

CS 353 Database Systems

Design Report



Group - 20

Group Members

Rüzgar Ayan

Kamil Kaan Erkan

Cankat Anday Kadim

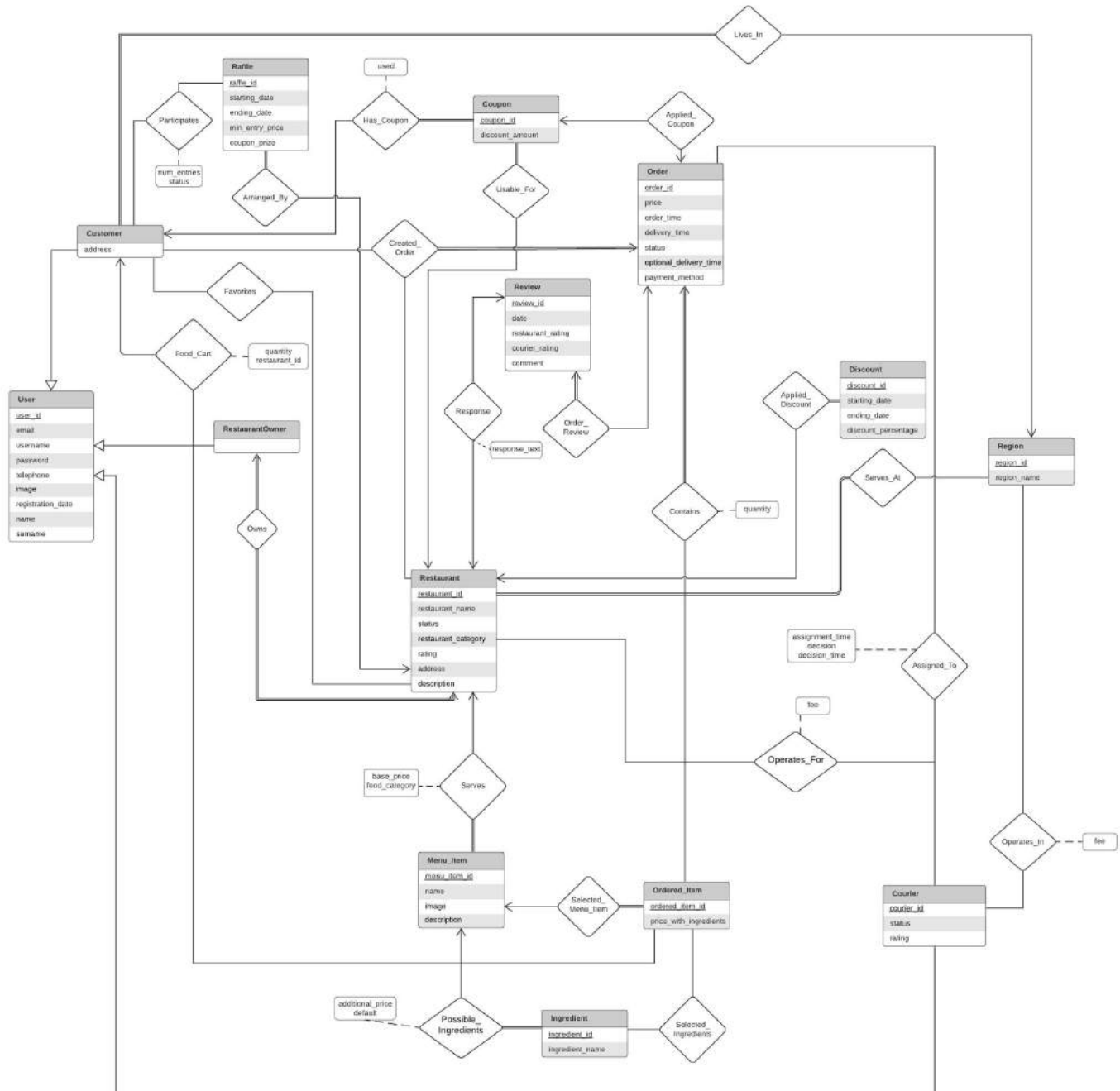
Ege Türker

Table of Contents

1. Revised E/R Diagram	4
2. Relational Schemas	5
2.1. User Table	5
2.2. Customer Table	6
2.3. Restaurant Table	7
2.4. Courier Table	8
2.5. Region Table	9
2.6. Order Table	10
2.7. Ordered_Item Table	11
2.8. Menu_Item Table	12
2.9. Review Table	13
2.10. Ingredient Table	14
2.11. Response Table	15
2.12. Discount Table	16
2.13. Favorites Table	17
2.14. Serves_At Table	18
2.15. Operates_In Table	19
2.16. Assigned_To Table	20
2.17. Contains Table	21
2.18. Selected_Ingredients Table	22
2.19. Possible_Ingredients Table	23
2.20. Operates_For Table	24
2.21. Raffle Table	25
2.22. Coupon Table	26
2.23. Participates Table	27
2.24. Applied_Coupon Table	28
2.25. Food_Cart Table	29
3. User Interface Design and Corresponding SQL Statements	30
3.1. Login and Signup Functionalities	30
3.1.1. Sign-Up Page	30
3.1.2. Login Page	31
3.2. Additional requirement - Raffles	32
3.2.1. Restaurant Starting a Raffle	32
3.2.2. Customer Seeing Their Coupons In Their Profile	34

3.2.3. Current Raffle Details of a Restaurant	35
3.2.4. Customer Applying a Coupon When Finalizing an Order	36
3.2.5. Customer Gaining New Entry to a Raffle	37
3.3. Customer User Interface	38
3.3.1. Restaurant List Page	38
3.3.2. Restaurant Menu Page	40
3.3.3. Adding Item To Cart	41
3.3.4. Order History Page	42
3.3.5. Profile Page	43
3.3.6. Order Details and Review Page	44
3.3.7. Customer Finalize Order Page	45
3.4. Restaurant User Interface	46
3.4.1. Restaurant Administrative Page	46
3.4.2. Restaurant Modify Menu Page	47

1. Revised E/R Diagram



2. Relational Schemas

2.1. User Table

Relational Model

User(user_id, name, surname, email, username, password, telephone, registration_date, image)

Functional Dependencies

user_id → name, surname, email, username, password, telephone, registration_date, image

username → user_id, name, surname, email, password, telephone, registration_date, image

email → user_id, name, surname, username, password, telephone, registration_date, image

telephone → user_id, name, surname, email, username, password, registration_date, image

Candidate Keys

{(user_id), (username), (email), (telephone)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE User(  
  user_id INT AUTO_INCREMENT,  
  name VARCHAR(32) NOT NULL,  
  surname VARCHAR(32),  
  email VARCHAR(64) NOT NULL UNIQUE,  
  username VARCHAR(32) NOT NULL UNIQUE,
```

```

password VARCHAR(32) NOT NULL,
telephone CHAR(10) NOT NULL UNIQUE,
registration_date DATE NOT NULL,
image VARCHAR(64), --stored as link

PRIMARY KEY(user_id),
);

```

2.2. Customer Table

Relational Model

Customer(customer_id, address)

FK: customer_id references User

Functional Dependencies

customer_id → address

Candidate Keys

{{customer_id}}

Normal Form

BCNF

Table Definition

```

CREATE TABLE Customer(
    customer_id INT,
    address VARCHAR(128),

    PRIMARY KEY(customer_id),
    FOREIGN KEY(customer_id) REFERENCES User
);

```

2.3. Restaurant Table

Relational Model

Restaurant(restaurant_id, owner_id, restaurant_name, rating, address, description, restaurant_category, status)

FK: owner_id references User

Functional Dependencies

restaurant_id \rightarrow owner_id, restaurant_name, rating, address, description, restaurant_category, status

owner_id \rightarrow restaurant_id, restaurant_name, rating, address, description, restaurant_category, status

Candidate Keys

{{restaurant_id}, {owner_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Restaurant(  
    restaurant_id INT AUTO_INCREMENT,  
    owner_id INT NOT NULL,  
    restaurant_name VARCHAR(32) NOT NULL,  
    rating DOUBLE,  
    address VARCHAR(128),  
    description VARCHAR(128),  
    restaurant_category VARCHAR(32),  
    status VARCHAR(16),  
  
    PRIMARY KEY(restaurant_id),  
    FOREIGN KEY(owner_id) REFERENCES User,  
    CONSTRAINT `valid_rating` CHECK (rating BETWEEN 0 AND 5)  
);
```

2.4. Courier Table

Relational Model

Courier(courier_id, status, rating)
FK: courier_id references User

Functional Dependencies

courier_id \rightarrow status, rating

Candidate Keys

{{courier_id }}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Courier(  
    courier_id INT,  
    status VARCHAR(16),  
    rating DOUBLE,  
  
    PRIMARY KEY(customer_id),  
    FOREIGN KEY(courier_id) REFERENCES User,  
    CONSTRAINT 'valid_status'  
        CHECK (status in ('available', 'not_available'))  
);
```


2.5. Region Table

Relational Model

Region(region_id, region_name)

Functional Dependencies

region_id \rightarrow region_name

Candidate Keys

{(region_id)}

Note: Assuming that there could be two regions with the same name in different locations.

Normal Form

BCNF

Table Definition

```
CREATE TABLE Region(  
    region_id INT AUTO_INCREMENT,  
    region_name VARCHAR(32) NOT NULL,  
  
    PRIMARY KEY(region_id)  
);
```

2.6. Order Table

Relational Model

Order(order_id, restaurant_id, customer_id, price, order_time, delivery_time, status, optional_delivery_time)

FK: restaurant_id references Restaurant

FK: customer_id references Customer

Functional Dependencies

order_id → restaurant_id, customer_id, price, order_time, delivery_time, status, optional_delivery_time

restaurant_id, customer_id, order_time → order_id, price, delivery_time, status, optional_delivery_time

Candidate Keys

{{(order_id), (restaurant_id, customer_id, order_datetime)}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Order(  
    order_id INT AUTO_INCREMENT,  
    restaurant_id INT,  
    customer_id INT,  
    price DOUBLE,  
    order_time DATETIME DEFAULT CURRENT_TIMESTAMP,  
    delivery_time DATETIME,  
    status VARCHAR(32),  
    optional_delivery_time DATETIME,  
  
    PRIMARY KEY(order_id),  
    FOREIGN KEY(restaurant_id) REFERENCES Restaurant  
    FOREIGN KEY(customer_id) REFERENCES Customer  
);
```

2.7. Ordered_Item Table

Relational Model

Ordered_Item(ordered_item_id, price_with_ingredients, menu_item_id)
FK: menu_item_id references Menu_Item

Functional Dependencies

ordered_item_id \rightarrow price_with_ingredients, menu_item_id

Candidate Keys

{{ordered_item_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Ordered_Item(  
    ordered_item_id INT AUTO_INCREMENT,  
    price_with_ingredients DOUBLE,  
    menu_item_id INT,  
  
    PRIMARY KEY (order_id) ,  
    FOREIGN KEY (menu_item_id) REFERENCES Menu_Item  
);
```

2.8. Menu_Item Table

Relational Model

Menu_Item(menu_item_id, name, image, description, base_price, food_category, restaurant_id)

FK: restaurant_id references Restaurant

Functional Dependencies

menu_item_id → name, image, description, base_price, food_category, restaurant_id

Candidate Keys

{(menu_item_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Menu_Item(  
    menu_item_id INT AUTO_INCREMENT,  
    name VARCHAR(16),  
    image VARCHAR(64), --stored as link  
    description VARCHAR(128),  
    base_price DOUBLE NOT NULL,  
    food_category VARCHAR(16) NOT NULL,  
  
    PRIMARY KEY(menu_item_id),  
    FOREIGN KEY (restaurant_id) REFERENCES Restaurant  
);
```

2.9. Review Table

Relational Model

Review(review_id, date, restaurant_rating, courier_rating, comment, order_id)
FK: order_id references Order

Functional Dependencies

review_id → date, restaurant_rating, courier_rating, comment, order_id
order_id → review_id, restaurant_rating, courier_rating, comment

Candidate Keys

{(review_id), (order_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Review(  
    review_id INT AUTO_INCREMENT,  
    date DATE NOT NULL,  
    restaurant_rating INT NOT NULL,  
    courier_rating INT NOT NULL,  
    comment VARCHAR(128) ,  
    order_id INT,  
  
    PRIMARY KEY(review_id) ,  
    FOREIGN KEY(order_id) REFERENCES Order  
);
```

2.10. Ingredient Table

Relational Model

Ingredient(ingredient_id, ingredient_name, menu_item_id, additional_price, default)

FK: menu_item_id references Menu_Item

Functional Dependencies

ingredient_id → ingredient_name, menu_item_id, additional_price, default

Candidate Keys

{{ingredient_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Ingredient(  
    ingredient_id INT AUTO_INCREMENT,  
    ingredient_name VARCHAR(128) NOT NULL,  
    menu_item_id INT NOT NULL,  
    additional_price DOUBLE NOT NULL,  
    default BIT(1) NOT NULL,  
  
    PRIMARY KEY(ingredient_id),  
    FOREIGN KEY (menu_item_id) REFERENCES Menu_Item  
);
```

2.11. Response Table

Relational Model

Response(restaurant_id, review_id, response_text)

FK: restaurant_id references Restaurant

FK: review_id references Review

Functional Dependencies

restaurant_id, review_id \rightarrow response_text

Candidate Keys

{{restaurant_id, review_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Response (  
    restaurant_id INT NOT NULL,  
    review_id INT NOT NULL,  
    response_text VARCHAR(128) ,  
  
    PRIMARY KEY (restaurant_id, review_id) ,  
    FOREIGN KEY (restaurant_id) REFERENCES Restaurant ,  
    FOREIGN KEY (review_id) REFERENCES Review  
);
```

2.12. Discount Table

Relational Model

Discount(discount_id, starting_date, ending_date, discount_percentage, restaurant_id)

FK: restaurant_id references Restaurant

Functional Dependencies

discount_id → starting_date, ending_date, discount_percentage, restaurant_id

Candidate Keys

{{discount_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Discount (  
    discount_id INT NOT NULL AUTO_INCREMENT,  
    starting_date DATE NOT NULL,  
    ending_date DATE NOT NULL,  
    discount_percentage DOUBLE NOT NULL,  
    restaurant_id INT NOT NULL,  
  
    PRIMARY KEY (discount_id)  
);
```


2.13. Favorites Table

Relational Model

Favorites(customer_id, restaurant_id)
FK: customer_id references Customer
FK: restaurant_id references Restaurant

Functional Dependencies

None

Candidate Keys

{{customer_id, restaurant_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Favorites(  
    customer_id INT NOT NULL,  
    restaurant_id INT NOT NULL,  
  
    PRIMARY KEY(customer_id, restaurant_id),  
    FOREIGN KEY(customer_id) REFERENCES Customer,  
    FOREIGN KEY(restaurant_id) REFERENCES Restaurant  
);
```

2.14. Serves_At Table

Relational Model

Serves_At(restaurant_id, region_id)

FK: restaurant_id references Restaurant

FK: region_id references Region

Functional Dependencies

None

Candidate Keys

{{restaurant_id, region_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Serves_At (  
    restaurant_id INT NOT NULL,  
    region_id INT NOT NULL,  
  
    PRIMARY KEY (restaurant_id, region_id),  
    FOREIGN KEY (restaurant_id) REFERENCES Restaurant,  
    FOREIGN KEY (region_id) REFERENCES Region  
);
```

2.15. Operates_In Table

Relational Model

Operates_In(courier_id, region_id, fee)

FK: courier_id references Courier

FK: region_id references Region

Functional Dependencies

courier_id, region_id \rightarrow fee

Candidate Keys

{{courier_id, region_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Operates_In(  
    courier_id INT NOT NULL,  
    region_id INT NOT NULL,  
    fee DOUBLE NOT NULL,  
    PRIMARY KEY (courier_id, region_id),  
    FOREIGN KEY (courier_id) REFERENCES Courier,  
    FOREIGN KEY (region_id) REFERENCES Region  
);
```

2.16. Assigned_To Table

Relational Model

Assigned_To (order_id, courier_id, assignment_time, decision, decision_time)

FK: order_id references Order

FK: courier_id references Courier

Functional Dependencies

order_id, courier_id → assignment_time, decision, decision_time

Candidate Keys

{(order_id, courier_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Assigned_To(  
    order_id INT NOT NULL,  
    courier_id INT NOT NULL,  
    assignment_time TIME NOT NULL,  
    decision VARCHAR(20) NOT NULL,  
    decision_time TIME NOT NULL,  
    PRIMARY KEY (order_id, courier_id),  
    FOREIGN KEY (order_id) REFERENCES Order,  
    FOREIGN KEY (courier_id) REFERENCES Courier  
);
```

2.17. Contains Table

Relational Model

Contains (order_id, ordered_item_id, quantity)

FK: order_id references Order

FK: ordered_item_id references Order_Item

Functional Dependencies

order_id, ordered_item_id \rightarrow quantity

Candidate Keys

{{order_id, ordered_item_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Contains(  
    order_id INT NOT NULL,  
    ordered_item_id INT NOT NULL,  
    quantity INT NOT NULL,  
    PRIMARY KEY (order_id, ordered_item_id),  
    FOREIGN KEY (order_id) REFERENCES Order,  
    FOREIGN KEY (ordered_item_id) REFERENCES Ordered_Item  
);
```

2.18. Selected_Ingredients Table

Relational Model

Selected_Ingredients (order_item_id, ingredient_id)

FK: order_id references Order

FK: ingredient_id references Ingredient

Functional Dependencies

None

Candidate Keys

{{order_item_id, ingredient_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Selected_Ingredients(  
    ordered_item_id INT NOT NULL,  
    ingredient_id INT NOT NULL,  
    PRIMARY KEY (ordered_item_id, ingredient_id),  
    FOREIGN KEY (ordered_item_id) REFERENCES Order_Item,  
    FOREIGN KEY (ingredient_id) REFERENCES Ingredient  
);
```

2.19. Possible_Ingredients Table

Relational Model

Possible_Ingredients (ingredient_id, menu_item_id, additional_price, default)

FK: ingredient_id references Ingredient

FK: menu_item_id references Menu_Item

Functional Dependencies

ingredient_id, menu_item_id → additional_price, default

Candidate Keys

{{ingredient_id, menu_item_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Possible_Ingredients(  
    ingredient_id INT NOT NULL,  
    menu_item_id INT NOT NULL,  
    additional_price DOUBLE NOT NULL,  
    default BIT(1) NOT NULL,  
  
    PRIMARY KEY (ingredient_id, menu_item_id),  
    FOREIGN KEY (ingredient_id) REFERENCES Ingredient,  
    FOREIGN KEY (menu_item_id) REFERENCES Menu_Item  
);
```

2.20. Operates_For Table

Relational Model

Operates_For (restaurant_id, courier_id, fee)

FK: restaurant_id references Restaurant

FK: courier_id references Courier

Functional Dependencies

restaurant_id, courier_id \rightarrow fee

Candidate Keys

{{restaurant_id, courier_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Operates_For(  
    restaurant_id INT NOT NULL,  
    courier_id INT NOT NULL,  
    fee DOUBLE NOT NULL,  
  
    PRIMARY KEY(restaurant_id, courier_id),  
    FOREIGN KEY(restaurant_id) REFERENCES Restaurant,  
    FOREIGN KEY(courier_id) REFERENCES Courier  
);
```


2.21. Raffle Table

Relational Model

Raffle(raffle_id, starting_date, ending_date, min_entry_price, coupon_prize, restaurant_id)

Functional Dependencies

raffle_id → starting_date, ending_date, min_entry_price, coupon_prize, restaurant_id

restaurant_id, starting_date → raffle_id, ending_date, min_entry_price, coupon_prize

restaurant_id, ending_date → raffle_id, starting_date, min_entry_price, coupon_prize

Note: Assuming that a restaurant arranges only one raffle at a time.

Candidate Keys

{(raffle_id), (restaurant_id, starting_date), (restaurant_id, ending_date)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Raffle(  
    raffle_id INT NOT NULL AUTO_INCREMENT,  
    starting_date DATETIME NOT NULL,  
    ending_date DATETIME NOT NULL,  
    min_entry_price DOUBLE,  
    coupon_prize DOUBLE NOT NULL,  
    restaurant_id INT NOT NULL,  
  
    PRIMARY KEY(raffle_id),  
    FOREIGN KEY(restaurant_id) REFERENCES Restaurant  
);
```

2.22. Coupon Table

Relational Model

Coupon (coupon_id, discount_amount, customer_id, used, restaurant_id)

FK: customer_id references Customer

FK: restaurant_id references Restaurant

Functional Dependencies

coupon_id → discount_amount, customer_id, used, restaurant_id

Candidate Keys

{(coupon_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Coupon (  
    coupon_id INT NOT NULL AUTO_INCREMENT,  
    discount_amount DOUBLE NOT NULL,  
    customer_id INT NOT NULL,  
    used BIT(1),  
    restaurant_id INT NOT NULL,  
  
    PRIMARY KEY (coupon_id),  
    FOREIGN KEY (customer_id) REFERENCES Customer  
    FOREIGN KEY (restaurant_id) REFERENCES Restaurant  
);
```

2.23. Participates Table

Relational Model

Participates (user_id, raffle_id, num_entries, status)

FK: user_id references Customer

FK: raffle_id references Raffle

Functional Dependencies

user_id, raffle_id → num_entries, status

Candidate Keys

{(user_id, raffle_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Participates(  
    user_id INT NOT NULL,  
    raffle_id INT NOT NULL,  
    num_entries INT NOT NULL,  
    status VARCHAR(16) ,  
  
    PRIMARY KEY(user_id, raffle_id),  
    FOREIGN KEY(user_id) REFERENCES Customer,  
    FOREIGN KEY(raffle_id) REFERENCES Raffle
```

2.24. Applied_Coupon Table

Relational Model

Applied_Coupon(order_id, coupon_id)

FK: order_id references Order

FK: coupon_id references Coupon

Functional Dependencies

order_id \rightarrow coupon_id

coupon_id \rightarrow order_id

Candidate Keys

{{(order_id), (coupon_id)}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Applied_Coupon(  
    order_id INT NOT NULL,  
    coupon_id INT NOT NULL,  
  
    PRIMARY KEY (order_id),  
    FOREIGN KEY (order_id) REFERENCES Order,  
    FOREIGN KEY (coupon_id) REFERENCES Coupon  
);
```

2.25. Food_Cart Table

Relational Model

Food_Cart (user_id, ordered_item_id, restaurant_id)

FK: user_id references Customer

FK: ordered_item_id references Ordered_Item

FK: restaurant_id references Restaurant

Functional Dependencies

None

Candidate Keys

{{user_id, ordered_item_id, restaurant_id}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Food_Cart(  
    user_id INT NOT NULL,  
    ordered_item_id INT NOT NULL,  
    restaurant_id INT NOT NULL,  
  
    PRIMARY KEY(user_id, ordered_item_id, restaurant_id),  
    FOREIGN KEY(user_id) REFERENCES Customer,  
    FOREIGN KEY(ordered_item_id) REFERENCES Ordered_Item  
    FOREIGN KEY(restaurant_id) REFERENCES Restaurant  
);
```

3. User Interface Design and Corresponding SQL Statements

3.1. Login and Signup Functionalities

3.1.1. Sign-Up Page

lotteatry

REGISTER

Name
John

Surname
Doe

Username
john.eats

E-mail
john.doe@mail.com

Password

Telephone
01279 231326

☒ Customer ☐ Restaurant ☐ Courier

Already have an account?
[Sign in.](#)

Sign Up

Customer Sign-up

```
INSERT INTO User (name, surname, email, username, password, telephone,
registration_date)
VALUES (@name, @surname, @email, @username, @password, @telephone,
CURDATE())
```

```
--@user_id is the auto-incremented id of newly created User above
INSERT INTO Customer VALUES (@user_id)
```

Restaurant Sign-up

```
INSERT INTO User (name, surname, email, username, password, telephone,
registration_date)
VALUES (@name, @surname, @email, @username, @password, @telephone,
CURDATE())
```

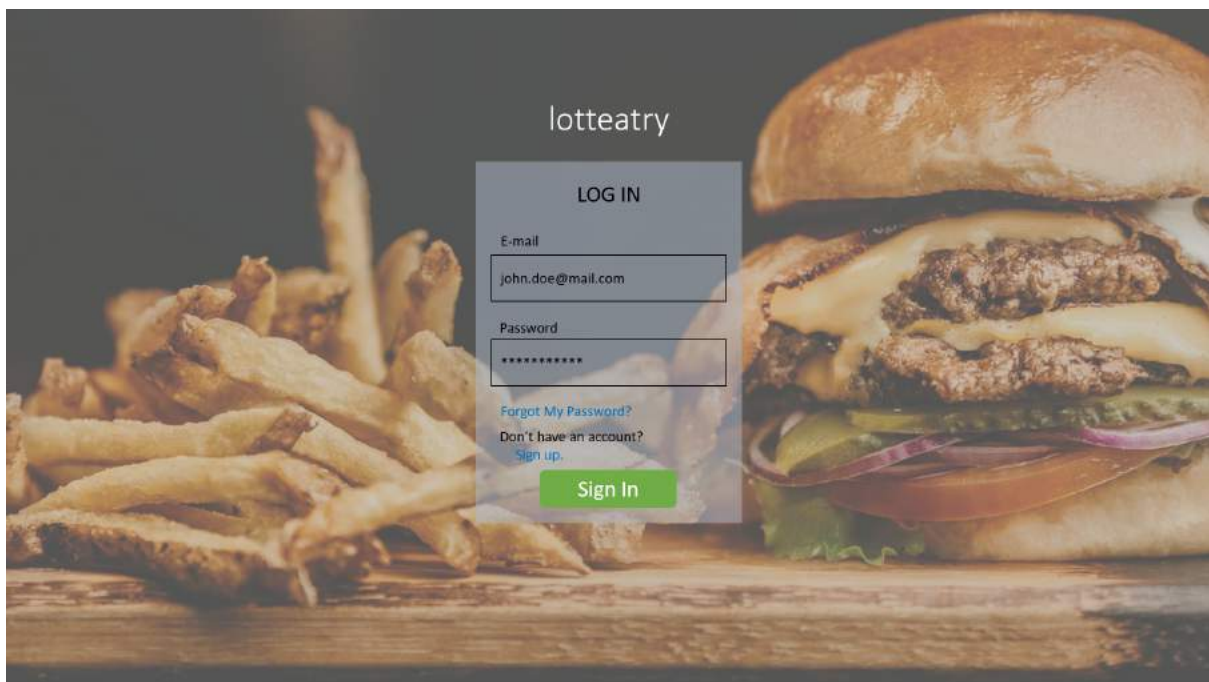
```
--@user_id is the auto-incremented id of newly created User above
INSERT INTO Restaurant (owner_id, restaurant_name) VALUES (@user_id,
@restaurant_name)
```

Courier Sign-up

```
INSERT INTO User (name, surname, email, username, password, telephone,
registration_date)
VALUES (@name, @surname, @email, @username, @password, @telephone,
CURDATE())
```

```
--@user_id is the auto-incremented id of newly created User above
INSERT INTO Courier (courier_id) VALUES (@user_id)
```

3.1.2. Login Page



General Login For all User Types

```
select *
FROM User
WHERE email = @login_email
AND password = @login_password
```

3.2. Additional requirement - Raffles

3.2.1. Restaurant Starting a Raffle

The screenshot shows a mobile application interface with a 'Raffle' modal open. The modal has a title bar with a red close button. The main content area is divided into two sections: 'No ongoing raffle.' and 'Start A New Raffle'. The 'Start A New Raffle' section contains two rows of input fields. The first row is for 'Start Date' (12, 30, 25.03.2021) and 'Prize Amount' (£ 50,00). The second row is for 'End Date' (00, 00, 29.03.2021) and 'Entry Order Cost' (£ 15,00). A green 'Start Raffle' button is at the bottom. The background shows a 'Pending Orders' list with items like 'John Doe' and 'Jane Doe'.

Raffle

No ongoing raffle.

Start A New Raffle

Start Date: 12 30 25.03.2021

Prize Amount: £ 50,00

End Date: 00 00 29.03.2021

Entry Order Cost: £ 15,00

Start Raffle

The screenshot shows the same mobile application interface, but the 'Raffle' modal now displays a message: 'You have an ongoing raffle, to start a new Raffle please wait for the current one to end.' Below this message is the 'Ongoing Raffle' section, which contains the same input fields as the previous screenshot. The background shows the same 'Pending Orders' list.

Raffle

You have an ongoing raffle, to start a new Raffle please wait for the current one to end.

Ongoing Raffle

Start Date: 12 30 25.03.2021

Prize Amount: £ 50,00

End Date: 00 00 29.03.2021

Entry Order Cost: £ 15,00

Starting a new Raffle

```
INSERT INTO Raffle (starting_date, ending_date, min_entry_price,
coupon_prize, restaurant_id)
VALUES (@starting_date, @ending_date, @min_entry_price, @coupon_prize,
@restaurant_id)
```

Getting The Current Raffle Info

```
SELECT * FROM Raffle
WHERE restaurant_id = @restaurant_id AND (NOW() BETWEEN starting_date AND
ending_date)
```

Giving the Coupon To Winner

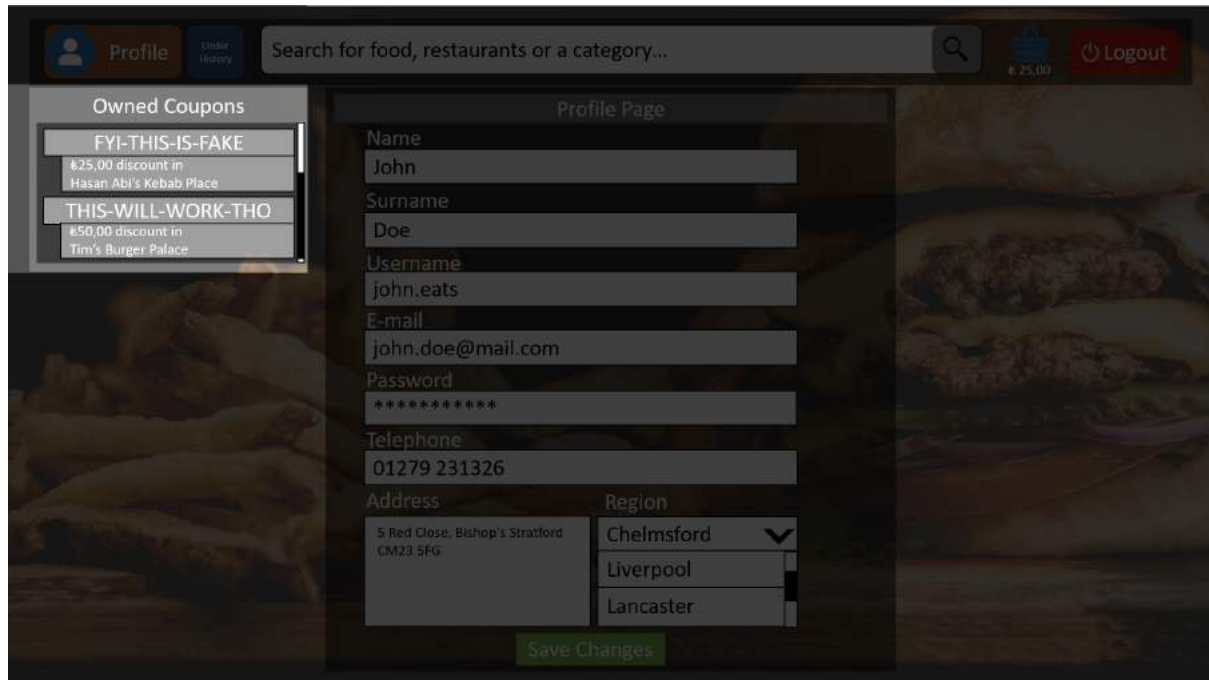
ID of the winner “@winner_user_id” is determined in the back-end when the raffle time ends and their coupon is given as:

```
INSERT INTO Coupon (discount_amount, customer_id, used, restaurant_id)
VALUES (@discount_amount, @winner_user_id, 0, @restaurant_id)
```

```
UPDATE Participates
SET status = 'Won'
WHERE user_id = @winner_user_id
```

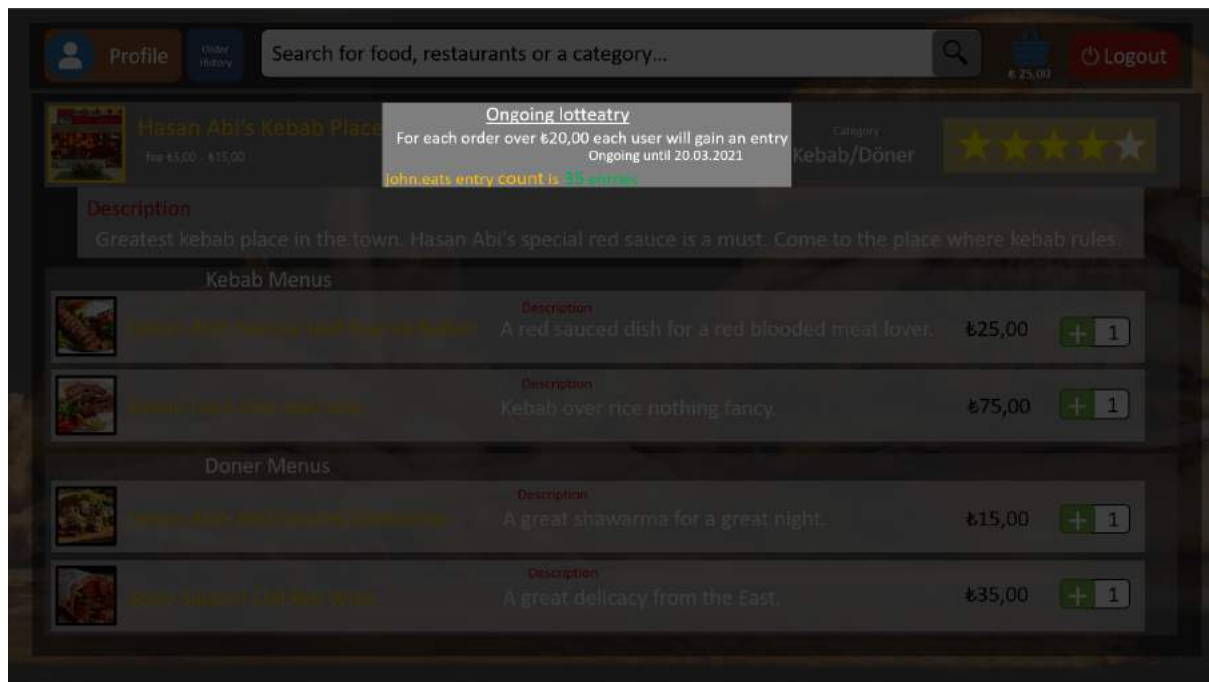
```
UPDATE Participates
SET status = 'Lost'
WHERE user_id <> @winner_user_id
```

3.2.2. Customer Seeing Their Coupons In Their Profile



```
SELECT coupon_id, discount_amount, restaurant_name
FROM Coupon
INNER JOIN Restaurant ON Coupon.restaurant_id = Restaurant.restaurant_id
WHERE customer_id = @customer_id AND used <> 1
```

3.2.3. Current Raffle Details of a Restaurant

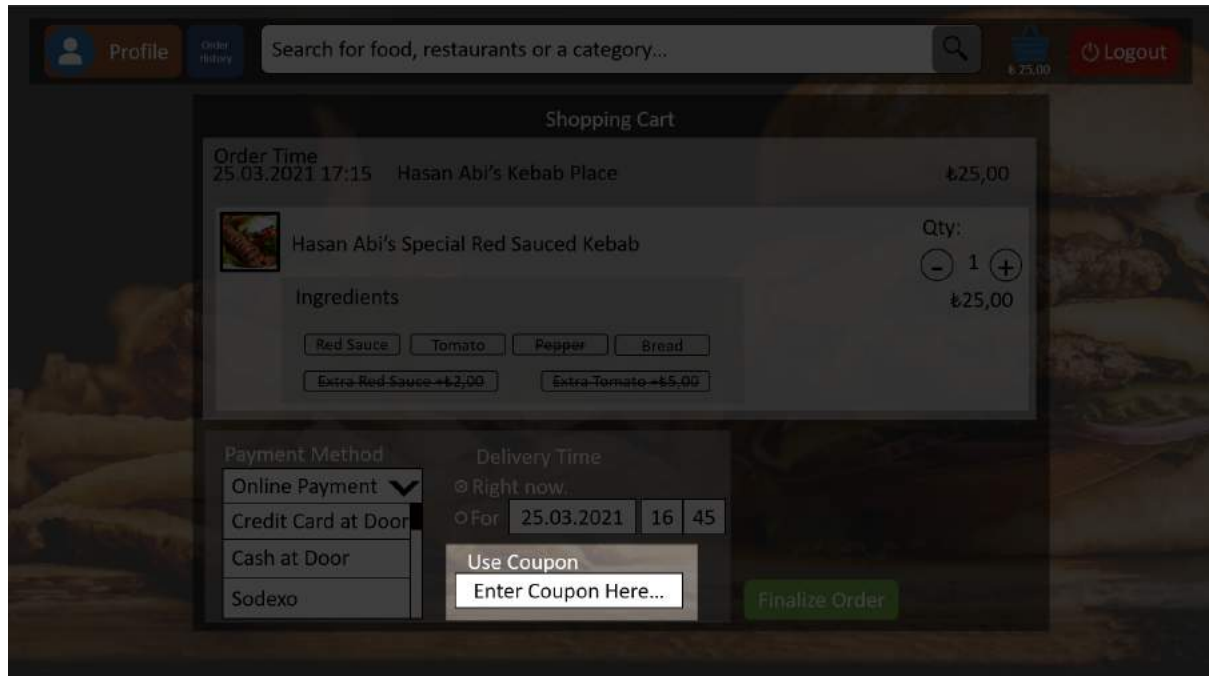


```

SELECT    starting_date,    ending_date,    min_entry_prize,    coupon_prize,
num_entries
FROM Raffle
INNER JOIN Restaurant ON Raffle.restaurant_id = Restaurant.restaurant_id
INNER JOIN Participates ON Raffle.raffle_id = Participates.raffle_id
WHERE restaurant_id = @restaurant_id AND @customer_id = customer_id
AND (NOW() BETWEEN starting_date AND ending_date)

```

3.2.4. Customer Applying a Coupon When Finalizing an Order

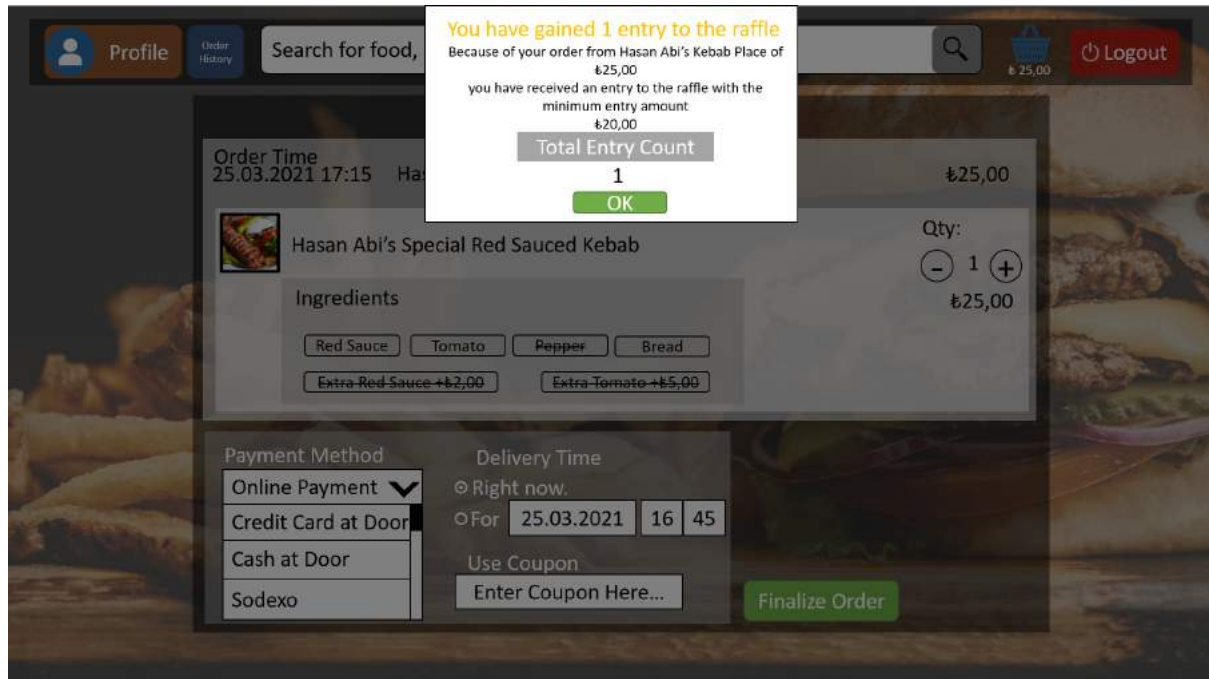


After The Order is Created:

```
INSERT INTO Applied_Coupon
VALUES (@coupon_id, @order_id)
```

```
UPDATE Coupon
SET used = 1
WHERE coupon_id = @coupon_id
```

3.2.5. Customer Gaining New Entry to a Raffle



To see whether the customer already has entries in the raffle, use

```
SELECT EXISTS (SELECT *  
FROM Participates WHERE user_id = @user_id)
```

If customer already have entries

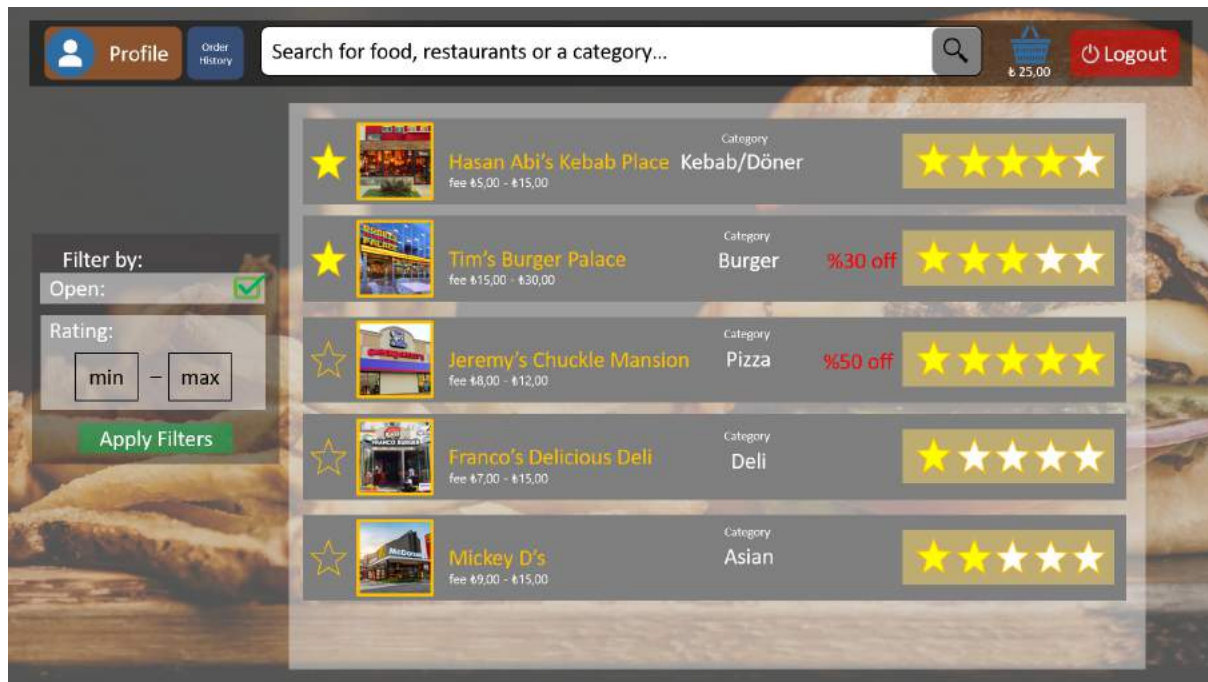
```
UPDATE Participates  
SET num_entries = num_entries + @new_entries  
WHERE user_id = @user_id
```

If customer does not have any entries

```
INSERT INTO Participates  
VALUES (@user_id, @raffle_id, @new_entries, 'WAITING')
```

3.3. Customer User Interface

3.3.1. Restaurant List Page



Listing Restaurants

Two separate select statements get the favorited and non-favorited restaurants from the database. The result of these queries are then merged and listed to the customer, displaying the “favorited” restaurants first.

```
SELECT restaurant_name, rating, restaurant_category,  
MIN(Operates_In.fee), MAX(Operates_In.fee), Restaurant.status, FROM  
Restaurant  
INNER JOIN Serves_At ON Serves_At.restaurant_id = Restaurant.restaurant_id  
INNER JOIN Region ON Region.region_id = Serves_At.region_id  
INNER JOIN Lives_In ON Lives_In.region_id = Region.region_id  
INNER JOIN Customer ON Customer.user_id = Lives_In.user_id  
INNER JOIN Operates_In ON Operates_in.region_id = Region.region_id  
INNER JOIN Courier ON Courier.courier_id = Operates_In.courier_id  
WHERE restaurant_id IN  
(SELECT DISTINCT restaurant_id FROM RESTAURANT  
INNER JOIN Serves_At ON Serves_At.restaurant_id = Restaurant.restaurant_id  
INNER JOIN Region ON Region.region_id = Serves_At.region_id  
INNER JOIN Lives_In ON Lives_In.region_id = Region.region_id  
INNER JOIN Customer ON Customer.user_id = Lives_In.user_id  
INNER JOIN Operates_In ON Operates_in.region_id = Region.region_id  
INNER JOIN Courier ON Courier.courier_id = Operates_In.courier_id
```

```

WHERE (Customer.user_id = @user_id)
AND (Courier.status = 'available')
AND (Restaurant.status = @status)
AND rating BETWEEN @rating_min AND @rating_max
AND restaurant_id IN
(SELECT restaurant_id FROM Restaurant
INNER JOIN Favorites ON Favorites.restaurant_id = Restaurant.restaurant_id
WHERE Favorites.user_id = session_user_id)
)

```

```

SELECT restaurant_name, rating, restaurant_category,
MIN(Operates_In.fee), MAX(Operates_In.fee), Restaurant.status, FROM
Restaurant
INNER JOIN Serves_At ON Serves_At.restaurant_id = Restaurant.restaurant_id
INNER JOIN Region ON Region.region_id = Serves_At.region_id
INNER JOIN Lives_In ON Lives_In.region_id = Region.region_id
INNER JOIN Customer ON Customer.user_id = Lives_In.user_id
INNER JOIN Operates_In ON Operates_In.region_id = Region.region_id
INNER JOIN Courier ON Courier.courier_id = Operates_In.courier_id
WHERE restaurant_id IN
(SELECT DISTINCT restaurant_id FROM Restaurant
INNER JOIN Serves_At ON Serves_At.restaurant_id = Restaurant.restaurant_id
INNER JOIN Region ON Region.region_id = Serves_At.region_id
INNER JOIN Lives_In ON Lives_In.region_id = Region.region_id
INNER JOIN Customer ON Customer.user_id = Lives_In.user_id
INNER JOIN Operates_In ON Operates_In.region_id = Region.region_id
INNER JOIN Courier ON Courier.courier_id = Operates_In.courier_id
WHERE (Customer.user_id = @user_id)
AND (Courier.status = 'available')
AND (Restaurant.status = @status)
AND rating BETWEEN @rating_min AND @rating_max
AND restaurant_id NOT IN
(SELECT restaurant_id FROM Restaurant
INNER JOIN Favorites ON Favorites.restaurant_id = Restaurant.restaurant_id
WHERE Favorites.user_id = session_user_id)
)

```

Displaying Ongoing Discounts

```

SELECT discount_percentage, restaurant_id FROM Discount
WHERE (NOW() BETWEEN starting_date AND ending_date)

```

Searching From The Searchbar at the Top

```

SELECT restaurant_name, rating, restaurant_category,
MIN(Operates_In.fee), MAX(Operates_In.fee), Restaurant.status,
discount_percentage FROM Restaurant
INNER JOIN Discount ON Discount.restaurant_id = Restaurant.restaurant_id
INNER JOIN Serves_At ON Serves_At.restaurant_id = Restaurant.restaurant_id

```



```

INNER JOIN Region ON Region.region_id = Serves_At.region_id
INNER JOIN Lives_In ON Lives_In.region_id = Region.region_id
INNER JOIN Customer ON Customer.user_id = Lives_In.user_id
INNER JOIN Operates_In ON Operates_in.region_id = Region.region_id
WHERE restaurant_id IN
(SELECT DISTINCT restaurant_id FROM Restaurant
INNER JOIN Serves_At ON Serves_At.restaurant_id = Restaurant.restaurant_id
INNER JOIN Region ON Region.region_id = Serves_At.region_id
INNER JOIN Lives_In ON Lives_In.region_id = Region.region_id
INNER JOIN Customer ON Customer.user_id = Lives_In.user_id
INNER JOIN Operates_In ON Operates_in.region_id = Region.region_id
INNER JOIN Menu_Item ON Menu_Item.restaurant_id = Restaurant.restaurant_id
WHERE restaurant_name LIKE '%@search_input%'
OR restaurant_category LIKE '%@search_input%'
OR Menu_Item.name LIKE '%@search_input%'
)

```

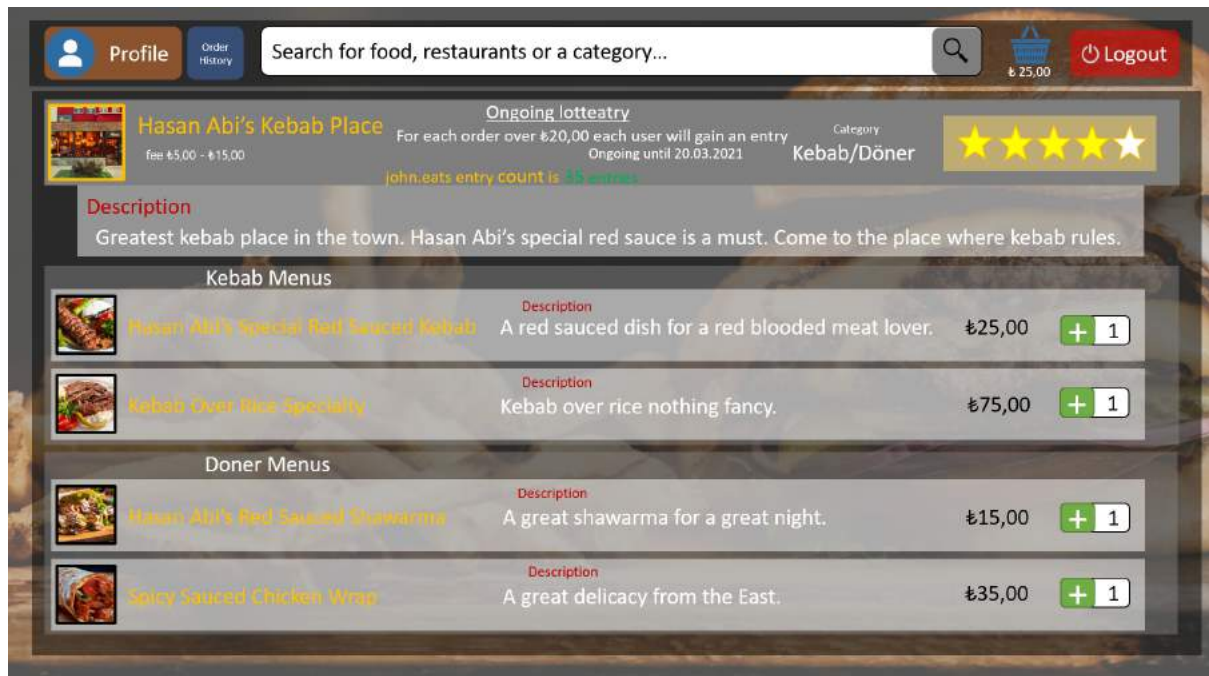
Adding a new Favorite Restaurant

```

INSERT INTO Favorites VALUES (@customer_id, @restaurant_id)

```

3.3.2. Restaurant Menu Page



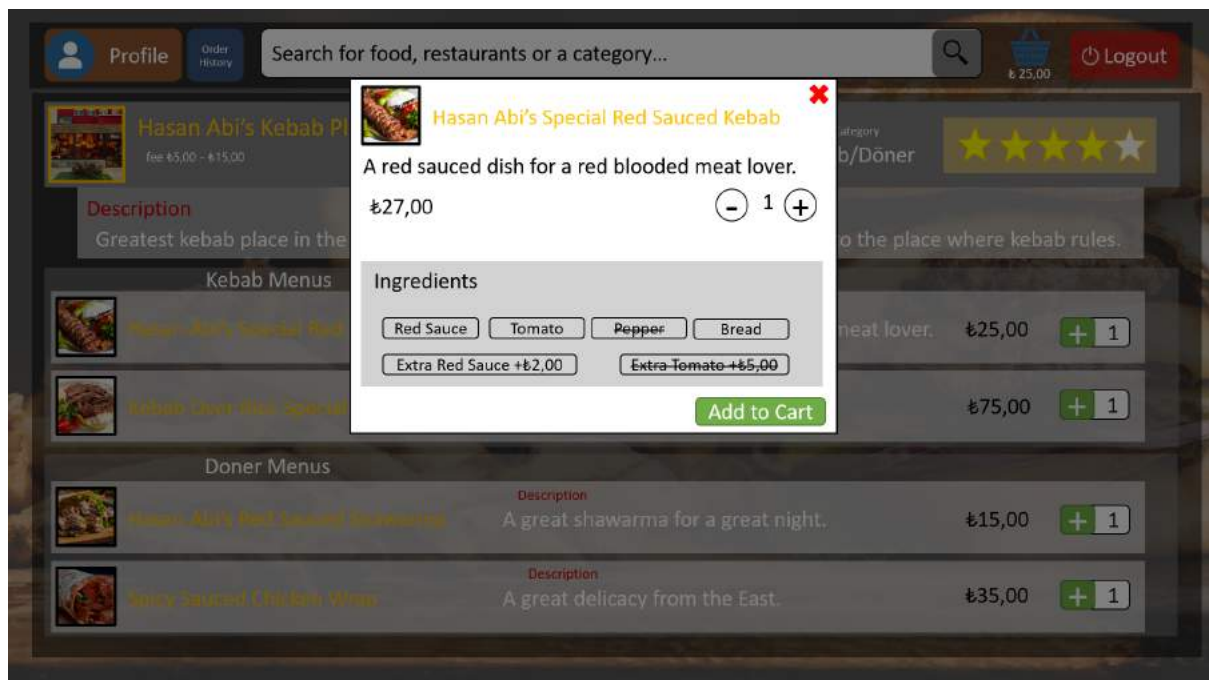
Restaurant Information

```
SELECT name, image, restaurant_category, rating, description
FROM Restaurant
WHERE restaurant_id = @restaurant_id
```

Menu Items

```
SELECT name, image, base_price, food_category, description
FROM Menu_Item INNER JOIN Restaurant ON Restaurant.restaurant_id =
Menu_Item.restaurant_id
WHERE restaurant_id = @restaurant_id
ORDER BY food_category
```

3.3.3. Adding Item To Cart



Food and Ingredients Information

```
SELECT * FROM Menu_Item
WHERE menu_item_id = @menu_item_id
```

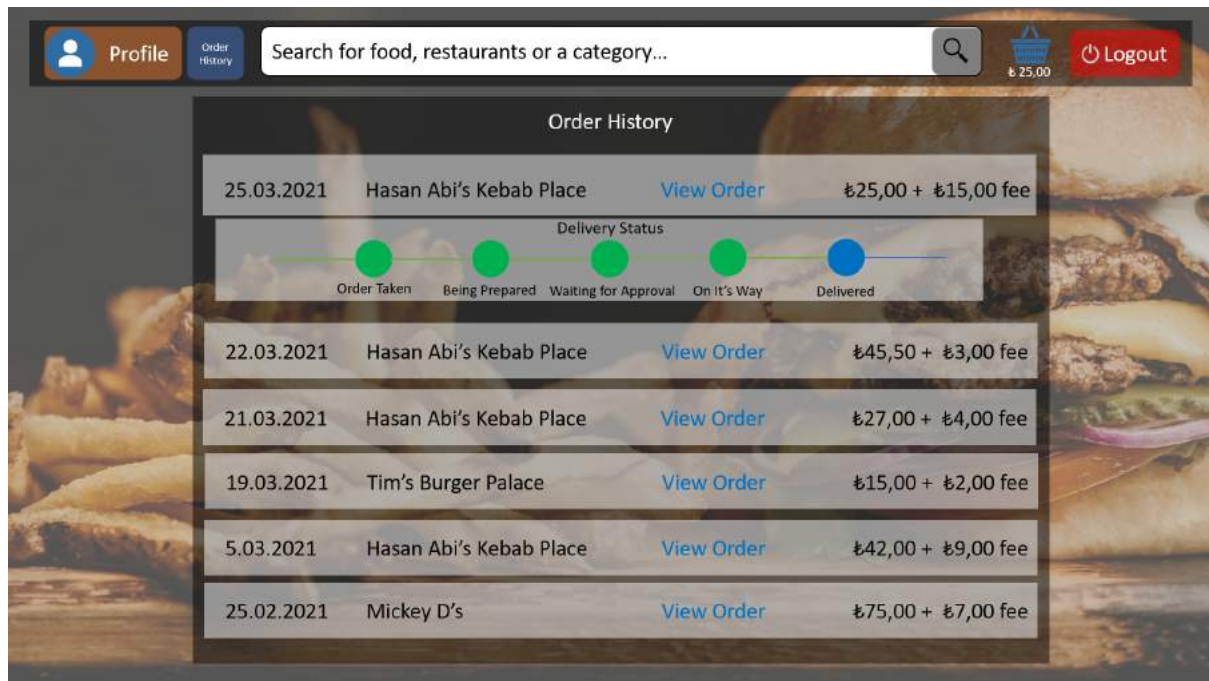
```
SELECT * FROM Ingredients
WHERE menu_item_id = @menu_item_id
```

Adding to the Cart

```
INSERT INTO Ordered_Item (price_with_ingredients)
VALUES (@price_with_ingredients)
--@ordered_item_id is the auto-incremented id of newly created Ordered_Item
above
Then, for each ingredient selected on the screen: @ingredient_id
    INSERT INTO Selected_Ingredients VALUES (@ordered_item_id,
@ingredient_id)

INSERT INTO Food_Cart (user_id, ordered_item_id, restaurant_id)
VALUES (@user_id, @ordered_item_id, @restaurant_id)
```

3.3.4. Order History Page



Getting the Order History

```
SELECT order_id, price, order_time, delivery_time, status, courier_tip,
restaurant_tip, payment_method,
restaurant_name, restaurant_id FROM Order
INNER JOIN Restaurant ON Restaurant.restaurant_id = Order.restaurant_id
WHERE user_id = @user_id
ORDER BY order_time DESC
```

3.3.5. Profile Page

Profile Page

Name: John

Surname: Doe

Username: john.eats

E-mail: john.doe@mail.com

Password: *****

Telephone: 01279 231326

Address: 5 Red Close, Bishop's Stratford CM23 5FG

Region: Chelmsford (dropdown menu with options: Liverpool, Lancaster)

Profile Picture: Click to upload a new picture

Save Changes

Getting the Customer Information

```
SELECT *  
from User INNER JOIN Customer ON User.user_id = Customer.customer_id  
WHERE user_id = @user_id
```

Getting The Region List

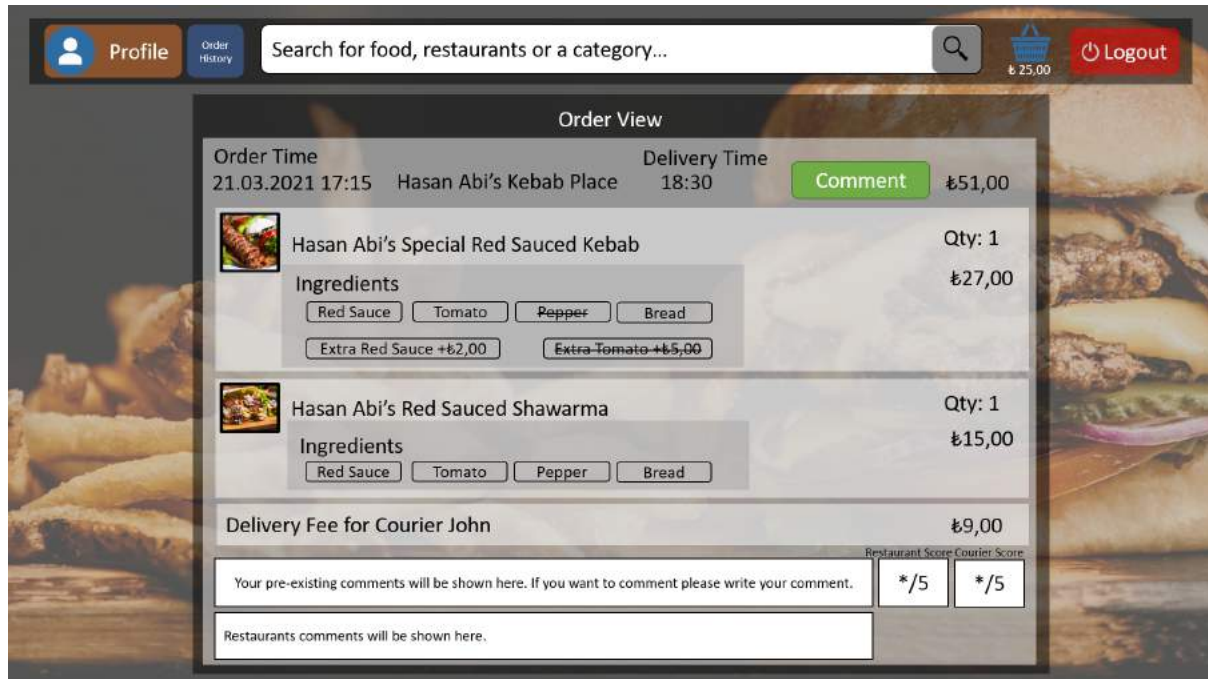
```
SELECT name FROM Region
```

Updating The Customer Information

```
UPDATE User  
SET email = @new_email,  
    username = @new_username,  
    password = @new_password,  
    telephone = @new_telephone,  
    name = @new_name,  
    surname = @new_surname,  
    region_id = @new_region_id,  
    image = @new_image_link,  
WHERE user_id = @user_id
```

```
UPDATE Customer  
SET address = @new_address  
WHERE customer_id = @user_id
```

3.3.6. Order Details and Review Page



Order Details

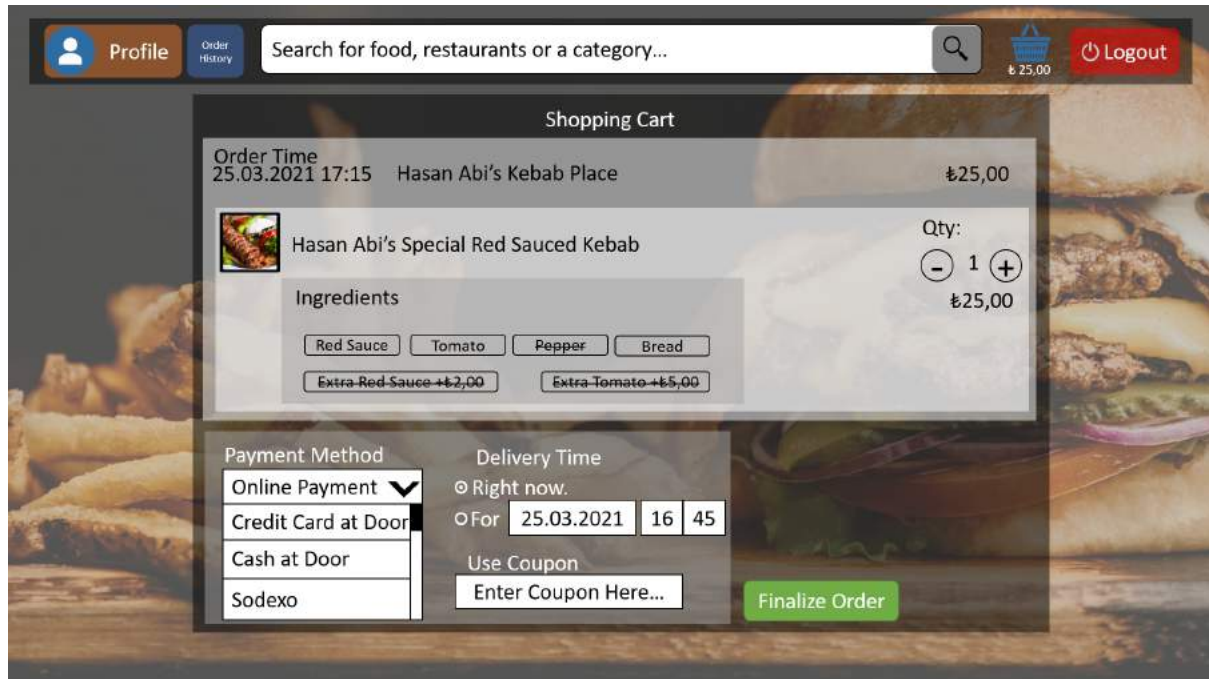
```
SELECT order_id, restaurant_id, restaurant_name, order_time, delivery_time,
status, courier_tip, restaurant_tip
FROM Order
INNER JOIN Restaurant ON Restaurant.restaurant_id = Order.restaurant_id
WHERE Order.order_id = @order_id
```

```
SELECT order_id, ordered_item_id, quantity, price_with_ingredients,
menu_item_id, ingredient_id, ingredient_name
FROM Order
INNER JOIN Contains ON Contains.order_id = Order.order_id
INNER JOIN Ordered_Item ON Ordered_Item.ordered_item_id =
Contains.ordered_item_id
INNER JOIN Selected_Ingredients ON Selected_Ingredients.ordered_item_id =
Ordered_Item.ordered_item_id
WHERE Order.order_id = @order_id
```

Adding a Review

```
INSERT INTO Review (date, restaurant_rating, courier_rating, comment,
order_id)
VALUES (@date, @restaurant_rating, @courier_rating, @comment, @order_id)
```

3.3.7. Customer Finalize Order Page



Customers can view the selected items in their cart and click the finalize order button to create the order.

Creating New Order in Database

```
INSERT INTO Order(restaurant_id, customer_id, price, order_time,
payment_method, status, optional_delivery_time)
VALUES (@restaurant_id, @user_id, @total_price, CURRENT_TIMESTAMP,
@payment_method, @status, @opt_time)
```

```
INSERT INTO Contains (order_id, ordered_item_id, quantity)
SELECT @order_id AS order_id, ordered_item_id, quantity
FROM Food_Cart
```

```
--Empty the food cart
DELETE FROM Food_Cart
```


3.4. Restaurant User Interface

For the time being we also did the mockups for the restaurant administrative and menu pages.

3.4.1. Restaurant Administrative Page

The mockup displays a restaurant administrative interface. The top navigation bar includes 'Profile', 'Orders', and 'Menu' buttons, along with 'Raffle' and 'Logout' links. The left sidebar contains 'Served Regions' (London, Liverpool, Chelmsford, Brighton, Chelsea) and 'Restaurant Properties' (Open status, Discount, Address, Telephone). The main content area features a 'Pending Orders' table and a 'Delivery Status' timeline.

Served Regions

- London
- Liverpool
- Chelmsford
- Brighton
- Chelsea

Restaurant Properties

Open: ☐ ☒ Discount: %10
Address: 7 Red Far, Jeremy Road L12 5FC
Telephone: 05225 241425


Pending Orders

Date	Time	Customer	Order Details	Status	Amount
25.03.2021	17:15	John Doe	5 Red Close, Bishop's Stratford CM23 5FG	Pending	£25,00
25.03.2021	15:15	Jane Doe	5 Blue Close, Bishop's Stratford CM25 7FG	Finished	£21,00
24.03.2021	19:45	Jane Doe	5 Blue Close, Bishop's Stratford CM25 7FG	Finished	£35,00
22.03.2021	16:15	John Doe	5 Red Close, Bishop's Stratford CM23 5FG	Finished	£45,50
21.03.2021	18:35	John Doe	5 Red Close, Bishop's Stratford CM23 5FG	Finished	£27,00
20.03.2021	15:35	Jane Doe	5 Blue Close, Bishop's Stratford CM25 7FG	Finished	£22,00


Delivery Status

Order Taken → Being Prepared → Waiting for Approval → On It's Way → Delivered


3.4.2. Restaurant Modify Menu Page



Profile



Orders



Menu


Category name

Add Category

Logout

Kebab Menu

Add Item



Hasan Abi's Special Red Sauced Kebab

Description

A red sauced dish for a red blooded meat lover.

Price

₺25,00

Ingredients

Red Sauce

Tomato

Pepper

Bread

Extra Red Sauce +₺2,00

Extra Tomato +₺5,00


New Ingredient

Ingredient Name:

Price: ☒ Default ☐ Enter Price...

Add

Save



Kebab Over Rice Specialty

Description

Kebab over rice nothing fancy.

Price

₺75,00

Ingredients

Red Sauce

Tomato

Pepper

Bread

Extra Red Sauce +₺2,00

Extra Tomato +₺5,00

New Ingredient

Ingredient Name:

Price: ☒ Default ☐ Enter Price...

Add

Save

Doner Menu

Add Item



Hasan Abi's Red Sauced Shawarma

Description

A great shawarma for a great night.

Price

₺15,00

Ingredients

Red Sauce

Tomato

Pepper

Bread

Extra Red Sauce +₺2,00

Extra Tomato +₺5,00

New Ingredient

Ingredient Name:

Price: ☒ Default ☐ Enter Price...

Add

Save