BlinkCart

An Online Retail Store Management System

Built as a course project for

CSE202: Fundamentals of Database Management

Systems Project Progress Report and Documentation

SYED YASIR ALI (2022530)
SAHIL 2022427

Date-

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Deadline 1

Project Scope for the Business requirements

January 22, 2024

Project Scope For Business Requirements

➔ Introduction:

In the ever-evolving landscape of e-commerce, the need for a robust, user-friendly, and efficient Online Retail Store Management System is paramount. Introducing BlinkCart, an innovative solution designed to revolutionize the online retail experience for both customers and administrators. BlinkCart aims to provide a seamless and secure platform that not only facilitates smooth customer transactions but also empowers administrators with the tools they need to efficiently manage their online retail store.

➔ Vision:

BlinkCart envisions a future where online shopping is not just a transaction but an immersive and enjoyable experience. The vision is to create a platform that transcends traditional online retail by combining cutting-edge technology, user-centric design, and comprehensive management features.

➔ Overview Of Industry:

The online retail industry, in which BlinkCart operates, is a dynamic and rapidly evolving sector that has seen remarkable growth and transformation. As consumers increasingly turn to online platforms for their shopping needs, BlinkCart enters a competitive landscape characterized by a diverse range of e-commerce platforms. The industry is marked by continuous technological innovations, such as AI-driven personalization, augmented reality features, and enhanced security measures in payment solutions. The prominence of mobile commerce and the global reach of e-commerce are significant trends, requiring BlinkCart to ensure a seamless and secure experience across various devices and cater to a diverse international customer base. The logistics and supply chain play a crucial role, with efficient processes and timely deliveries being essential for customer satisfaction. Sustainability is gaining importance, and companies, including BlinkCart, are adopting eco-friendly practices to address environmental concerns.

The impact of the COVID-19 pandemic has further accelerated the industry's shift towards online shopping, emphasizing the need for resilient and adaptive e-commerce strategies. BlinkCart aims to navigate these industry dynamics by offering a user-centric, secure, and innovative online retail store management system to meet the evolving needs of both customers and administrators.

➔ Objectives:

- Create a user-friendly and responsive online platform for customers to browse, search, and purchase products.
- Develop a robust administrative backend for managing products, orders, inventory, and customer data.
- Implement secure payment gateways to facilitate online transactions.
- Enable order tracking and communication channels for customers.
- Implement reporting and analytics tools for administrators to make informed business decisions.

➔ Features:

★ Customer-Facing Features:

- ❖ User Registration and Authentication
- ❖ Product Browsing and Search Functionality
- ❖ Shopping Cart and Checkout Process
- ❖ Multiple Payment Options
- ❖ Order Tracking and Notifications
- ❖ Customer Support Chat
- ❖ Product Reviews and Ratings
- ❖ User Account Management

★ Admin-Facing Features:

- ❖ Product Management
- ❖ Inventory Management
- ❖ Order Processing and Fulfillment
- ❖ Customer Management
- ❖ Sales Reporting and Analytics
- ❖ User Access Control
- ❖ Content Management System (CMS) for Website Content
- ❖ Add/Modify discounts
- ❖ View transaction history
- ❖ Change Price/Quantity

★ Supplier-Facing Features:

- ❖ Manage product listings, including descriptions, prices, and availability.
- ❖ Monitor sales statistics and update product information.
- ❖ Apply special deals or combos.
- ❖ Efficiently manage and update product inventory.
- ❖ View products currently on sale.

★ Delivery agent-Facing Features:

- ❖ View and manage assigned delivery orders.
- ❖ Set activity status to indicate availability.
- ❖ Access customer feedback for continuous improvement.

➔ Entities

Admin: Attributes: AdminID (Primary Key), Username, Password, Name, Email.

- Admins are pivotal stakeholders in the system, serving as store managers responsible for overseeing and managing the inventory. They play a crucial role in maintaining the store's database, ensuring accurate and up-to-date information about products, orders, and customers.

Customer: Attributes: Username(Primary Key), Password, Name,Mobile Email,Address.

- Customers are the primary users interacting with the system. They are required to create an account and log in to use the application. Customers engage in activities such as browsing products, placing orders, and providing feedback through product reviews. Each customer has a unique identifier (CustomerID) for personalized transactions.

Supplier: Attributes: SupplierID (Primary Key), Name, ContactInfo.

- Suppliers are external entities responsible for supplying products to the store. Multiple suppliers may provide the same products. Their role involves ensuring a steady flow of inventory by delivering products to the store, contributing to the overall availability and variety of items.

DeliveryAgent: Attributes: DeliveryAgentID (Primary Key), Name, ContactInfo,Rating.

- Delivery agents are individuals assigned to specific orders. They play a critical role in the order fulfillment process, responsible for delivering products to customers. Each delivery agent has a unique identifier (DeliveryAgentID), and their performance may be subject to customer reviews.

Product: Attributes: ProductID (Primary Key), Name, Description, Price, StockQuantity.

- Products represent the items supplied to the store by suppliers. Each product is characterized by a set of attributes, including a unique identifier (ProductID), name, description, price, and stock quantity. Products form the core inventory that customers browse and purchase.

Order: Attributes: OrderID (Primary Key), CustomerID (Foreign Key), DeliveryAgentID (Foreign Key), OrderDate, Status, TotalAmount.

- Orders are entities used to track and manage customer purchases. Each order is associated with a specific customer and a delivery agent. Order details include the order date, status (e.g., processing, delivered), and the total amount. This entity facilitates efficient order processing and delivery.

Product Review: Attributes: username (Primary Key), ProductID (Foreign Key), Rating, Comments, ReviewDate.

- Description: Product reviews are feedback provided by customers on specific products they have purchased. Customers can rate products and provide comments, influencing the overall perception of product quality and helping other customers make informed decisions.

Delivery Agent Reviews: Attributes: username (Primary Key), DeliveryAgentID (Foreign Key), Rating, Comments, ReviewDate.

- Description: Delivery agent reviews capture customer feedback on the performance of delivery agents. Customers can rate delivery agents and share comments based on their experiences, contributing to the evaluation of delivery services.

Payment: Attributes: CustomerID (Primary Key),Order ID(Foreign Key) Payment Method,Amount,Date.

- The Payment entity plays a pivotal role in facilitating secure and seamless financial transactions within the BlinkCart system. It records the financial interactions between customers and the system during the checkout process. Each payment is associated with a specific order, linking the financial transaction to the products or services being purchased.

➔ Relationships & Cardinality Constraints

Admin-Product Relationship:

- Type: Binary
- Cardinality: One-to-Many (1:M)
- Description: An admin manages multiple products, and each product is managed by only one admin.

Customer-Order Relationship:

- Type: Binary
- Cardinality: One-to-Many (1:M)
- Description: A customer places multiple orders, and each order is placed by only one customer.

Supplier-Product Relationship:

- Type: Binary
- Cardinality: One-to-Many (1:M)
- Description: A supplier supplies multiple products, and each product is supplied by only one supplier.

DeliveryAgent-Order Relationship:

- Type: Binary
- Cardinality: One-to-Many (1:M)
- Description: A delivery agent handles multiple orders, and each order is handled by only one delivery agent.

Customer-ProductReview Relationship:

- Type: Binary
- Cardinality: One-to-Many (1:M)
- Description: A customer writes reviews for multiple products, and each product review is written by only one customer.

Customer-DeliveryAgentReview Relationship:

- Type: Binary
- Cardinality: One-to-Many (1:M)
- Description: A customer writes reviews for multiple delivery agents, and each delivery agent review is written by only one customer.

Customer-Payment Relationship:

- Type: Binary
- Cardinality: One-to-Many (1:M)
- Description: A customer makes multiple payments, and each payment is made by only one customer.

Order-Payment Relationship:

- Type: Binary
- Cardinality: One-to-One (1:1)
- Description: Each order involves one and only one payment, creating a direct binary relationship between orders and payments.

Customer-Product-Supplier Relationship:

- Type: Ternary
- Cardinality: Many-to-Many-to-Many (M:M:M)
- Description: Customers purchase products, and each product is supplied by one or more suppliers. This relationship involves three entities: Customer, Product, and Supplier. Each customer can purchase multiple products, each product can be purchased by multiple customers, and each product is supplied by one or more suppliers.

Most cardinality constraints have been mentioned above.Some other constraints are:

Product - Supplier:

- Requirement: Every product must have an associated supplier.

Order - Customer:

- Requirement: An order must be linked to a customer.

Order - Delivery Agent:

- Requirement: An order cannot exist without being associated with a delivery agent.

Product Review - Customer has purchased Product:

- Requirement: Customers are only allowed to review products they have purchased.

Delivery Agent Review - Delivery Agent delivered to Customer:

- Requirement: Customers can only provide reviews for delivery agents who have delivered orders to them.

Integrity Constraints:

Price Range Constraint:

- Requirement: Product prices must fall within a predefined range to maintain pricing consistency and adherence to business policies.

Quantity Constraint:

- Requirement: The quantity of a product in an order must be a positive integer, ensuring valid and meaningful order information.

Unique Username Constraint:

- Requirement: Usernames and ID for customers, suppliers, and delivery agents must be unique to avoid ambiguity and ensure proper identification.

Referential Constraints:

Foreign Key Constraint:

- Requirement: Any reference to a customer, supplier, or delivery agent in the database must correspond to a valid and existing entity.

Product-Category Relationship:

- Requirement: Each product must belong to a specific category, ensuring proper organization and search functionality.

Consistency Constraints:

Delivery Timeframe Constraint:

- Requirement: The delivery time for an order must be within realistic and predefined limits to manage customer expectations and ensure timely service.

Product Availability Constraint:

- Requirement: Products listed in the order must be currently available in the supplier's catalog to avoid issues related to out-of-stock items.

Security Constraints:

Password Encryption Constraint:

- Requirement: All user passwords, including those of customers, suppliers, and delivery agents, must be securely encrypted to protect sensitive information.

Access Log Constraint:

- Requirement: Maintain a log of user access and modifications to sensitive data for auditing and security purposes.

Performance Constraints:

Search Index Constraint:

- Requirement: Implement efficient indexing mechanisms for product searches to enhance search speed and overall application performance.

Order Processing Time Constraint:

- Requirement: Set performance benchmarks for order processing times to ensure timely and efficient order fulfillment.

➔Implementation

I plan to make a full-stack project with a backend and a frontend. According to the requirements, I plan to use the following tools and technologies!!

1. Front-End Development:

- Technology: HTML, CSS, JavaScript, and a front-end framework like React, Angular.
- Flow:
    - Design the user interface (UI) with a focus on usability and responsiveness.
    - Use a front-end framework to create dynamic and interactive components.
    - Implement user authentication features for customers, suppliers, and delivery agents.
    - Ensure a consistent and user-friendly design.

2. Back-End Development:

- Technology: Django, or Flask.
- Database: MySQL
- Flow:
    - Set up server-side logic to handle requests and responses.
    - Design and create database tables based on the specified constraints.
    - Implement business logic to enforce constraints, such as price range checks and quantity validations.
    - Implement user authentication and authorization mechanisms.
    - Ensure data integrity with the enforcement of referential constraints.

➔Expected Flow And Timeline

- Design of Conceptual Model of your project and converting it to Relational model

  By 31st Jan 2024 !!

- Database schema and indexes creation (with integrity constraints) and data insertion (populate simulated data satisfying the constraints)

  By 14 Feb 2024 !!

- Write and execute at least TEN SQL queries pertaining to your application involving various relational algebraic operations supporting the application features involving database access and manipulation.

  By 20 Feb 2024 !!

- Write applications (embedded SQL in your favorite programming language) for ‘Ordering the items’ and ‘inventory or customer analysis’ and define at least TWO triggers which checks the database conditions and takes appropriate actions as desired by your application (for example, a/cv blocking after 3 login failures, etc).

  By 30th March 2024 !!

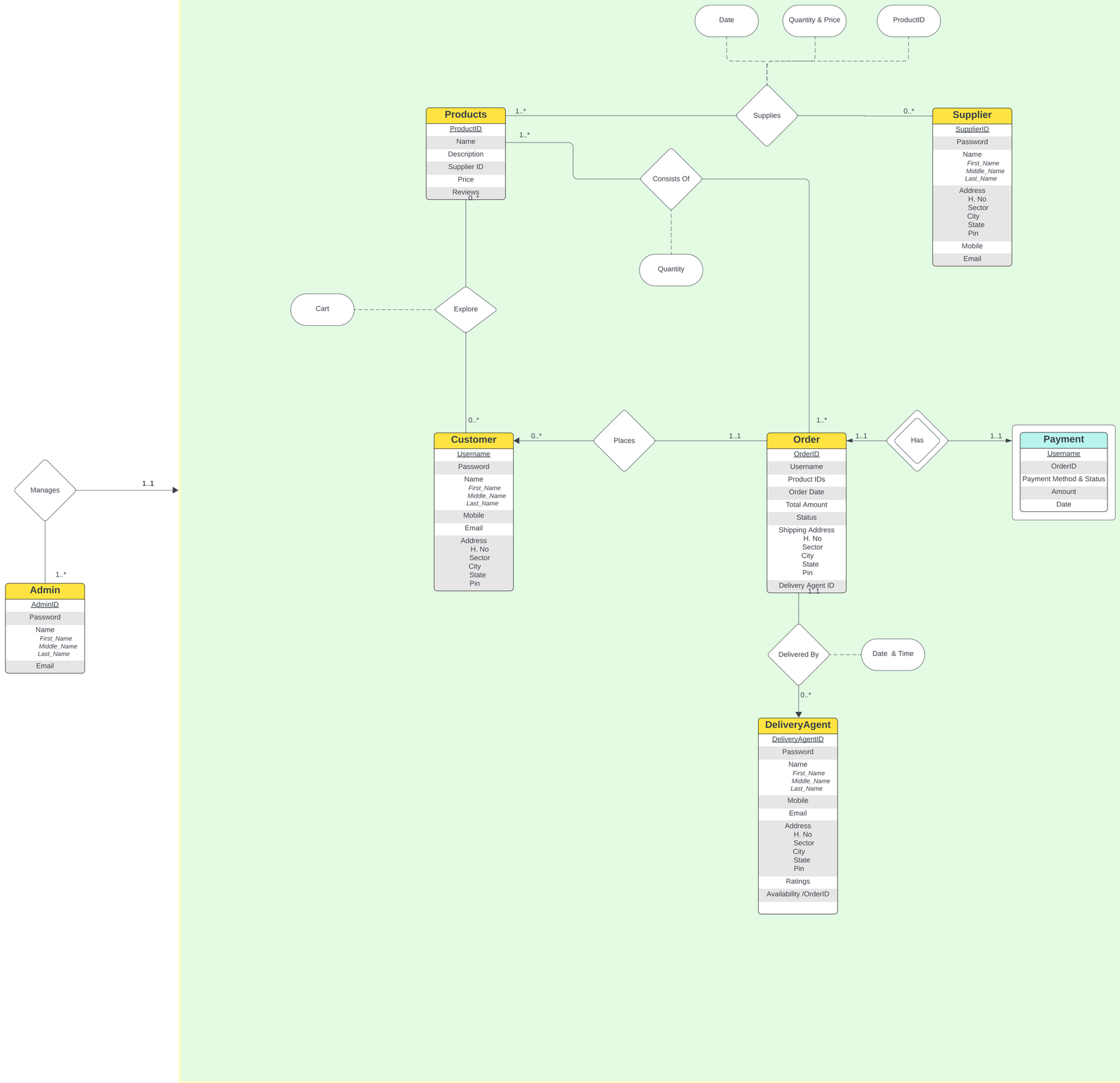
- Write and execute db transactions (including conflicting ones) and check the effect on your db.

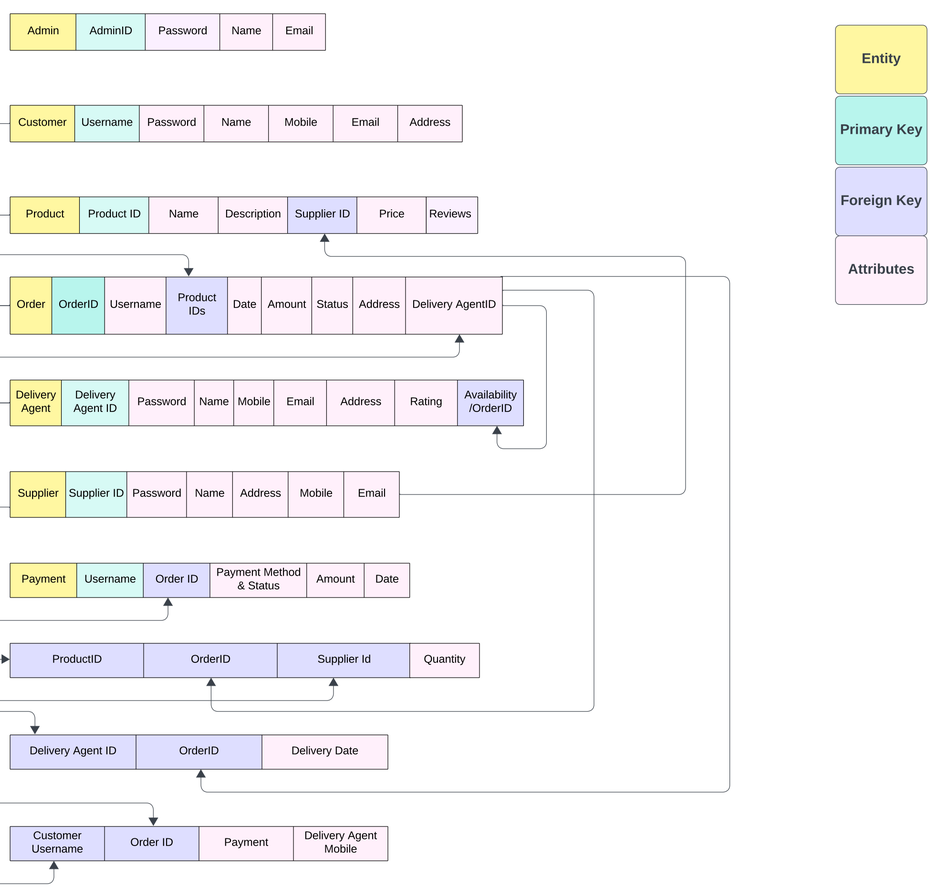
  By 12th April 2024 !!

Future Plans

The Plan is to make an optimized full stack project with security features & protocols and deploy on some hosting Platform like AWS after fully testing. !!!

DEADLINE2


Date
Quantity & Price
ProductID
Supplies
Products
ProductID
Name
Description
Supplier ID
Price
Reviews
1..*
0..*
Supplier
SupplierID
Password
Name
First_Name
Middle_Name
Last_Name
Address
H. No
Sector
City
State
Pin
Mobile
Email
Consists Of
Quantity
Cart
Explore
Manages
1..1
Admin
AdminID
Password
Name
First_Name
Middle_Name
Last_Name
Email
Customer
Username
Password
Name
First_Name
Middle_Name
Last_Name
Mobile
Email
Address
H. No
Sector
City
State
Pin
Places
Order
OrderID
Username
Product IDs
Order Date
Total Amount
Status
Shipping Address
H. No
Sector
City
State
Pin
Delivery Agent ID
Has
Payment
Username
OrderID
Payment Method & Status
Amount
Date
Delivered By
Date & Time
DeliveryAgent
DeliveryAgentID
Password
Name
First_Name
Middle_Name
Last_Name
Mobile
Email
Address
H. No
Sector
City
State
Pin
Ratings
Availability /OrderID


Admin
AdminID
Password
Name
Email
Customer
Username
Password
Name
Mobile
Email
Address
Product
Product ID
Name
Description
Supplier ID
Price
Reviews
Order
OrderID
Username
Product IDs
Date
Amount
Status
Address
Delivery AgentID
Delivery Agent
Delivery Agent ID
Password
Name
Mobile
Email
Address
Rating
Availability /OrderID
Supplier
Supplier ID
Password
Name
Address
Mobile
Email
Payment
Username
Order ID
Payment Method & Status
Amount
Date
ProductID
OrderID
Supplier Id
Quantity
Delivery Agent ID
OrderID
Delivery Date
Customer Username
Order ID
Payment
Delivery Agent Mobile

Entity
Primary Key
Foreign Key
Attributes

DEADLINE 3

Suppliers

Query to create the Suppliers table

```
CREATE TABLE Suppliers (
    SupplierID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    SuppliersAddress VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL
);
```

Insert

```
INSERT INTO Suppliers (SupplierID, Password, Name, SuppliersAddress, Mobile, Email)
VALUES
    (1, 'supplier1pass', 'Supplier 1', '123 Main St', '1234567890', 'supplier1@example.com'),
    (2, 'supplier2pass', 'Supplier 2', '456 Elm St', '0987654321', 'supplier2@example.com'),
    (3, 'supplier3pass', 'Supplier 3', '789 Oak St', '5551234567', 'supplier3@example.com');

INSERT INTO Suppliers (SupplierID, Password, Name, SuppliersAddress, Mobile, Email)
VALUES (4, 'supplier4pass', 'Supplier 4', '123 Main St', '5678901234', 'supplier4@example.com');

INSERT INTO Suppliers (SupplierID, Password, Name, SuppliersAddress, Mobile, Email)
VALUES (2, 'supplier2pass', 'Supplier 2', '456 Elm St', '0987654321', 'supplier2@example.com');
```

Delete

```
DELETE FROM Suppliers WHERE SupplierID = 1;

DELETE FROM Suppliers WHERE Mobile = '0987654321';
```

Products

Query to create the Products table

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Description TEXT,
    SupplierID INT NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    Review VARCHAR(255),
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
);
```

Insert

```
INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES (1, 'Product 1', 'Description of Product 1', 1, 10.99, 'Nice Product');

INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES (1, 'Product 1', 'Description of Product 1', 2, 10.99, 'Nice Product');

INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES (1, 'Product 1', 'Description of Product 1', 2, 10.99, 'Nice Product');

INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES (2, 'Product 2', 'Description of Product 2', 2, 10.99, 'Nice Product');

INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES (4, 'Product 4', 'Description of Product 4', 3, 10.99, 'Nice Product');

INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES (5, 'Product 5', 'Description of Product 5', 3, 100.99, 'Nice Product');
```

```
INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES (6, 'Product 6', 'Description of Product 6', 3, 100.99, NULL);
```

Delete

```
DELETE FROM Products WHERE SupplierID is 4;

DELETE FROM Products WHERE Price > 100.00;

DELETE FROM Products WHERE Review IS NULL;
```

Customers

Query to create the Customers table

```
CREATE TABLE Customers (
    Username VARCHAR(255) PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Pin_Code INT NOT NULL
);
```

Insert

```
INSERT INTO Customers (Username, Password, Name, Mobile, Email, Address, Pin_Code)
VALUES ('john_doe', 'password123', 'John Doe', '1234567890', 'john@example.com', '123 Main
St', 12345);
```

```
INSERT INTO Customers (Username, Password, Name, Mobile, Email, Address, Pin_Code)
VALUES ('john_doe1234@doe', 'password123', 'John Doe', '1234567890', 'john@example.com',
'123 Main St', 12345);
```

```
INSERT INTO Customers (Username, Password, Name, Mobile, Email, Address, Pin_Code)
VALUES
    ('jane_doe', 'password456', 'Jane Doe', '9876543210', 'jane@example.com', '456 Elm St',
54321),
    ('alice_smith', 'password789', 'Alice Smith', '5551234567', 'alice@example.com', '789 Oak St',
67890);
```

Delete

```
DELETE FROM Customers WHERE Username = 'john_doe';
```

```
DELETE FROM Customers WHERE Pin_Code = 54321;
```

Orders

Query to create the Orders table

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY NOT NULL,
    Username VARCHAR(255) NOT NULL,
    ProductIDs VARCHAR(255) NOT NULL, -- Adjust the length as needed
    OrderDate DATE NOT NULL,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    Status VARCHAR(255) NOT NULL,
    ShippingAddress VARCHAR(255) NOT NULL,
    Pin_Code INT NOT NULL,
    DeliveryAgentID INT,
    FOREIGN KEY (Username) REFERENCES Customers(Username)
);
```

Insert

```
INSERT INTO Orders (OrderID, Username, ProductIDs, OrderDate, TotalAmount, Status,
ShippingAddress, Pin_Code, DeliveryAgentID)
VALUES (1, 'alice_smith', '1,6', '2024-02-12', 50.99, 'Pending', '789 Oak St', 67890, NULL),
       (2, 'john_doe1234@doe', '1,2', '2024-02-12', 35.99, 'Pending', '123 Maple Ave', 12345,
NULL);
```

Delete

```
DELETE FROM Orders
WHERE Username = 'alice_smith';

DELETE FROM Orders
```

```
WHERE OrderID = 1;
```

Payments

Query to make Payments table

```
CREATE TABLE Payments (
    Username VARCHAR(255) NOT NULL,
    OrderID INT NOT NULL,
    PaymentMethodStatus VARCHAR(255) NOT NULL,
    Amount DECIMAL(10, 2) NOT NULL,
    Date DATE NOT NULL,
    FOREIGN KEY (Username) REFERENCES Customers(Username),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

Insert

```
INSERT INTO Payments (Username, OrderID, PaymentMethodStatus, Amount, Date)
VALUES ('alice_smith', 2, 'Paid', 50.99, '2024-02-12');
```

Delete;

```
DELETE FROM Payments
WHERE OrderID = 2;
```

DeliveryAgent

Query to make DeliveryAgents table

```
CREATE TABLE DeliveryAgents (
    DeliveryAgentID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Ratings FLOAT,
    Availability BOOLEAN NOT NULL,
    OrderID INT,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

Insert

```
INSERT INTO DeliveryAgents (DeliveryAgentID, Password, Name, Mobile, Email, Ratings,
Availability, OrderID)
VALUES (1, 'agent123', 'John Smith', '123-456-7890', 'john@example.com', 4.5, true, NULL);

INSERT INTO DeliveryAgents (DeliveryAgentID, Password, Name, Mobile, Email, Ratings,
Availability, OrderID)
VALUES (2, 'agent456', 'Alice Johnson', '987-654-3210', 'alice@example.com', 4.2, true, NULL);

INSERT INTO DeliveryAgents (DeliveryAgentID, Password, Name, Mobile, Email, Ratings,
Availability, OrderID)
VALUES (3, 'pass789', 'Bob Anderson', '555-555-5555', 'bob@example.com', 4.7, false, NULL);
```

Delete

```
DELETE FROM DeliveryAgents
WHERE DeliveryAgentID = 1;
```

```
DELETE FROM DeliveryAgents
WHERE Ratings > 4.6;
```

Admins

Query to make Admins table

```
CREATE TABLE Admins (
    AdminID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL
);
```

Insert

```
INSERT INTO Admins (AdminID, Password, Name, Email)
VALUES (1, 'admin123', 'John Doe', 'john@example.com');

INSERT INTO Admins (AdminID, Password, Name, Email)
VALUES (2, 'admin456', 'Jane Smith', 'jane@example.com');

INSERT INTO Admins (AdminID, Password, Name, Email)
VALUES (3, 'securePass123', 'Alice Johnson', 'alice@example.com');
```

Delete

```
DELETE FROM Admins
WHERE Email = 'john@example.com';


DELETE FROM Admins
WHERE Name = 'Jane Smith';
```

DEADLINE 4

-- Suppliers

-- Query to create the Suppliers table

```
CREATE TABLE Suppliers (
    SupplierID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    SuppliersAddress VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL
);
```

-- Insert

```
INSERT INTO Suppliers (SupplierID, Password, Name, SuppliersAddress, Mobile, Email)
VALUES
    (1, 'supplier1pass', 'Supplier 1', '123 Main St', '1234567890', 'supplier1@example.com'),
    (2, 'supplier2pass', 'Supplier 2', '456 Elm St', '0987654321', 'supplier2@example.com'),
    (3, 'supplier3pass', 'Supplier 3', '789 Oak St', '5551234567', 'supplier3@example.com'),
    (4, 'supplier4pass', 'Supplier 4', '321 Pine St', '9876543210', 'supplier4@example.com'),
    (5, 'supplier5pass', 'Supplier 5', '678 Maple St', '5555555555', 'supplier5@example.com'),
    (6, 'supplier6pass', 'Supplier 6', '987 Cedar St', '1112223333', 'supplier6@example.com'),
    (7, 'supplier7pass', 'Supplier 7', '456 Birch St', '4445556666', 'supplier7@example.com'),
    (8, 'supplier8pass', 'Supplier 8', '234 Oak St', '7778889999', 'supplier8@example.com'),
    (9, 'supplier9pass', 'Supplier 9', '890 Pine St', '8889990000', 'supplier9@example.com'),
    (10, 'supplier10pass', 'Supplier 10', '567 Elm St', '9990001111', 'supplier10@example.com');
```

-- Products

-- Query to create the Products table

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Description TEXT,
    SupplierID INT NOT NULL,
    Price DECIMAL(10, 2) CHECK (Price >= 0) NOT NULL,  -- Modified constraint
    Review VARCHAR(255),
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
);
```

-- Insert

```
INSERT INTO Products (ProductID, Name, Description, SupplierID, Price, Review)
VALUES
    (1, 'Product 1', 'Description of Product 1', 1, 10.99, 'Nice Product'),
    (2, 'Product 2', 'Description of Product 2', 2, 20.99, 'Great Product'),
    (3, 'Product 3', 'Description of Product 3', 3, 15.99, 'Amazing Product'),
    (4, 'Product 4', 'Description of Product 4', 5, 12.99, 'Excellent Product'),
    (5, 'Product 5', 'Description of Product 5', 5, 18.99, 'Awesome Product'),
    (6, 'Product 6', 'Description of Product 6', 1, 25.99, 'Fantastic Product'),
    (7, 'Product 7', 'Description of Product 7', 7, 14.99, 'Superb Product'),
    (8, 'Product 8', 'Description of Product 8', 2, 22.99, 'Outstanding Product'),
    (9, 'Product 9', 'Description of Product 9', 9, 19.99, 'Terrific Product'),
    (10, 'Product 10', 'Description of Product 2', 10, 30.99, 'Splendid Product');
```

-- Customers

-- Query to create the Customers table

```
CREATE TABLE Customers (
    Username VARCHAR(255) PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Pin_Code INT NOT NULL
);
```

-- Insert

```
INSERT INTO Customers (Username, Password, Name, Mobile, Email, Address, Pin_Code)
VALUES
    ('john_doe', 'password123', 'John Doe', '1234567890', 'john@example.com', '123 Main St',
12345),
    ('jane_doe', 'password456', 'Jane Doe', '9876543210', 'jane@example.com', '456 Elm St',
54321),
    ('alice_smith', 'password789', 'Alice Smith', '5551234567', 'alice@example.com', '789 Oak St',
67890),
    ('bob_smith', 'password987', 'Bob Smith', '5555555555', 'bob@example.com', '456 Maple St',
45678),
    ('lisa_johnson', 'password135', 'Lisa Johnson', '1112223333', 'lisa@example.com', '789 Pine
St', 98765),
    ('tom_anderson', 'password246', 'Tom Anderson', '9998887777', 'tom@example.com', '321
Oak St', 23456),
    ('emily_wilson', 'password357', 'Emily Wilson', '3332221111', 'emily@example.com', '567 Elm
St', 87654),
```

```
    ('michael_brown', 'password468', 'Michael Brown', '6667778888', 'michael@example.com', '890 Cedar St', 34567),
    ('susan_jones', 'password579', 'Susan Jones', '4443332222', 'susan@example.com', '654 Birch St', 78901),
    ('peter_clark', 'password680', 'Peter Clark', '8889990000', 'peter@example.com', '987 Maple St', 90123);
```

-- DeliveryAgents

-- Query to make DeliveryAgents table

```
CREATE TABLE DeliveryAgents (
    DeliveryAgentID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Ratings FLOAT CHECK (Ratings >= 0 AND Ratings <= 5),
    Availability BOOLEAN NOT NULL,
    OrderID VARCHAR(255)
);
```

-- Insert

```
INSERT INTO DeliveryAgents (DeliveryAgentID, Password, Name, Mobile, Email, Ratings, Availability, OrderID)
VALUES
    (1, 'agent123', 'John Smith', '123-456-7890', 'john@example.com', 4.5, true,'5,8,9' ),
    (2, 'agent456', 'Alice Johnson', '987-654-3210', 'alice@example.com', 4.2, true, '6'),
    (3, 'agent789', 'Bob Anderson', '555-555-5555', 'bob@example.com', 4.7, false, '3'),
    (4, 'agent987', 'Jane Smith', '777-777-7777', 'jane@example.com', 4.0, true, NULL),
    (5, 'agent567', 'Michael Brown', '111-222-3333', 'michael@example.com', 4.8, true, '7'),
    (6, 'agent234', 'Lisa Johnson', '444-555-6666', 'lisa@example.com', 4.3, true, '10'),
    (7, 'agent890', 'Tom Anderson', '999-888-7777', 'tom@example.com', 4.6, true, NULL),
    (8, 'agent765', 'Emily Wilson', '666-555-4444', 'emily@example.com', 4.1, false, NULL),
    (9, 'agent321', 'Susan Jones', '888-999-0000', 'susan@example.com', 4.4, true, NULL),
    (10, 'agent543', 'Peter Clark', '222-333-4444', 'peter@example.com', 4.9, true, NULL);
```

.

-- Orders

-- Query to create the Orders table

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY NOT NULL,
    Username VARCHAR(255) NOT NULL,
    ProductIDs VARCHAR(255) NOT NULL, -- Adjust the length as needed
    OrderDate DATE NOT NULL,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    Status VARCHAR(255) NOT NULL,
    ShippingAddress VARCHAR(255) NOT NULL,
    Pin_Code INT NOT NULL,
    DeliveryAgentID INT,
    FOREIGN KEY (Username) REFERENCES Customers(Username),
    FOREIGN KEY (DeliveryAgentID) REFERENCES DeliveryAgents(DeliveryAgentID)
);
```

-- Insert

```
INSERT INTO Orders (OrderID, Username, ProductIDs, OrderDate, TotalAmount, Status,
ShippingAddress, Pin_Code, DeliveryAgentID)
VALUES
    (1, 'john_doe', '1,2', '2024-02-12', 35.99, 'Pending', '123 Maple Ave', 12345, Null),
    (2, 'john_doe', '3,4', '2024-02-13', 28.99, 'Pending', '123 Maple Ave', 12345, Null),
    (3, 'peter_clark', '1,3,5', '2024-02-21', 68.99, 'Shipped', '987 Maple St', 90123, 3),
    (4, 'bob_smith', '1,3,4', '2024-02-15', 58.99, 'Pending', '456 Maple St', 45678, Null),
```

```
    (5, 'susan_jones', '3,5', '2024-02-20', 48.99, 'Shipped', '654 Birch St', 78901, 1),
    (6, 'tom_anderson', '1,2,3', '2024-02-17', 46.99, 'Shipped', '321 Pine St', 23456, 2),
    (7, 'emily_wilson', '5,6', '2024-02-18', 35.99, 'Shipped', '567 Elm St', 87654, 5),
    (8, 'susan_jones', '2,5', '2024-02-20', 48.99, 'Shipped', '654 Birch St', 78901, 1),
    (9, 'susan_jones', '2,3,5', '2024-02-20', 48.99, 'Shipped', '654 Birch St', 78901, 1),
    (10, 'peter_clark', '1,3,4,5', '2024-02-21', 68.99, 'Shipped', '987 Maple St', 90123, 6);
```

-- Payments

-- Query to make Payments table

```
CREATE TABLE Payments (
    Username VARCHAR(255) NOT NULL,
    OrderID INT NOT NULL,
    PaymentMethodStatus VARCHAR(255) NOT NULL,
    Amount DECIMAL(10, 2) NOT NULL,
    Date DATE NOT NULL,
    FOREIGN KEY (Username) REFERENCES Customers(Username),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

-- Insert

```
INSERT INTO Payments (Username, OrderID, PaymentMethodStatus, Amount, Date)
VALUES
    ('john_doe', 1, 'Paid', 35.99, '2024-02-12'),
    ('john_doe', 2, 'Paid', 28.99, '2024-02-13'),
    ('peter_clark', 3, 'Pending', 68.99, '2024-02-21'),
    ('bob_smith', 4, 'Pending', 58.99, '2024-02-15'),
    ('susan_jones', 5, 'Pending', 48.99, '2024-02-20'),
    ('tom_anderson', 6, 'Pending', 46.99, '2024-02-17'),
    ('emily_wilson', 7, 'Pending', 35.99, '2024-02-18'),
    ('susan_jones', 8, 'Pending', 48.99, '2024-02-20'),
    ('susan_jones', 9, 'Pending', 48.99, '2024-02-20'),
    ('peter_clark', 10, 'Pending', 68.99, '2024-02-21');
```

-- Admins

-- Query to make Admins table

```
CREATE TABLE Admins (
    AdminID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL
);
```

-- Insert

```
INSERT INTO Admins (AdminID, Password, Name, Email)
VALUES
    (1, 'admin123', 'John Doe', 'john_admin@example.com'),
    (2, 'admin456', 'Jane Smith', 'jane_admin@example.com'),
    (3, 'admin789', 'Alice Johnson', 'alice_admin@example.com'),
    (4, 'admin987', 'Bob Anderson', 'bob_admin@example.com'),
    (5, 'admin654', 'Michael Brown', 'michael_admin@example.com'),
    (6, 'admin321', 'Lisa Johnson', 'lisa_admin@example.com'),
    (7, 'admin890', 'Tom Anderson', 'tom_admin@example.com'),
    (8, 'admin765', 'Emily Wilson', 'emily_admin@example.com'),
    (9, 'admin234', 'Susan Jones', 'susan_admin@example.com'),
    (10, 'admin543', 'Peter Clark', 'peter_admin@example.com');
```

DEADLINE 5

```
CREATE DATABASE BLINKCART;

CREATE TABLE FailedLoginAttempts (--
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE FailedLoginAttempts (
    username VARCHAR(255) PRIMARY KEY,
    attempt_count INT DEFAULT 0
);

CREATE TABLE user_cart (--
    username VARCHAR(255) PRIMARY KEY,
    total_price DECIMAL(10, 2)
);

-- Suppliers

-- Query to create the Suppliers table

CREATE TABLE Suppliers (--
    SupplierID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    SuppliersAddress VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL
);

-- Insert

INSERT INTO Suppliers (SupplierID, Password, Name, SuppliersAddress,
Mobile, Email)--
VALUES
```

```
    (1, 'supplier1pass', 'Supplier 1', '123 Main St', '1234567890',
'supplier1@example.com'),
    (2, 'supplier2pass', 'Supplier 2', '456 Elm St', '0987654321',
'supplier2@example.com'),
    (3, 'supplier3pass', 'Supplier 3', '789 Oak St', '5551234567',
'supplier3@example.com'),
    (4, 'supplier4pass', 'Supplier 4', '321 Pine St', '9876543210',
'supplier4@example.com'),
    (5, 'supplier5pass', 'Supplier 5', '678 Maple St', '5555555555',
'supplier5@example.com'),
    (6, 'supplier6pass', 'Supplier 6', '987 Cedar St', '1112223333',
'supplier6@example.com'),
    (7, 'supplier7pass', 'Supplier 7', '456 Birch St', '4445556666',
'supplier7@example.com'),
    (8, 'supplier8pass', 'Supplier 8', '234 Oak St', '7778889999',
'supplier8@example.com'),
    (9, 'supplier9pass', 'Supplier 9', '890 Pine St', '8889990000',
'supplier9@example.com'),
    (10, 'supplier10pass', 'Supplier 10', '567 Elm St', '9990001111',
'supplier10@example.com');

-- Products

-- Query to create the Products table

CCREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Description TEXT,
    SupplierID INT NOT NULL,
    Price DECIMAL(10, 2) CHECK (Price > 0) NOT NULL,
    Ratings VARCHAR(255),
    Quantity DECIMAL(10, 2) CHECK (Quantity > 0) NOT NULL,
    ImageLink VARCHAR(255),
    Weight VARCHAR(255) NOT NULL DEFAULT '0',
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
);

-- Step 3: Insert data into the Products table
```

```sql
INSERT INTO Products (ProductID, Name, Description, SupplierID, Price,
Ratings, Quantity, ImageLink, Weight)
VALUES
    (1, 'Product 1', 'Description of Product 1', 1, 10.99, '4.5', 500,
'product1.png', '500g'),
    (2, 'Product 2', 'Description of Product 2', 2, 20.99, '4.8', 1000,
'product2.png', '1 ltr'),
    (3, 'Product 3', 'Description of Product 3', 3, 15.99, '4.3', 250,
'product3.png', '250g'),
    (4, 'Product 4', 'Description of Product 4', 5, 12.99, '4.2', 750,
'product4.png', '750ml'),
    (5, 'Product 5', 'Description of Product 5', 5, 18.99, '4.6', 200,
'product5.png', '200g'),
    (6, 'Product 6', 'Description of Product 6', 1, 25.99, '4.7', 1000,
'product6.png', '1 kg'),
    (7, 'Product 7', 'Description of Product 7', 7, 14.99, '4.4', 2000,
'product7.png', '2 ltr'),
    (8, 'Product 8', 'Description of Product 8', 2, 22.99, '4.9', 500,
'product8.png', '500ml');

-- Customers

-- Query to create the  Customers table

CREATE TABLE Customers (
    Username VARCHAR(255) PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Pin_Code INT NOT NULL,
    AccountStatus VARCHAR(50) DEFAULT 'Active' NOT NULL
);


-- DeliveryAgents

-- Query to make DeliveryAgents table
```

```sql
CREATE TABLE DeliveryAgents (--
    DeliveryAgentID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Mobile VARCHAR(20) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Ratings FLOAT CHECK (Ratings >= 0 AND Ratings <= 5),
    Availability BOOLEAN NOT NULL,
    OrderID VARCHAR(255)
);



-- Insert

INSERT INTO DeliveryAgents (DeliveryAgentID, Password, Name, Mobile,
Email, Ratings, Availability, OrderID)--
VALUES
    (1, 'agent123', 'John Smith', '123-456-7890', 'john@example.com', 4.5,
true,'5,8,9' ),
    (2, 'agent456', 'Alice Johnson', '987-654-3210', 'alice@example.com',
4.2, true, '6'),
    (3, 'agent789', 'Bob Anderson', '555-555-5555', 'bob@example.com',
4.7, false, '3'),
    (4, 'agent987', 'Jane Smith', '777-777-7777', 'jane@example.com', 4.0,
true, NULL),
    (5, 'agent567', 'Michael Brown', '111-222-3333',
'michael@example.com', 4.8, true, '7'),
    (6, 'agent234', 'Lisa Johnson', '444-555-6666', 'lisa@example.com',
4.3, true, '10'),
    (7, 'agent890', 'Tom Anderson', '999-888-7777', 'tom@example.com',
4.6, true, NULL),
    (8, 'agent765', 'Emily Wilson', '666-555-4444', 'emily@example.com',
4.1, false, NULL),
    (9, 'agent321', 'Susan Jones', '888-999-0000', 'susan@example.com',
4.4, true, NULL),
    (10, 'agent543', 'Peter Clark', '222-333-4444', 'peter@example.com',
4.9, true, NULL);
```

```sql
CREATE TABLE cart_items (--
    cart_item_id SERIAL PRIMARY KEY,
    username VARCHAR(255),
    product_id INT,
    product_name VARCHAR(255),
    quantity INT,
    price_per_unit DECIMAL(10, 2),
    total_price DECIMAL(10, 2),
    image_url VARCHAR(255), -- Add column to store image URL
    FOREIGN KEY (username) REFERENCES user_cart(username)
);


CREATE TABLE Orders (--
    OrderID INT PRIMARY KEY,
    Username VARCHAR(255) NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    Status VARCHAR(255) NOT NULL,
    ShippingAddress VARCHAR(255) NOT NULL,
    Pin_Code INT NOT NULL,
    DeliveryAgentID INT,
    payment_method VARCHAR(50),
    FOREIGN KEY (Username) REFERENCES Customers(Username),
    FOREIGN KEY (DeliveryAgentID) REFERENCES
DeliveryAgents(DeliveryAgentID)
);

CREATE TABLE order_counts (--
    order_count SERIAL PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    price_per_unit DECIMAL(10, 2),
    product_name VARCHAR(255),
    product_image_url VARCHAR(255),
    -- Add other fields as needed
    FOREIGN KEY (order_id) REFERENCES Orders(OrderID),
```

```
    FOREIGN KEY (product_id) REFERENCES Products(productid)
);



CREATE TABLE Payments (--
    Username VARCHAR(255) NOT NULL,
    OrderID INT ,
    PaymentMethodStatus VARCHAR(255),
    Amount DECIMAL(10, 2) NOT NULL,
    Date  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (Username) REFERENCES Customers(Username),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);

-- Admins

-- Query to make Admins table

CREATE TABLE Admins (--
    AdminID INT PRIMARY KEY NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL
);

-- Insert

INSERT INTO Admins (AdminID, Password, Name, Email)--
VALUES
    (1, 'admin123', 'John Doe', 'john_admin@example.com'),
    (2, 'admin456', 'Jane Smith', 'jane_admin@example.com'),
    (3, 'admin789', 'Alice Johnson', 'alice_admin@example.com'),
    (4, 'admin987', 'Bob Anderson', 'bob_admin@example.com'),
    (5, 'admin654', 'Michael Brown', 'michael_admin@example.com'),
    (6, 'admin321', 'Lisa Johnson', 'lisa_admin@example.com'),
    (7, 'admin890', 'Tom Anderson', 'tom_admin@example.com'),
    (8, 'admin765', 'Emily Wilson', 'emily_admin@example.com'),
    (9, 'admin234', 'Susan Jones', 'susan_admin@example.com'),
```

```
    (10, 'admin543', 'Peter Clark', 'peter_admin@example.com');
```

TRIGGERS

```
DELIMITER //

CREATE TRIGGER AfterFailedLoginAttempt AFTER INSERT ON FailedLoginAttempts
FOR EACH ROW
BEGIN
    DECLARE attempts INT;

    -- Count the number of failed login attempts for the user within the
last 30 minutes
    SET attempts = (
        SELECT COUNT(*)
        FROM FailedLoginAttempts
        WHERE username = NEW.username
        OR timestamp >= DATE_SUB(NOW(), INTERVAL 2 MINUTE)
    );

    -- Check if the number of failed attempts is greater than or equal to
3
    IF attempts >= 3 THEN
        -- Update the account status to 'Blocked' in the Customers table
        UPDATE Customers
        SET AccountStatus = 'Blocked'
        WHERE Username = NEW.username;
    END IF;
END;
//

DELIMITER ;
```

```
DELIMITER //

CREATE TRIGGER AddToUserCartAfterCustomerInsert
AFTER INSERT ON Customers
FOR EACH ROW
BEGIN
    -- Insert the new customer's username and initial price into user_cart
    INSERT INTO user_cart (username, total_price) VALUES (NEW.Username,
0.00);
END;
//

DELIMITER ;
```

```
DELIMITER //

CREATE TRIGGER add_payment_details
AFTER INSERT ON Customers
FOR EACH ROW
BEGIN
    DECLARE order_id INT;
```

```
    -- Check if the inserted row has a valid username
    IF NEW.Username IS NOT NULL THEN
        -- Insert default payment details for the new customer
        INSERT INTO Payments (Username, OrderID, PaymentMethodStatus,
Amount, Date)
        VALUES (NEW.Username, NULL, NULL, 0.00, CURRENT_DATE());
    END IF;
END;
//

DELIMITER ;
```

DEADLINE 6 (TRANSACTIONS)

CONFLICTING TRANSACTIONS

1

When two different admins try to change the decrease the quantity and other admin try to increase the quantity of the same product,Also Same thing for the Price,If both try to change the price of same product,For this I have used the Locks.

ACTION	TRANSACTION1	TRANSACTION2
Acquire quantity lock	Lock(quantity)	Wait for Lock(quantity)
Acquire price lock	Lock(price)	Wait for Lock(price)
Read quantity from database	Read(quantity)	Wait for Lock(quantity)
Read price from database	Read(price)	Wait for Lock(price)
Calculate new quantity	new_quantity = current_quantity + quantity_change	Wait for Read(quantity)
Calculate new price	new_price = current_price + price_change	Wait for Read(price)
Update quantity in database	Update quantity = new_quantity	Wait for Lock(price)
Update price in database	Update price = new_price	Wait for Read(price)
Commit	Commit	
Release price lock	Unlock(price)	Wait for Unlock(price)
Release quantity lock	Unlock(quantity)	Wait for Unlock(quantity)
Acquire quantity lock		Lock(quantity)
Acquire price lock		Lock(price)
Read quantity from database		Read(quantity)
Read price from database		Read(price)
Calculate new quantity		new_quantity = current_quantity + quantity_change
Calculate new price		new_price = current_price + price_change
Update quantity in database		Update quantity = new_quantity
Update price in database		Update price = new_price
Commit		Commit
Release price lock		Unlock(price)
Release quantity lock		Unlock(quantity)

ACTION	TRANSACTION1	TRANSACTION 2
Acquire quantity lock	Lock(Q)	
Acquire price lock	Lock(P)	
Read quantity from database	Read(Q)	
Read price from database	Read(P)	
Calculate new quantity(Update quantity = new_quantity)	Q: = Q + q	
Calculate new price(new_price = current_price + price_change)	P:=P+p	
abase(new_quantity = current_quan	Write(Q)	
Update price in database	Write(P)	
Commit	Commit	
Release price lock	Unlock(P)	
Release quantity lock	Unlock(Q)	
Acquire Product ID lock		Lock(PID)
Acquire quantity lock		Lock(Q)
Acquire price lock		Lock(P)
Read quantity from database		Read(Q)
Read price from database		Read(P)
Calculate new quantity(Update quantity = new_quantity)		Q: = Q + q
Calculate new price(new_price = current_price + price_change)		P:=P+p
Update quantity in database		Write(Q)
Update price in database		Write(P)
Commit		Commit
Release price lock		Unlock(P)
Release quantity lock		Unlock(Q)

2

Now,If two customers try to order the same product,then it is necessary so that the stock quantity does not go below zero,So if one user tries to order product 1 ,then the lock would be created around this product id.

Transaction 1	Transaction 2
Read customer details	Read customer details
Read product IDs from cart_items	Read product IDs from cart_items
Acquire locks for each product ID	Wait for the lock to be released
Read Product Price	Wait for the lock to be released
Read Product Quantity	Wait for the lock to be released
New Qunatity=Original Quantity-Order Quantity	Wait for the lock to be released
Create and insert order into database	Wait for the lock to be released
Commit	Wait for the lock to be released
Release locks for each product ID	Wait for the lock to be released
	Acquire locks for each product ID
	Read Product Price
	Read Product Quantity
	New Qunatity=Original Quantity-Order Quantity
	Create and insert order into database
	Commit
	Release locks for each product ID

Action	Transaction 1	Transaction 2
Read customer details	Read(Customer)	Read(Customer)
Read product IDs from cart_items	Read(Pid)	Read(Pid)
Acquire locks for each product ID	Lock(Pids)	
Read Product Price	Read(P)	
Read Product Quantity	Read(Q)	
New Qunatity=Original Quantity-Order Quantity	Q:=Q-q	
Write new quantity in database	Write(Q)	
Commit	Commit	
Release locks for each product ID	Unlock(Pids)	
Acquire locks for each product ID		Lock(Pids)
Read Product Price		Read(P)
Read Product Quantity		Read(Q)
New Qunatity=Original Quantity-Order Quantity		Q:=Q-q
Create and insert order into database		Write(Q)
Commit		Commit
Release locks for each product ID		Unlock(Pids)

3

Now,If two admins try to add the same product to the inventory then we can resolve this conflict by taking lock around the product ID.

Action	Transaction 1	Transaction 2
Acquire locks for product ID	Lock(Pid)	
Write Prduct ID	Write(PID)	
Write Name	Write(N)	
Write Description	Write(D)	
Write Supplier Id	Write(S)	
Write Price	Write(P)	
Write Quantity	Write(Q)	
Write Image Link	Write(IL)	
Write Weight	Write(W)	
Commiy	Commit	
Release Lock for product ID	Unlock(Pid)	
		Write(Pid)
		Primary Key constrains

Non Conflicting Transactions

1

Here when two different customers try to add the same item in their cart,it is allowed as they are just adding the items,so non conflicting

Transaction 1	Transaction 2
Read(Pid)	
	Read(Pid)
	Read(P)
Read(P)	
Read(Q)	
	Read(Q)
Write(P)	
	Write(P)
Write(Q)	
Commit	
	Write(Q)
	Commit

Read(P) means, reading the price of respective Pid from invenrory	Read(Q) means,reading the quantity of respective Pid from invenrory	Write(P) means, writing the price of respective Pid to cart	Write(Q) means,writing the quantity of respective Pid to cart

2

Here when two different admins try to modify the product of different product ids.

Transaction 1	Transaction 2
Read(Pid1)	
	Read(Pid2)
	P2:=P2-p2
P1:=P1-p1	
	Q2:=Q2-q2
	Write(P2)
Q1:=Q1-q1	
Write(P1)	
Write(Q1)	
Commit	
	Write(Q2)
	Commit

Read(P1) means,reading the price of Pid1 from inventory	Read(Q2) means,reading the quantity Pid2 from inventory	Write(P1) means,writing the price of Pid1 to inventory	Write(Q2) means,writing the quantity of Pid2 to inventory

3

When two different admins try to add the different products

Transaction 1	Transaction 2
Write(Pid1)	
	Write(Pid2)
Write(N1)	
	Write(N2)
	Write(D2)
Write(D1)	
Write(S1)	
Write(P1)	
	Write(S2)
Write(Q1)	
	Write(P2)
Write(IL1)	
Write(W1)	
	Write(Q2)
	Write(IL2)
Commit	
	Write(W2)
	Commit

Pid1-Product ID1,N2-Name2,D1-Description1,S1-Suppliers ID of product1,P1-Price1,Q2-Quantity2,IL1-Image Link,W2-Weight of product 2

4

Here when two different admins try to do customer or inventory analysis

Transaction 1	Transaction 2
Read(Pids)	
	Read(Pids)
Read(C)	
	Read(C)
	Read(O)
Read(O)	
Read(P)	
	Read(P)

Read(C) means, reading the customer table

Read(Pids) means,reading the Product table

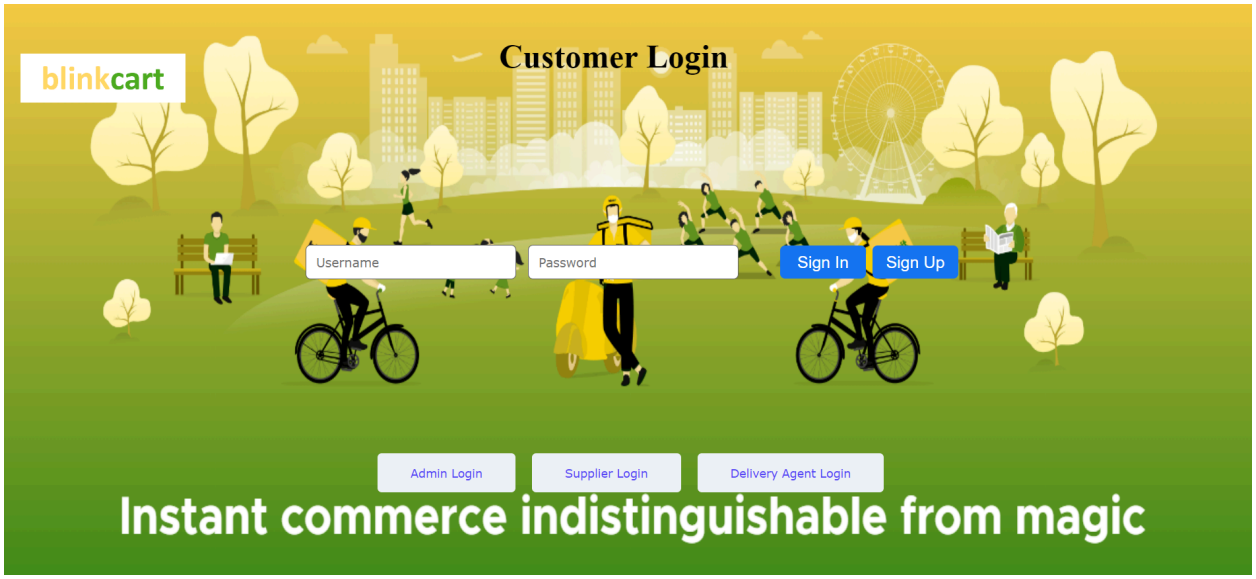
Read(P) means,reading the payments table

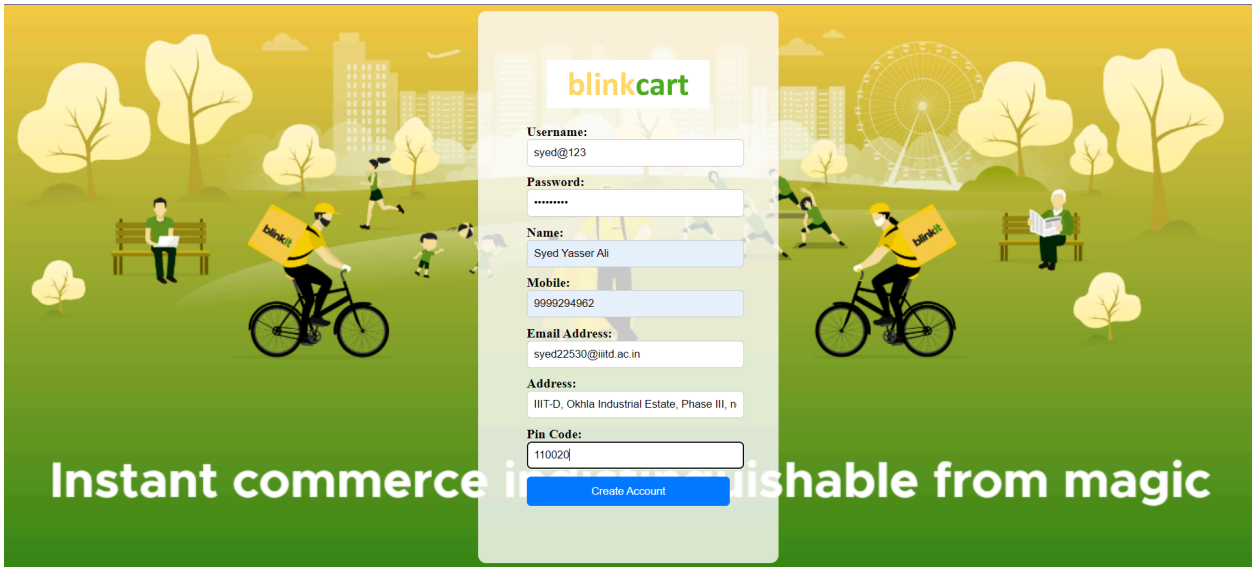
Read(O) means reading the Orders table

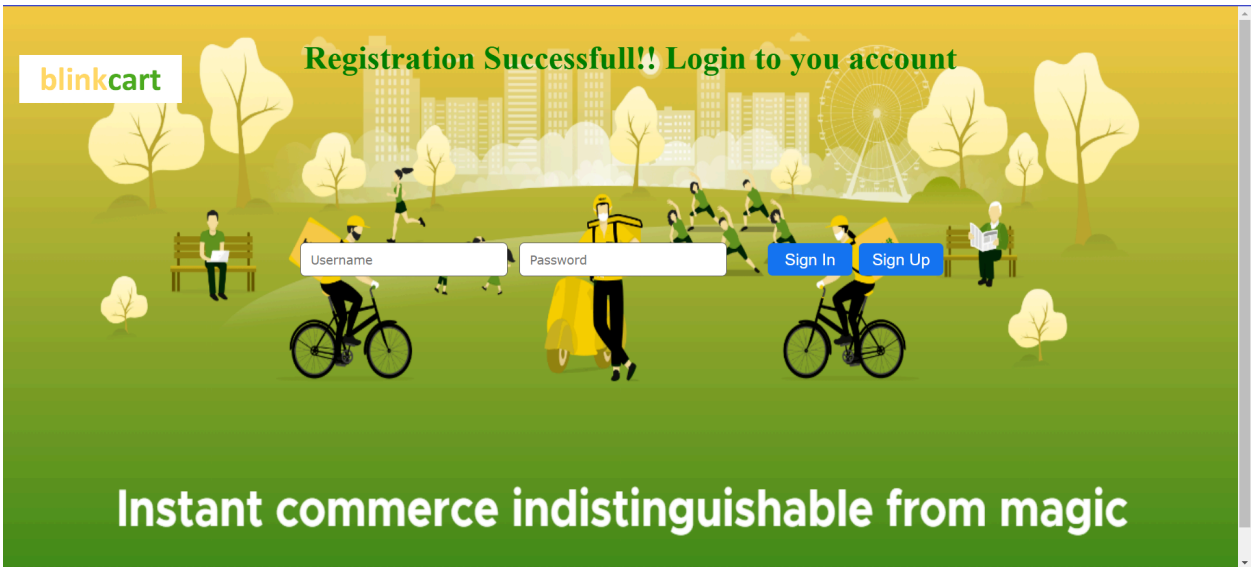
DEADLINE 6
User Guide

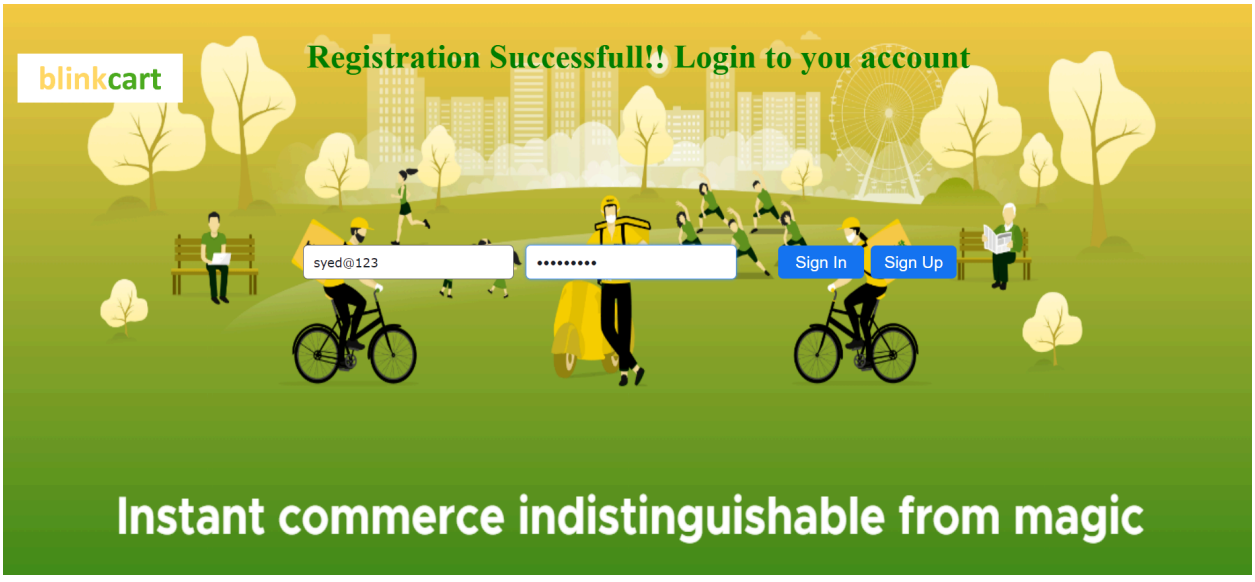
Customer side

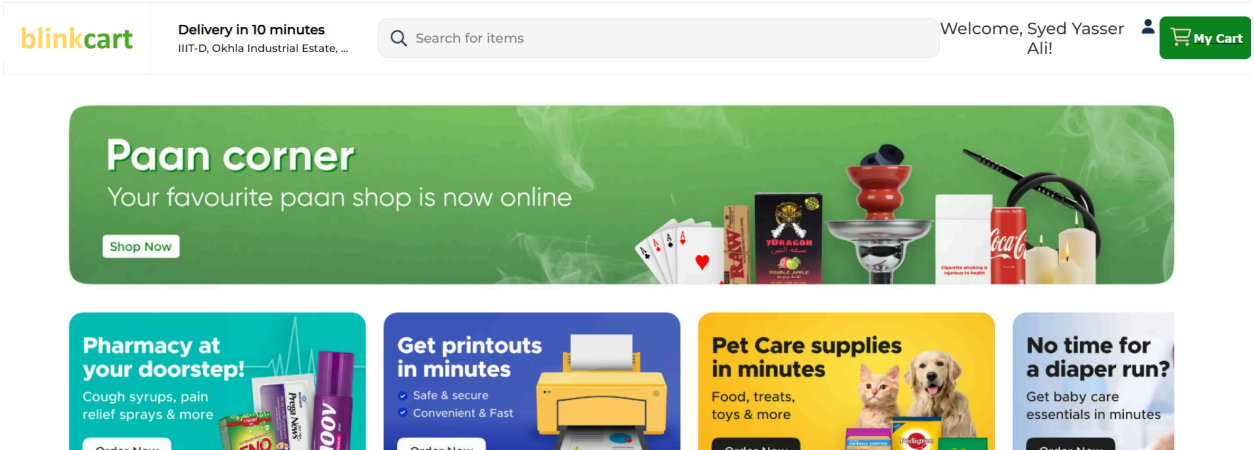
- Customer first needs to Sign Up.
- In the SignUp User needs to provide details.
- After Sign Up,User Can Sign In.
- On entering wrong password three times continuously,the Account gets Blocked.
- After Login User will go to the Main Page of Website,where user can see her/his name.
- Here User can add the items available and their respective quantities to their cart.
  Now User can click the My Cart Button to view Cart items,here user can remove items if needed.
- Now the User can proceed to pay.
- Now The user has to select the payment option
  and place an order.
- After Placing the order,Order will be displayed on
  screen.

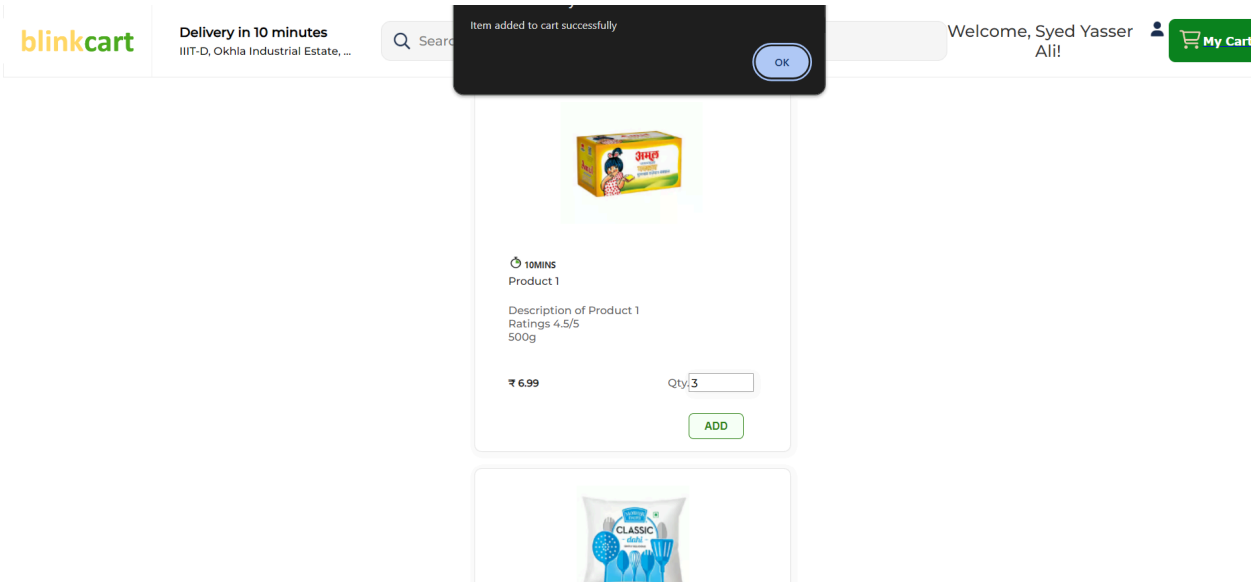

blinkcart
Customer Login
Username
Password
Sign In
Sign Up
Admin Login
Supplier Login
Delivery Agent Login
Instant commerce indistinguishable from magic

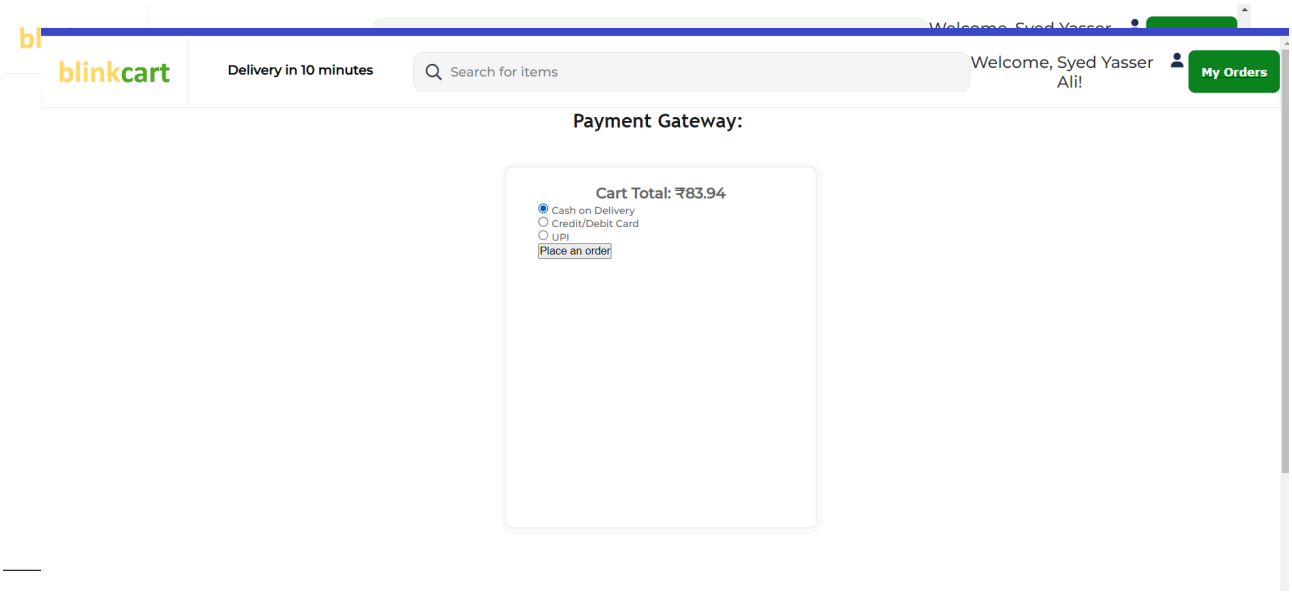

blinkcart
Username:
syed@123
Password:
Name:
Syed Yasser Ali
Mobile:
9999294962
Email Address:
syed22530@iiitd.ac.in
Address:
IIIT-D, Okhla Industrial Estate, Phase III, n
Pin Code:
110020
Create Account
Instant commerce
ishable from magic

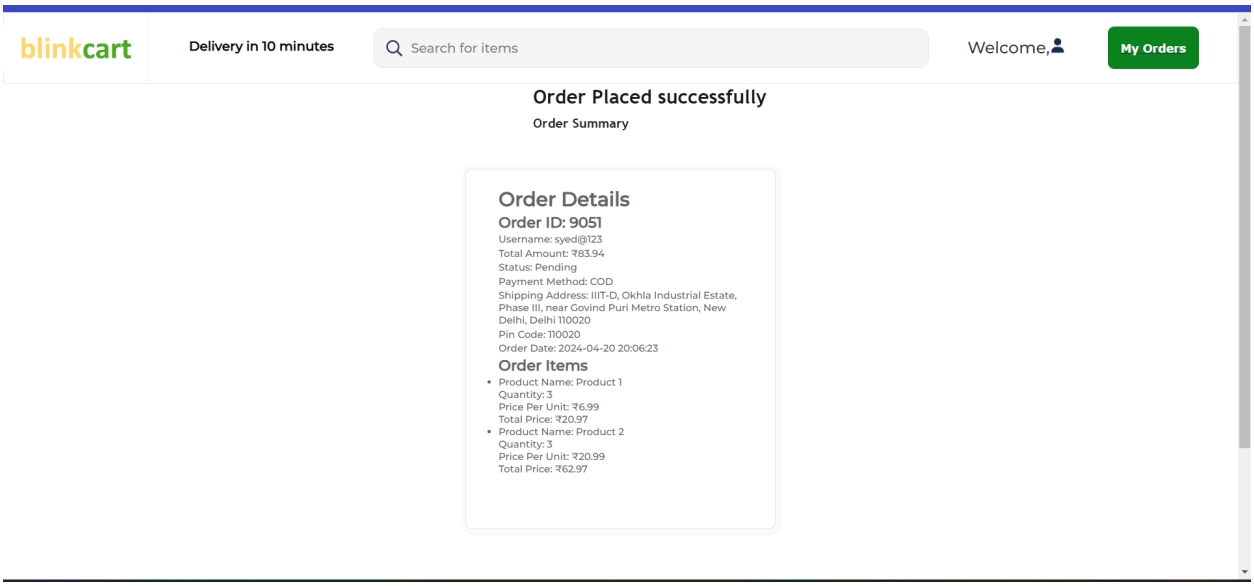

blinkcart
Registration Successfull!! Login to you account
Username
Password
Sign In
Sign Up
Instant commerce indistinguishable from magic


blinkcart
Registration Successfull!! Login to you account
syed@123
Sign In
Sign Up
Instant commerce indistinguishable from magic


blinkcart
Delivery in 10 minutes
IIIT-D, Okhla Industrial Estate, ...
Search for items
Welcome, Syed Yasser Ali!
My Cart
Paan corner
Your favourite paan shop is now online
Shop Now
Pharmacy at your doorstep!
Cough syrups, pain relief sprays & more
Get printouts in minutes
Safe & secure
Convenient & Fast
Pet Care supplies in minutes
Food, treats, toys & more
No time for a diaper run?
Get baby care essentials in minutes


blinkcart
Delivery in 10 minutes
IIIT-D, Okhla Industrial Estate, ...
Item added to cart successfully
OK
Welcome, Syed Yasser Ali!
My Cart
10MINS
Product 1
Description of Product 1
Ratings 4.5/5
500g
₹ 6.99
Qty 3
ADD

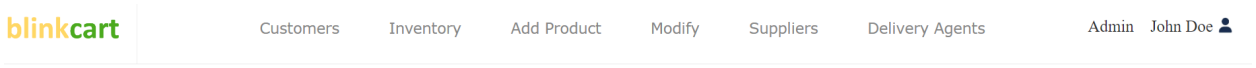

blinkcart
Delivery in 10 minutes
Search for items
Welcome, Syed Yasser Ali!
My Orders
Payment Gateway:
Cart Total: ₹83.94
Cash on Delivery
Credit/Debit Card
UPI
Place an order

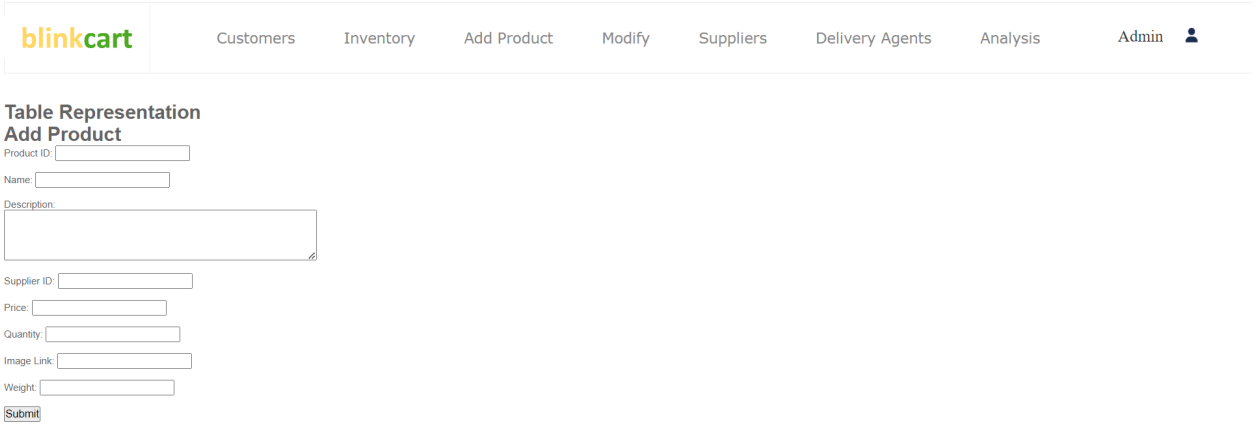


Admin Side

- Here Admin can login using her/his credentials.
- Admin can View inventory,Add Product,Modify Products etc.


blinkcart
Admin Login
1
Sign In
Instant commerce indistinguishable from magic


blinkcart
Customers
Inventory
Add Product
Modify
Suppliers
Delivery Agents
Admin
John Doe


blinkcart
Customers
Inventory
Add Product
Modify
Suppliers
Delivery Agents
Analysis
Admin
Table Representation
Add Product
Product ID:
Name:
Description:
Supplier ID:
Price:
Quantity:
Image Link:
Weight:
Submit

blinkcart Customers Inventory Add Product Modify Suppliers Delivery Agents Analysis Admin

Inventory_analysis Representation

ProductID	Name	Description	SupplierID	Price	Ratings	Quantity	ImageLink	Weight
1	Product 1	Description of Product 1	1	6.99	4.5	492.00	product1.png	500g
2	Product 2	Description of Product 2	2	20.99	4.8	995.00	product2.png	1 ltr
3	Product 3	Description of Product 3	3	15.99	4.3	247.00	product3.png	250g
4	Product 4	Description of Product 4	5	12.99	4.2	750.00	product4.png	750ml
5	Product 5	Description of Product 5	5	18.99	4.6	200.00	product5.png	200g
6	Product 6	Description of Product 6	1	25.99	4.7	1000.00	product6.png	1 kg
7	Product 7	Description of Product 7	7	14.99	4.4	2000.00	product7.png	2 ltr
8	Product 8	Description of Product 8	2	22.99	4.9	500.00	product8.png	500ml
11	p11	this is p11	1	10.00	None	8.00	na	500g
12	p12	12	1	10.00	None	11.00	na	500g

blinkcart Customers Inventory Add Product Modify Suppliers Delivery Agents Analysis Admin

Table Representation

ModifyProduct

Product ID:

Price Change:

Quantity Change:

Submit