ONLINE FOOD DELIVERY SYSTEM

DBMS Mini Project

(505, 515, 522, 523)

PROJECT SRS :

Online Food Delivery System(OFDS) implementation that creates the necessary databases and tables, inserts records, and defines triggers, procedures, and methods for managing customers, menu items, orders, and payments.

Functional Requirements :

1. Customer Management:

- Store customer details (ID, name, mobile, DOB, city).

- Classify customers as Prime or Regular with separate attributes (e.g., membership dates, points).

2. Menu Management:

- Define menu items with attributes such as ID, name, price, type (Veg, Non-veg) and availability

3. Order and Payment Handling:

- Create orders and associate each order with items and quantities.

- Calculate total prices using triggers for items ordered.

- Manage payment records for each order with methods (Cash, UPI, Card) and status (Paid, Not paid).

4. JDBC Integration:

- Java classes connect to the MySQL database for creating and managing tables.

- Methods are implemented for inserting, updating, and deleting customer, order, and payment records.

5. Stored Procedures and Triggers:

- A trigger calculates the total price (`o_price`) of each order based on quantity and item price.

- A stored procedure identifies customers with unpaid orders.

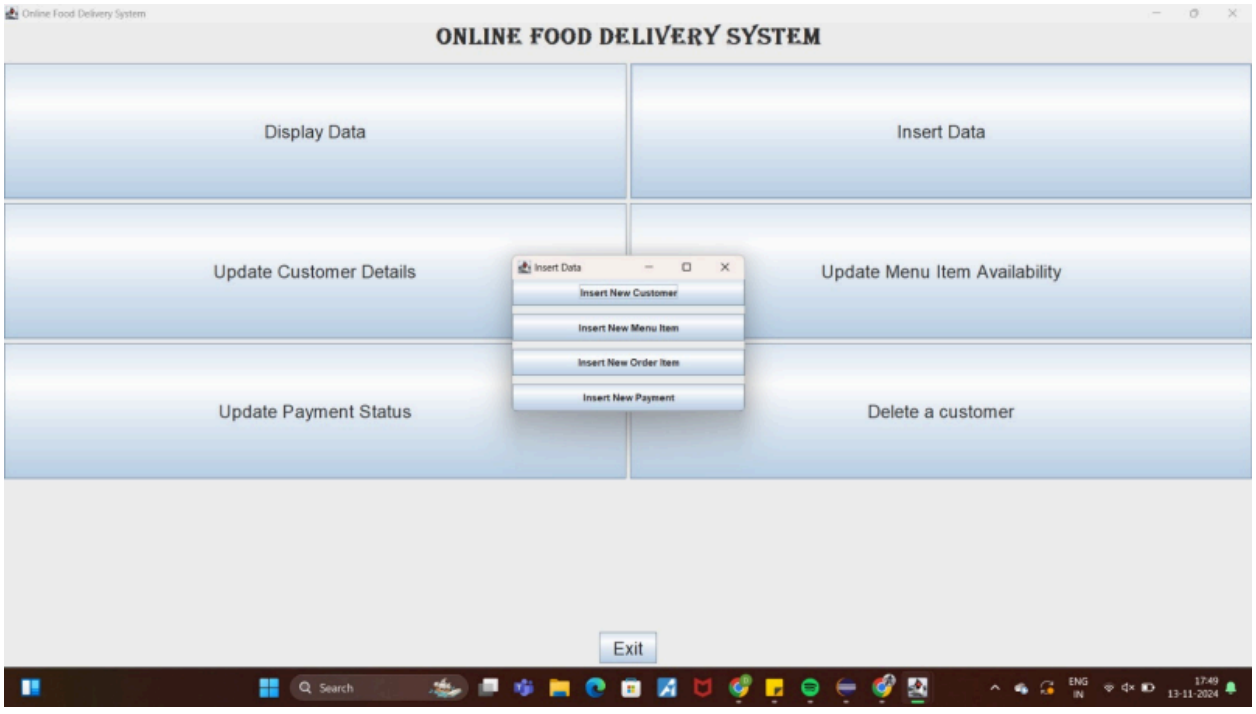
Database Setup: SQL scripts to create `Customer`, `PrimeCustomer`, `RegularCustomer`, `Menu`, `OrderItems`, and `Payment` tables.

Java Class: Manages connections, inserts, updates, and retrieves data via JDBC.

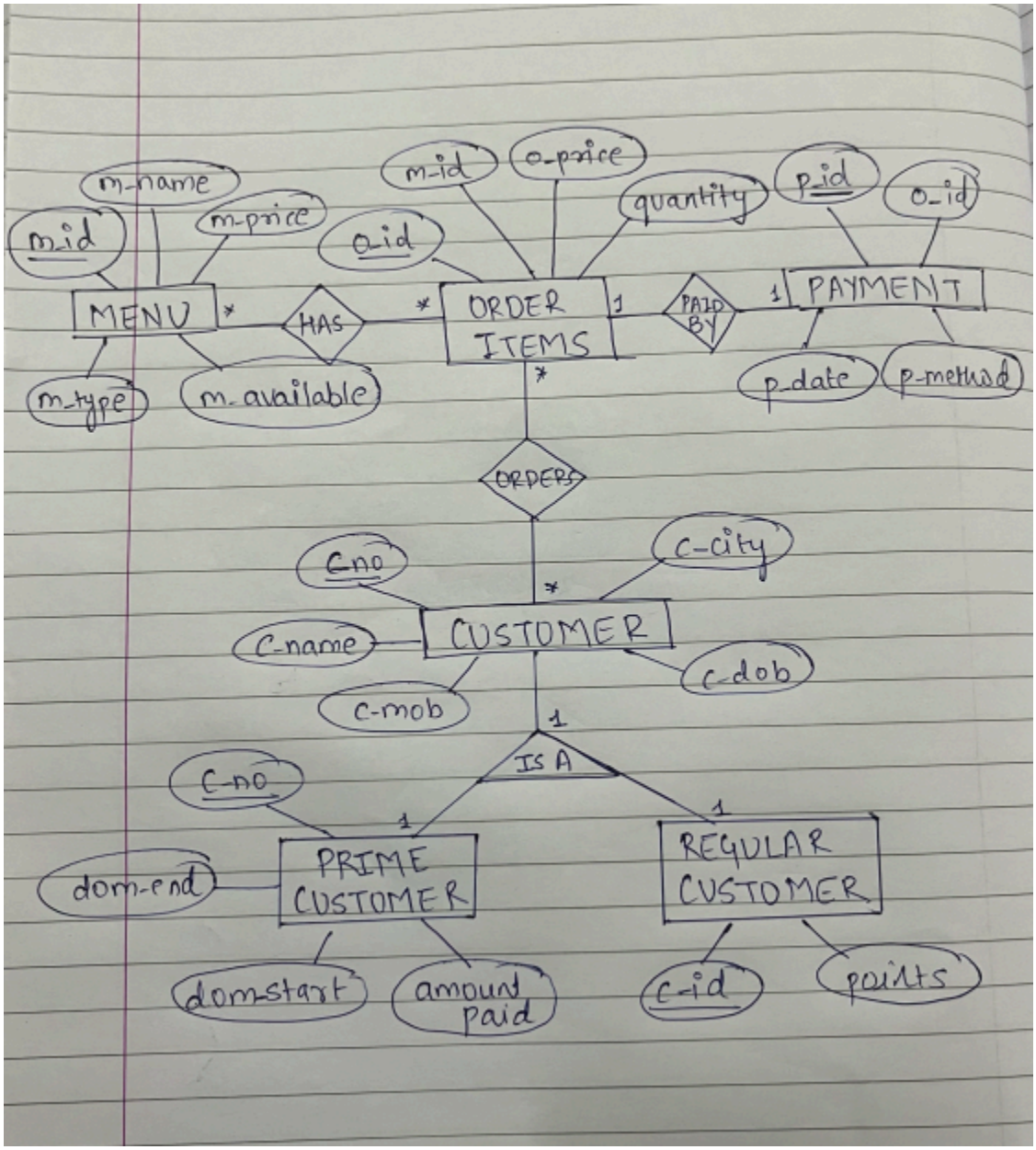
Trigger `calc_price`: Automatically calculates `o_price` for each order based on `quantity`.

Procedure `customer_notpaid`: Lists customers with unpaid orders using a LEFT JOIN and filtering by `p_status`.

This setup provides a functional system for managing customer records, menu items, order processing, and payments in a food delivery system. The code also supports CRUD operations, making it extensible for additional requirements or integration.



ER DIAGRAM :



Entities and Attributes :

Customer

Attributes: c_no (PK), c_name, c_mob, c_dob, c_city

Menu

Attributes: m_id (PK), m_name, m_price, m_type, m_available

OrderItems

Attributes: o_id (PK), m_id (FK to Menu), quantity, o_price, c_no (FK to Customer)

Payment

Attributes: p_id (PK), o_id (FK to OrderItems), p_date, p_method, p_status

RegularCustomer

Attributes: c_no (PK, FK to Customer), points

PrimeCustomer

Attributes: c_no (PK, FK to Customer), dom_start, dom_end, amount_paid

NORMALISATION :

1NF (First Normal Form): All tables should have atomic values (no repeating groups or arrays).

2NF (Second Normal Form): The table should be in 1NF, and every non- prime attribute must be fully functionally dependent on the primary key.

3NF (Third Normal Form): The table should be in 2NF, and there should be no transitive dependency (i.e., non-prime attributes should not depend on other non-prime attributes).

1. Customer Table

The `Customer` table already appears to be Normalized.

```
CREATE TABLE Customer (
    c_no INT PRIMARY KEY,
    c_name
    VARCHAR(50),
    c_mob
    VARCHAR(15),
    c_dob DATE,
    c_city VARCHAR(50)
);
```

2. PrimeCustomer and RegularCustomer Tables

The `PrimeCustomer` and `RegularCustomer` tables are also Normalized.

```
 CREATE TABLE PrimeCustomer (
    c_no INT PRIMARY KEY,
    dom_start DATE,
    dom_end DATE,
    amount_paid
    DOUBLE,
    FOREIGN KEY (c_no) REFERENCES Customer(c_no)
);
    CREATE TABLE
        RegularCustomer ( c_no
        INT PRIMARY KEY,
        points INT,
        FOREIGN KEY (c_no) REFERENCES Customer(c_no)
);
```

3. Menu Table

The `Menu` table is already normalized.

```
CREATE TABLE Menu (
    m_id INT PRIMARY KEY,
    m_name
    VARCHAR(50),
    m_price DOUBLE,
    m_type ENUM('Veg',
    'Non-Veg'),
     m_available BOOLEAN
);
```

4. OrderItems Table

The `OrderItems` table is also Normalized.

```
CREATE TABLE OrderItems (
    o_id INT PRIMARY KEY,
    m_id INT,
    quantity
    INT, o_price
    DOUBLE,
    FOREIGN KEY (m_id) REFERENCES Menu(m_id)
);
```

5. Payment Table

The `Payment` table is in 3NF:

```
    CREATE TABLE
        Payment
     ( p_id INT PRIMARY
        KEY,
        O_id
    INT
       p_date
    DATE,
       p_method ENUM('Cash', 'UPI', 'Card')
       FOREIGN KEY (o_id) REFERENCES OrderItems(o_id)
);
```

Each table satisfies 1NF, 2NF, and 3NF:

- 1 NF : All tables have atomic values.
- 2 NF : Non-key attributes fully depend on the primary key.
- 3 NF : There are no transitive dependencies.

Hence, all above tables are normalized upto 3 NF.