+Key Constraint = primary key	
CategoryId+SubCategoryId+ProductId	Referential Integrity Constraint = foreign key	

Weak Entity

In our Relational schema of Online Retail store, we have implemented multiple weak entites:

1) SubCategory is a weak entity of Category because a subcategory exists under some category. And Product is a weak entity of Subcategory because just like above a product belongs to some category and subcategory. It is irrelevant for a subcategory to exist without a category above it and similarly same goes for product and subcategory.

2) Cart is a weak entity of customer because cart is related to a user and entry in a cart exists only when a customer adds some product to the cart.

Ternary Relation

In our Relational schema of Online Retail store, we have implemented multiple ternary relations:

1) Between Customer, Product and Cart: Customer can add products in their cart.

List of SQL Queries (Embedded + Non-Embedded):

Type 1: Queries (Admin Interaction with tables)

1) Viewing the number of products in various categories.

This Query is to view the number of products in various categories by left joining the category table on product and counting the number of products with help of count() aggregate function.

SELECT Category.CategoryId as 'Id' ,CategoryName as 'Category Name',
COUNT(product.ProductId)
AS 'Number of Products'
FROM Category
LEFT JOIN Product on Category.CategoryId=Product.CategoryId
GROUP BY Category.CategoryId;

2) Number of different products in various subcategories

This Query is to view the number of products in various subcategories by inner joining the subcategory table on product and counting the number of products with help of count().

SELECT SubCategory.CategoryId AS 'Category Id', SubCategory.SubCategoryId AS 'Sub-Category Id',SubCategory.SubCategoryName AS 'Sub-Category Name', COUNT(Product.ProductId) AS 'Number of Products'
FROM SubCategory
INNER JOIN Product on (SubCategory.SubCategoryId=Product.SubCategoryId and Product.CategoryId=SubCategory.CategoryId)
GROUP BY SubCategory.SubCategoryId;

3) Admin wants to know information regarding the product which are expiring in less than 3 days

This Query projects all details of product which are about to expire in less than 3 days where TIMESTAMPDIFF() has been used to calculate time left for product to expire from the present day.

SELECT P.CategoryId as 'Category Id', P.SubCategoryId as 'Sub-Category Id', P.ProductId as 'Product-Id' ,P.ProductName as 'Name'
FROM product as P
WHERE (TIMESTAMPDIFF(DAY, CURDATE(), P.ExpiryDate)<3);

4) Deleting the expired products by admin

This Query deletes all products from the table which have expired. Here we've used ' SET sql_safe_updates=0' to delete the entries from product table without the use of primary key.

SET sql_safe_updates=0;
DELETE from product WHERE (TIMESTAMPDIFF(DAY,CURDATE(),product.ExpiryDate)<=0);
SET sql_safe_updates=1;

Type 2: Queries (Admin and Transporter Requests)

1) Updating TransporterRequests based on actions of Admin

Based on the accept/reject input by the admin on the frontend, changes are made in the database using the update query in transporter requests table and the entry is then deleted from TransporterRequests table using delete query which is further handled by trigger.

UPDATE TransporterRequests SET RequestStatus = %s WHERE RequestId = %s
DELETE FROM TransporterRequests Where RequestId = %s

2) Admin Rejects the request of a transporter

This Query sets the 'RequestStatus=2' for TransporterRequests who have been Rejected by the admin with the help of 'UPDATE' command.

INSERT INTO Reviews (AdminId,RequestId) VALUES(3,2);
UPDATE TransporterRequests
SET
 RequestStatus=2
Where
 RequestId=2;

DBMS PROJECT REPORT

3) Displaying Transporter Requests to Admin on frontend:

Displays all the transporter requests on the admin page which is ordered by the RequestId (Primary key of TransporterRequests Table)

1) SELECT RequestId, User_FirstName, User_MiddleName, User_LastName,PhoneNumber FROM TransporterRequests ORDER BY RequestId

Type 3: Queries (Customer and Products/Categories/SubCategories)

1) Customer shown all the categories on the home page.

This Query shows customer all the categories from the Category table

*SELECT * from Category;*

2) Customer shown the subcategories for the given category.

This Query shows customer subcategories corresponding to a given Category.

SELECT SubCategory.CategoryId,SubCategory.SubCategoryId,SubCategory.SubCategoryName
FROM SubCategory
WHERE SubCategory.CategoryId=%s
Order by SubCategoryId;

Type 4: Queries (Filters Queries)

1) Displays product in a particular SubCategory in order by finalPrice

This Query shows the list of products of a particular subcategory in order by their final price=(MRP-(Discount/100)*MRP).

SELECT product.ProductName, product.finalPrice
FROM product ORDER BY product.FinalPrice;

2) **Display products in a particular SubCategory in order by discount.**

This Query shows a list of products in a particular subcategory in order by 'discount' set on each product.

SELECT product.productName,product.Discount
from product order by product.Discount;

3) Display products that belong to a particular manufacturer.

This Query displays a list of products that belongs to a particular manufacturer (here 'Haldiram')

SELECT product.productName, product.CompanyName
from product
WHERE product.CompanyName='Haldiram';

Type 5: Queries (Cart and Orders related)

1) Viewing the cart of a particular Customer

This Query shows cart details of a particular customer using the inner join from cart table on product table and accessing the row entries with the help of customerId.

SELECT Cart.ProductId as 'Product Id', Product.ProductName as 'Product Name', Cart.Quantity as 'Quantity Added'
FROM Cart
INNER JOIN Product ON Product.ProductId=Cart.ProductId
WHERE Cart.CustomerId=1;

2) Viewing order history of a particular customer

This Query shows the order history of a particular customer using the inner join from the delivery table on the product table using the DeliveryStatus attribute from the delivery table marked as 2 which indicates order delivered.

SELECT delivery.OrderId as 'Order Id', delivery.PaymentId as 'Payment Id', delivery.PaymentType as 'Payment Type', delivery.ProductId as 'Product Id', Product.ProductName as 'Product Name', delivery.Quantity as 'Quantity Added'
FROM delivery
INNER JOIN Product ON Product.ProductId=delivery.ProductId
WHERE delivery.CustomerId=1 and delivery.DeliveryStatus=2;

3) Viewing currently active orders of a customer

This Query shows the present orders of a particular customer using the inner join from the delivery table on the product table using the DeliveryStatus attribute from the delivery table marked as 0 or 1 which indicates order not delivered.

SELECT delivery.OrderId as 'Order Id', delivery.PaymentId as 'Payment Id', delivery.PaymentType as 'Payment Type', delivery.ProductId as 'Product Id', Product.ProductName as 'Product Name', delivery.Quantity as 'Quantity Added'
FROM delivery
INNER JOIN Product ON Product.ProductId=delivery.ProductId
WHERE ((delivery.CustomerId=3) and (delivery.DeliveryStatus=0 or delivery.DeliveryStatus=1));

4) Cancel Order of a particular Customer
DELETE FROM Delivery WHERE CustomerId = %s and CheckOutDateAndTime = %s
UPDATE Product Set Quantity = Quantity + %s WHERE ProductId = %s
UPDATE Transporter Set ActiveStatus = 1 , CurrentActiveOrders = CurrentActiveOrders - 1 WHERE TransporterId = %s

3) Display products that belong to a particular manufacturer.

This Query displays a list of products that belongs to a particular manufacturer (here 'Haldiram')

SELECT product.productName, product.CompanyName
from product
WHERE product.CompanyName='Haldiram';

Type 6: Queries (Relationship related)

1) Displays a list of Active transporters with orders<=10

Display the list of those transporters which are eligible for taking orders (i.e. those which are online and have <3 active orders

SELECT transporter.TransporterId as 'Id', transporter.Username as 'TransporterName'
from transporter
where transporter.CurrentActiveOrders<=10 and ActiveStatus=1;

2) Order of a customer gets assigned a transporter

Assigning the order of a particular customer to an eligible transporter. For this query, first the delivery table is updated and the transporter id is set to the available transporter that is eligible for taking that order. In the second part of the query, the CurrentActiveOrders of that transporter are incremented by 1.

UPDATE Delivery
SET
 Delivery.TransporterId=(SELECT Transporter.TransporterId from Transporter WHERE
Transporter.ActiveStatus=1 and Transporter.CurrentActiveOrders<3 LIMIT 1)
WHERE
 Delivery.CustomerId=4;

UPDATE Transporter
RIGHT JOIN Delivery
ON Transporter.TransporterId=Delivery.TransporterId
SET
  Transporter.CurrentActiveOrders=Transporter.CurrentActiveOrders+1
WHERE
 Delivery.CustomerId=4;

Type 7: Queries (Showcasing Constraints)

1) Constraints for Admin Table

SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='AppAdmin';

2) Constraints for transporter table

```
SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='transporter';
```

Here Phone Number has to be between 1000000000 and 9999999999. Therefore, domain constraint violation occurs here.

```
INSERT INTO Transporter(Username,UserPassword,User_FirstName,PhoneNumber,DrivingLicenseNumber,
PresentSalary,CurrentActiveOrders,ActiveStatus) VALUES ('Constraint Violation Test
2','abcde123','Hello',1234,123,3000,0,1);
```

Here ActiveStatus has to be either 0 or 1. Therefore here domain constraint violation occurs here as ActiveStatus has been entered 3.

```
INSERT INTO
 Transporter(Username,UserPassword,User_FirstName,PhoneNumber,DrivingLicenseNumber,PresentSalary,CurrentActiv
eOrders,ActiveStatus) VALUES ('Constraint Violation Test','abcde123','Hello',1234567890,123,30,0,3);
```

3) Constraints for TransporterRequests Table

```
SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='TransporterRequests';
```

4) Constraints for Customer Table

```
SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='Customer';
```

5) Constraints for Category Table

```
SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='Category';
```

6) Constraints for SubCategory Table

```
SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='SubCategory';
```

7) Constraints for Product Table

```
SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='Product';
```

Here Discount has been entered as -10. But Discount attribute has a domain constraint>=0 hence domain constraint violated here and entry fails to get updated in Product table.

INSERT INTO Product
(CategoryId,SubCategoryId,ProductName,MRP,Discount,ManufacturingDate,ExpiryDate,CompanyName,Qu
antity) VALUES (5,17,'Sting33',100,-10,'2023-02-10','2023-02-12','Stingz',10);

Here CategoryId has been entered as '6'. CategoryId is a Foreign key referenced from 'Category' Table. But CategoryId '6' does not exist in 'Category' hence cannot be added in our product table therefore foreign key constraint violated here.

INSERT INTO Product
(CategoryId,SubCategoryId,ProductName,MRP,Discount,ManufacturingDate,ExpiryDate,CompanyName,Qu
antity) VALUES (6,17,'Sting3223',100,0,'2023-02-10','2023-02-12','Stingz',10);

8) Constraints for Delivery Table

SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='Delivery';

9) Constraints for Cart Table

SELECT TABLE_NAME,CONSTRAINT_TYPE,CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='Cart';

Here this entry lacks entry corresponding to the quantity attribute or quantity=NULL.
But our Quantity attribute has a 'NOT NULL' constraint. Hence Constraint violation occurs
`and entry fails to get inserted in our table.

INSERT INTO Cart (CustomerId,CategoryId,SubCategoryId,ProductId) VALUES (10,5,16,13);

Triggers Implemented:

1) Trigger due to changes made in transporterrequests table by admin (Before Delete Trigger)

This Trigger runs before deleting an entry from TransporterRequests. Whenever RequestStatus of any TransporterRequests entry is set to 1 on admin page at frontend, it indicates that the request of the transporter has been accepted by the admin and the record of this transporter request is inserted into Transporter Table.

delimiter //
CREATE TRIGGER del_and_insert_req
BEFORE DELETE
ON TransporterRequests
FOR EACH ROW
BEGIN
if old.RequestStatus =1 THEN
INSERT INTO
Transporter(UserName,User_FirstName,User_MiddleName,User_LastName,PhoneNumber,AadharNumber,DrivingLicenseNu
mber)
SELECT CONCAT(User_FirstName,RequestId), User_FirstName,
User_MiddleName,User_LastName,PhoneNumber,AadharNumber,DrivingLicenseNumber
FROM TransporterRequests
WHERE RequestId = OLD.RequestId;
end if;
end//
delimiter ;

2) Trigger due to deleting an entry from cart and placing order for it (AFTER DELETE TRIGGER)

This Trigger runs after deleting an entry from cart whose order is to be placed. To Place the order for the deleted entry of cart, we select transporterId and CurrentActiveOrders of a transporter to be assigned to the order into variables. Trigger ensures that customer cannot place an order for a product in a quantity which is greater than the quantity of product present in the product table representing the stock of products. Trigger also ensures changes are made in transporter table corresponding to the transporter who has been allotted this order.

```
delimiter //
CREATE TRIGGER del_and_insert_cart
BEFORE DELETE
ON Cart
FOR EACH ROW
BEGIN
SELECT TransporterId INTO @Transporter_Id FROM Transporter WHERE Transporter.ActiveStatus = 1 and
Transporter.currentActiveOrders<=10 ORDER BY TransporterId LIMIT 1;
SELECT CurrentActiveOrders INTO @Active_Orders FROM Transporter WHERE Transporter.TransporterId =
@Transporter_Id;
SELECT QUANTITY INTO @Prod_Quantity FROM PRODUCT WHERE Product.ProductId = OLD.ProductId and
Product.CategoryId = OLD.CategoryID and Product.SubCategoryId = OLD.SubCategoryId;
SELECT COUNT(ProductId) INTO @TOT_PROD FROM Cart WHERE CustomerId = OLD.CustomerId Group By CustomerId;

if @Prod_Quantity>OLD.Quantity THEN
  INSERT INTO Delivery
(PaymentId,CustomerId,TransporterId,CategoryId,SubCategoryId,ProductId,Quantity,PaymentType,CheckOutDateAn
dTime,DeliveryStatus) VALUES
(2000+Old.CustomerId,Old.CustomerId,@Transporter_Id,OLD.CategoryId,OLD.SubCategoryId,OLD.ProductId,OLD.Qua
ntity,"PayTM",NOW(),1);

  UPDATE Transporter
  SET CurrentActiveOrders = CurrentActiveOrders +1
  WHERE CurrentActiveOrders<=10 and Transporter.TransporterId = @Transporter_Id;

  UPDATE Product
  SET Quantity = Quantity - OLD.Quantity
  WHERE Product.ProductId = OLD.ProductId and Product.CategoryId = OLD.CategoryId and Product.SubcategoryId =
OLD.subCategoryId ;

 if @Active_Orders >=9 THEN
   UPDATE Transporter
   SET ActiveStatus = 0
   WHERE TransporterId = @Transporter_Id;
  end if;
end if;
end //
delimiter ;
```

OLAP Queries:

- *SELECT CategoryId, SubCategoryId, ProductId, count(CustomerId) as 'NumberOfCustomers'*
  FROM Cart
  GROUP BY CategoryId, SubCategoryId, ProductId WITH ROLLUP;

- *SELECT CheckOutDateAndTime, ProductId, count(OrderId), GROUPING(CheckOutDateAndTime), GROUPING(ProductId)*
  FROM Delivery
  GROUP BY CheckOutDateAndTime, ProductId WITH ROLLUP;

- *SELECT TIMESTAMPDIFF(DAY,CURDATE(),product.ExpiryDate) as 'TimeLeft', Discount, count(productId)*
  FROM Product
  GROUP BY TimeLeft, Discount WITH ROLLUP;

- *SELECT CustomerId, ProductId, CheckoutDateAndTime, count(Quantity)*
  FROM Delivery
  GROUP BY CustomerId, ProductId, CheckOutDateAndTime WITH ROLLUP HAVING (GROUPING(CustomerId)=0 and GROUPING(ProductId)=0) or (GROUPING(ProductId)=1 and GROUPING(CustomerId)=0);

- *SELECT AdminId,RequestStatus,Count(RequestId)*
  FROM Reviews JOIN TransporterRequests on TransporterRequests.RequestId=Reviews.RequestId
  GROUP BY Cube (AdminId,RequestStatus);

- *SELECT Transporter.TransporterId, Transporter.ActiveStatus, count(OrderId)*
  FROM GetsAssigned LEFT JOIN Transporter ON GetsAssigned.TransporterId=Transporter.TransporterId
  GROUP BY TransporterId, ActiveStatus WITH ROLLUP;

Transaction Queries:

Case 1: Two Admins trying to update the same transporter request

<u>Transaction T1:</u> Update Transporterrequests SET RequestStatus=1 where RequestId=23

<u>Transaction T2:</u> Update Transporterrequests SET RequestStatus=1 where RequestId=23
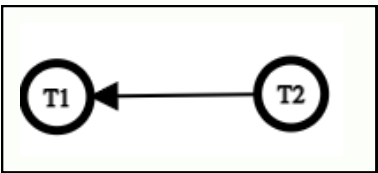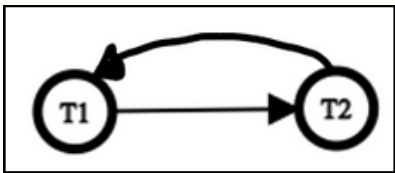
Directed Acyclic Graphs in case of (a) Nonconflicting serializable schedule and (b) Conflicting serializable schedule

<u>Non-conflicting Serializable schedule</u>

T1	T2
Read TransporterRequests Status of requestid= 23;	
	Read TransporterRequests Status of requestid= 23;
	SET status(requestid=23) = 2
SET status(requestid=23) = 2	
Write Status(requestid=23)	
	Write Status(requestid=23)
Delete from transporterrequests where requestId=23	
Insert into Transporter	
	Delete from transporterrequests where requestId=23 -> 0 row(s) changed
commit;	

Non-conflicting Serializable schedule (With Locking Mechanism)

T1	T2
Grant X(req_id=23)	
Read TransporterRequests Status of requestid= 23;	
	Request X(req_id=23) = BLOCKED
SET status(requestid=23) = 2	
Write Status(requestid=23)	
Delete from transporterrequests where requestId=23	
Insert into Transporter	
commit;	
Unlock X(req_id=23)	Grant X(req_id=23)
	Read TransporterRequests Status of requestid= 23;
	ABORTED because RequestId=23 no longer exists
	Unlock X(req_id=23)

Conflicting Serializable schedule

T1	T2
	Read TransporterRequests Status of requestid= 23;
	SET status(requestid=23) = 2
	Write Status(requestid=23)
	Delete from transporterrequests where requestId=23 -> 0 row(s) changed
	commit;
Read TransporterRequests Status of requestid= 23;	
ABORTED BECAUSE requestid=23 no longer exists	

Case 2: Three Customers attempting to buy the same product

Transaction T1,T2,T3: (Take quantity=10 of prodid=12 initially)
1) Delete from cart where productId=12 and customerId=%s;
2) Update Product SET Quantity=Quantity-10 where productId=12;
3) Insert into Delivery

Directed Acyclic Graphs in case of (a) Nonconflicting serializable schedule and (b) Conflicting serializable schedule

Non-conflicting Serializable schedule

T1	T2	T3
Read Product (prod_id=12)		
	Read Product (prod_id=12)	
		Read Product (prod_id=12)
		SET Quantity=Quantity-10
	SET Quantity=Quantity-10	
		delete from cart where proid=12 and cusid=3
		write prod.Quantity(id=12)
		Insert into Delivery
		commit;
SET Quantity=Quantity-10		

	delete from cart where proid=12 and cusid=2	
	write prod.Quantity(id=12)	
	Insert into Delivery	
delete from cart where proid=12 and cusid=1		
write prod.Quantity(id=12)		
Insert into Delivery		
	commit;	
commit;		

Non-conflicting Serializable schedule (With Locking Mechanism)

T1	T2	T3
GRANT X(prodid=12)		
Read Product (prod_id=12)		
	Request X(prodid=12) blocked	
		Request X(prodid=12) blocked
SET Quantity=Quantity-10		
delete from cart where proid=12 and cusid=1		
write prod.Quantity(id=12)		
Insert into Delivery		
commit;		
Unlock X(prodid=12)	Grant X(prodid=12)	
	Read Product (prod_id=12)	
	SET Quantity=Quantity-10	
	delete from cart where proid=12 and cusid=1	
	write prod.Quantity(id=12)	

	Insert into Delivery	
	commit;	
	Unlock X(prodid=12)	Grant X(prodid=12)
		Read Product (prod_id=12)
		SET Quantity=Quantity-10
		delete from cart where proid=12 and cusid=1
		write prod.Quantity(id=12)
		ROLLBACK;

Conflicting Serializable schedule

T1	T2	T3
Read Product (prod_id=12)		
SET Quantity=Quantity-10		
delete from cart where proid=12 and cusid=1		
write prod.Quantity(id=12)		
Insert into Delivery		
commit;		
	Read Product (prod_id=12)	
	SET Quantity=Quantity-10	
	delete from cart where proid=12 and cusid=1	
	write prod.Quantity(id=12)	
	Insert into Delivery	
	commit;	
		Read Product (prod_id=12)
		SET Quantity=Quantity-10

		delete from cart where proid=12 and cusid=1
		write prod.Quantity(id=12)
		ROLLBACK;

Case 3: An Admin trying to update quantity of some out of stock product and a customer attempting to buy the same product from cart concurrently. Consider initially quantity(productid=12) = 0

Transaction T1:Update Product set Quant=Quant+20 where proid=12
Transaction T2:
1) Delete from cart where prodid=12 and cusid=1;
2) Update Product set Quant=Quant-10 where proid=12
3) Insert into Delivery

Directed Acyclic Graphs in case of (a) Nonconflicting serializable schedule and (b) Conflicting serializable schedule

Non-conflicting Serializable schedule

T1	T2
Read Product (prod_id=12)	
	Read Cart.Quantity(prodid=12)
	Read Product (prod_id=12)
	SET Quantity=Quantity-10
SET Quantity=Quantity+20	
Write Product(prodid=12)	
commit;	
	delete from cart where proid=12 and cusid=1
	write prod.Quantity(id=12)
	ROLLBACK;

Non-Conflicting Serializable schedule (With Locking Mechanism)

T1	T2
Grant X(Prodid=12)	
Read Product (prod_id=12)	
	Grant X(cart_prodid=12_cusid=1)
	Read Cart.Quantity(prodid=12)
	Request X(prodid=12) BLOCKED
SET Quantity=Quantity+20	
Write Product(prodid=12)	
commit;	
Unlock X(prodid=12)	Grant X(prodid=12)
	Read Product (prod_id=12)
	SET Quantity=Quantity-10
	delete from cart where proid=12 and cusid=1
	write prod.Quantity(id=12)
	Insert into Delivery from cart
	commit;
	Unlock X(cart_prodid=12_cusid=1)
	Unlock X(prodid=12)

Conflicting Serializable schedule

T1	T2
Read Product (prod_id=12)	
SET Quantity=Quantity+20	
	Read Cart.Quantity(prodid=12)
Write Product(prodid=12)	
Commit;	

	Read Product (prod_id=12)
	SET Quantity=Quantity-10
	write prod.Quantity(id=12)
	delete from cart where proid=12 and cusid=1
	Insert into Delivery from cart
	commit;

Case 4: A Customer cancelling an order of product id=12 restoring quantity =10 in stock and another Customer attempting to buy the same product id=12 with quantity=10 from his cart. (Take product id12 quantity =0 before cancelling of order)

Transaction T1:
1) Delete from Delivery where cusid=1 and prodid=12;
2) Update Product set Quant=Quant+20 where proid=12;
Transaction T2:
1) Delete from cart where prodid=12 and cusid=2;
2) Update Product set Quant=Quant-10 where proid=12
3) Insert into Delivery
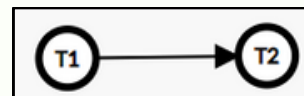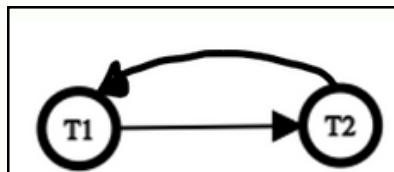
Directed Acyclic Graphs in case of (a) Nonconflicting serializable schedule and (b) Conflicting serializable schedule

Non-conflicting Serializable schedule

T1	T2
Read Delivery (prod_id=12)	
	Read Cart.Quantity(prodid=12)
	Read Product (prod_id=12)
Read Product(prodid=12)	
SET Quantity=Quantity+20	
Write Product(prodid=12)	
Delete from delivery where cusid=1 and proid=12	
commit;	
	SET Quantity=Quantity-10
	write prod.Quantity(id=12)
	ROLLBACK;
	delete from cart where proid=12 and cusid=1

Non-conflicting Serializable schedule (With Locking Mechanism)

T1	T2
GRANT X(Delivery(prodid=12))	
Read Delivery.Quantity(prod_id=12)	
	Grant X(Cart(prod_id=12))
	Read Cart.Quantity(prod_id=12)
	Grant X(Product(prod_id=12))
	Read Product.Quantity (prod_id=12)
	Set Product.Quantity(prod_id=12) = Set Product.Quantity(prod_id=12) + Cart.Quantity(prod_id=12)

	Write Product.Quanity(prod_id=12)
	rollback;
	U(Cart(prod_id=12)
Grant X(Product(prodid=12))	U(Product(prod_id=12);
Read Product(prod_id=12)	
Add	
Set Product.Quantity(prod_id=12) = Product.Quantity(prod_id=12) + Delivery.Quantity(prod_id=12);	
Write Product.Quantity(prod_id=12)	
Delete from delivery where cusid=1 and proid=12	
commit;	

Conflicting Serializable schedule (With Locking Mechanism)

T1	T2
Read Delivery.Quantity(prod_id=12)	
	Read Cart.Quantity(prod_id=12)
Read Product(prod_id=12)	
Set Product.Quantity(prod_id=12) = Product.Quantity(prod_id=12) + Delivery.Quantity(prod_id=12);	
Write Product.Quantity(prod_id=12)	
	Read Product.Quantity (prod_id=12)
Delete from delivery where cusid=1 and proid=12	
	SET Quantity=Quantity-Cart.Quantity(pid=12) cid=2
	Delete from cart where pid=12 cid=2
	Insert into Delivery pid=12
	Commit;
commit;	

Non-Conflicting Transaction Queries:

1) Admin enters a product to product table (Embedded SQL format)

```
try:
sqlFormula ="'' INSERT INTO Product
(CategoryId,SubCategoryId,ProductId,ProductName,specification,CompanyName,discount,mrp,ManufacturingDate,Expir
yDate,Quantity) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s) ''' values =
(categoryId,subCategoryId,productId,productName,specification,companyName,discount,mrp,manufacturing_date,expir
y_date,quantity)
 mycursor.execute(sqlFormula,values)

 if manufacturing_date>expiry_date:
 raise Exception("Manufacturing date cannot be before the expiry date")
mydb.commit()
return 1
except mysql.connector.Error as error:
mydb.rollback()
return -1
except Exception as e:
mydb.rollback()
return 2
```

2) A customer tries to place an order for an out of stock product from their cart

```
START TRANSACTION;
Read Cart.Quantity where pid=1 and cid=1
Read Product.Quantity where pid=1
if Cart.Quantity(pid=1 and cid=1) > Product.Quantity(pid=1) then
 ROLLBACK;
else
Delete from cart where pid=1 and cid=1
UPDATE Product set Product.Quantity=Product.Quantity+Cart.Quantity where pid=1
Insert into Delivery(cid,pid,quantity) values (1,1,cart.Quantity)
```

3) Two Customers (Cid=1 and Cid=2) tries to buy products(pid=1 and pid=2) of quantity 1 each

Transaction T1	Transaction T2
Delete from cart where pid=1 and cid=1;	
UPDATE Product SET Product.Quantity = Product.Quantity - 1 where pid=1;	
	Delete from cart where pid=2 and cid=2;
Insert into Delivery(cid,pid,quantity) values (1,1,1);	
	UPDATE Product SET Product.Quantity = Product.Quantity - 1 where pid=2;
Insert into Delivery(cid,pid,quantity) values (2,2,1);	
Commit;	
	Commit;

4) Two Admins T1 and T2 review the request of 2 different Transporters one with id=2 and the other with id =3

Transaction T1	Transaction T2
Read Transporter.ActiveStatus(2)	
	Read Transporter.ActiveStatus(3)
Set Transporter.ActiveStatus(2)=1	
	Set Transporter.ActiveStatus(3)=2
Write Transporter.ActiveStatus(2)	
commit;	
	Write Transporter.ActiveStatus(3)
	commit;

User Walkthrough:

1. **Admin's Perspective**

TransporterRequests shown to admin

Analytics Page
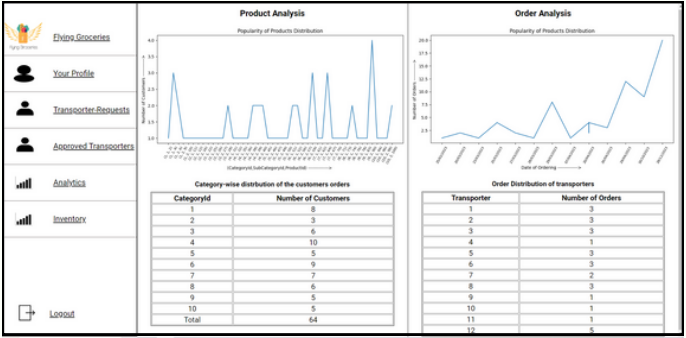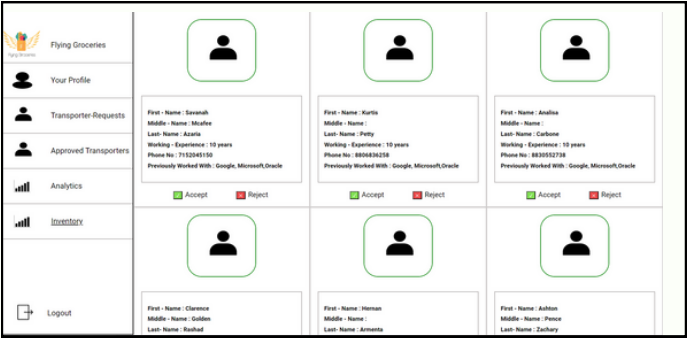
Admin adds Products to database here

Profile Page of admin

Admin adds Products to database here

Admin Login

Navigates to admin login page
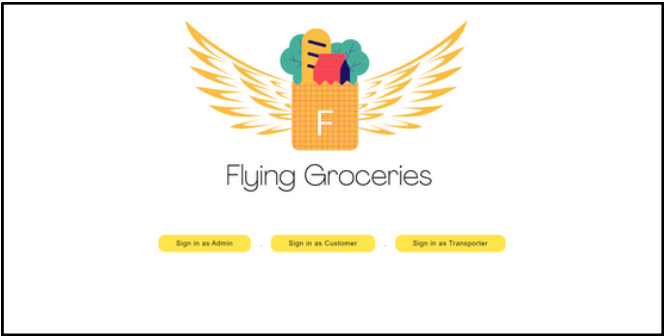
2. Customer's Perspective
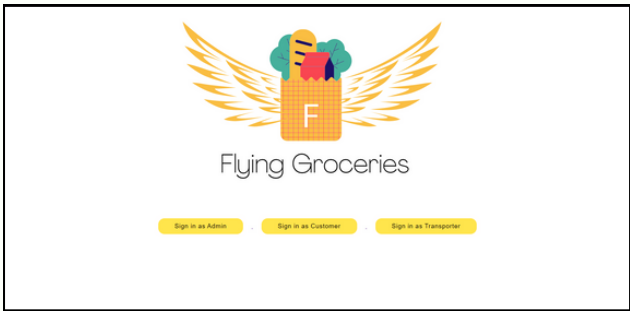
Customer Navigates to customer page
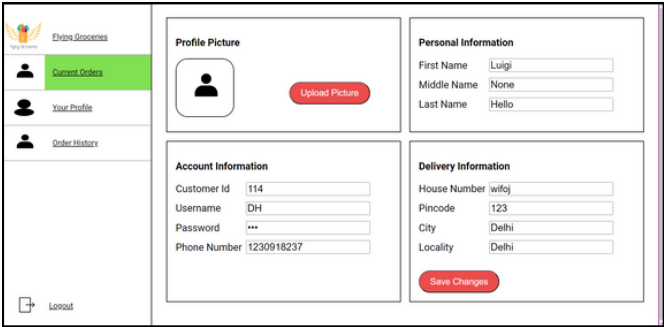
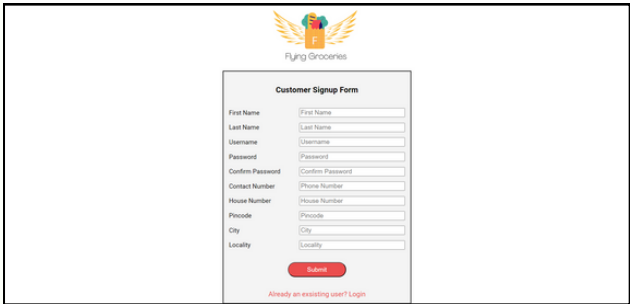Customer Profile Page
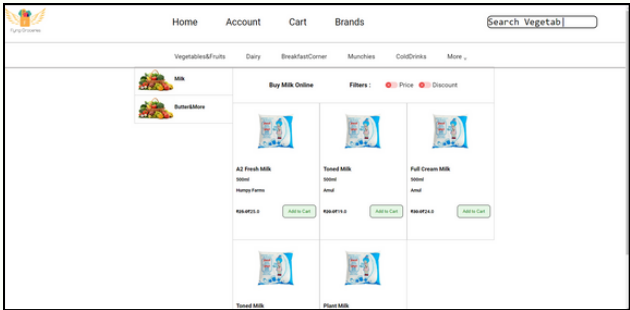
Customer views products from different Subcategories
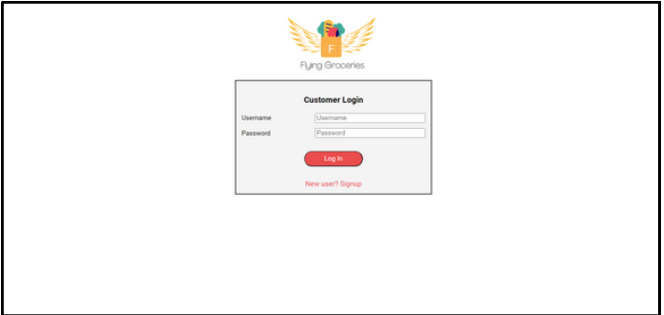
Customer Login Page
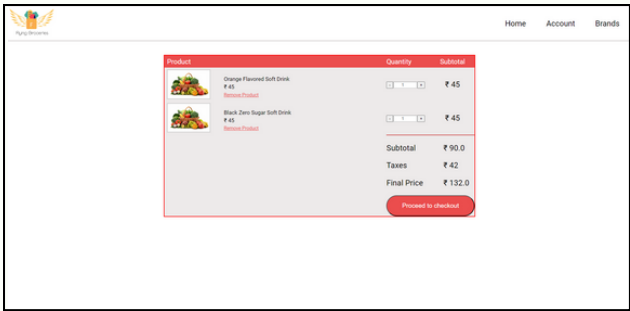
Customer cart page
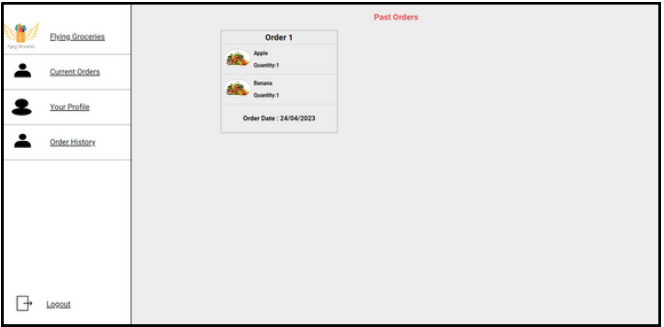
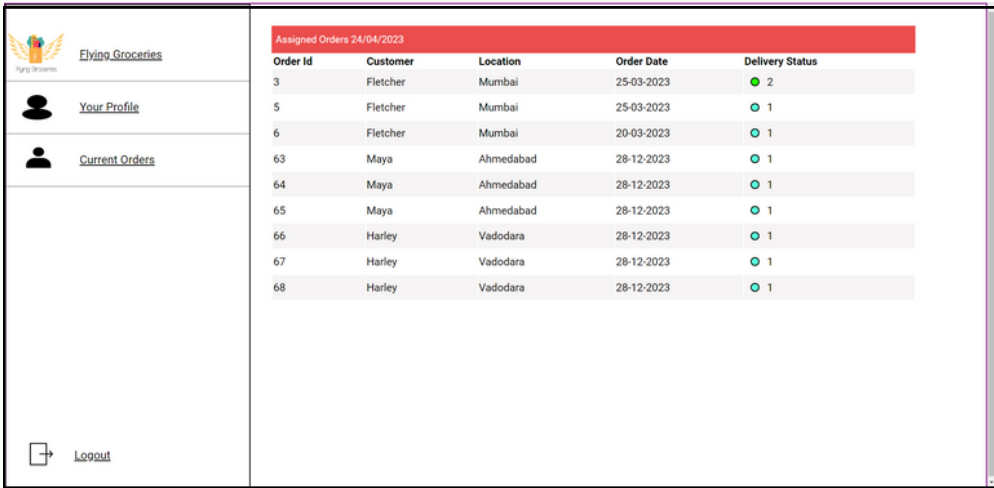
Past orders page

Customer Signup page
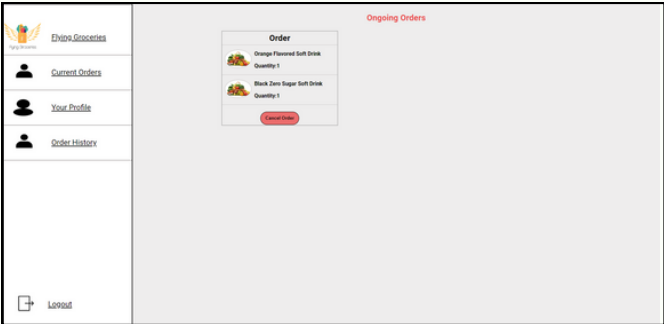
Current orders page
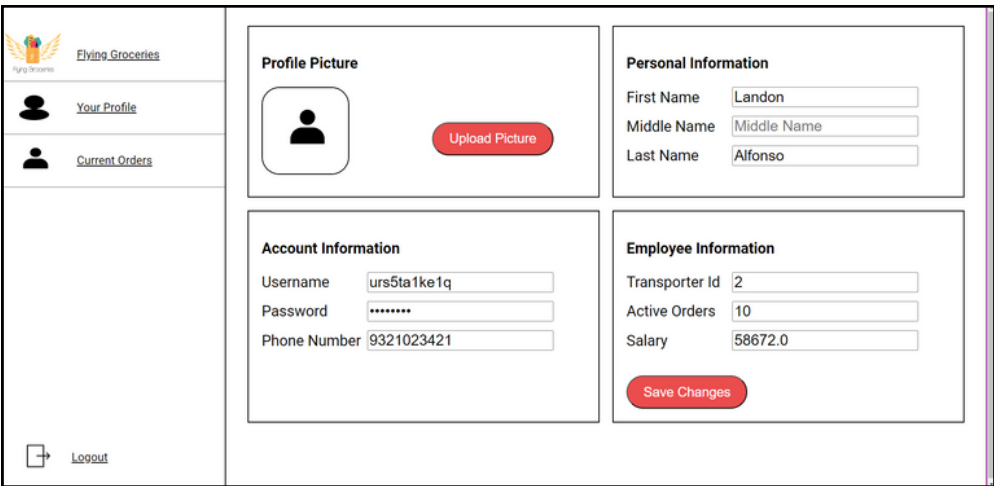
DBMS PROJECT REPORT

3. Transporter's Perspective
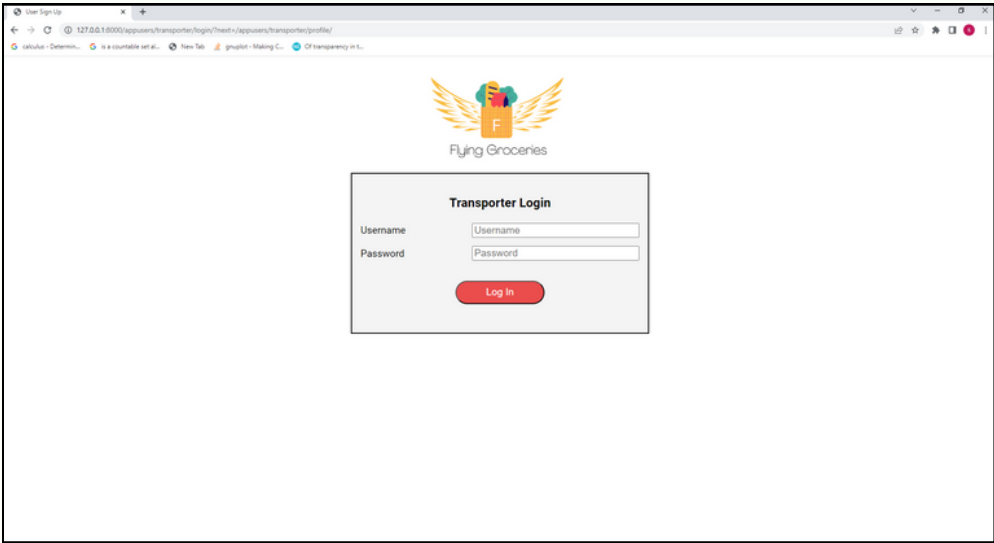
Transporter Login Page

Transporter can view their pending/delivered orders

Transporter Profile Page