Department of Computer Engineering

Bilkent University

# CS353 DATABASE SYSTEMS

*Online Restaurant Order System*

*Project URL: https://brker.github.io/hungerbox/*

# Project Design Report

GROUP 33

| | |
|---|---|
| Osman Sefa İbiş | 21301693 |
| İbrahim Berker Kırdök | 21502870 |
| Mustafa Mert Aşkaroğlu | 21400195 |
| Gökalp Köksal | 21400549 |

Submission Date: 02.04.2018

# *TABLE OF CONTENTS*

TABLE OF CONTENTS

# ***1. Revised E/R Model***

## 1.1 Explanation of Changes

According to Mustafa Can Çavdar's review, we revised our E/R model considering the feedback as follows:

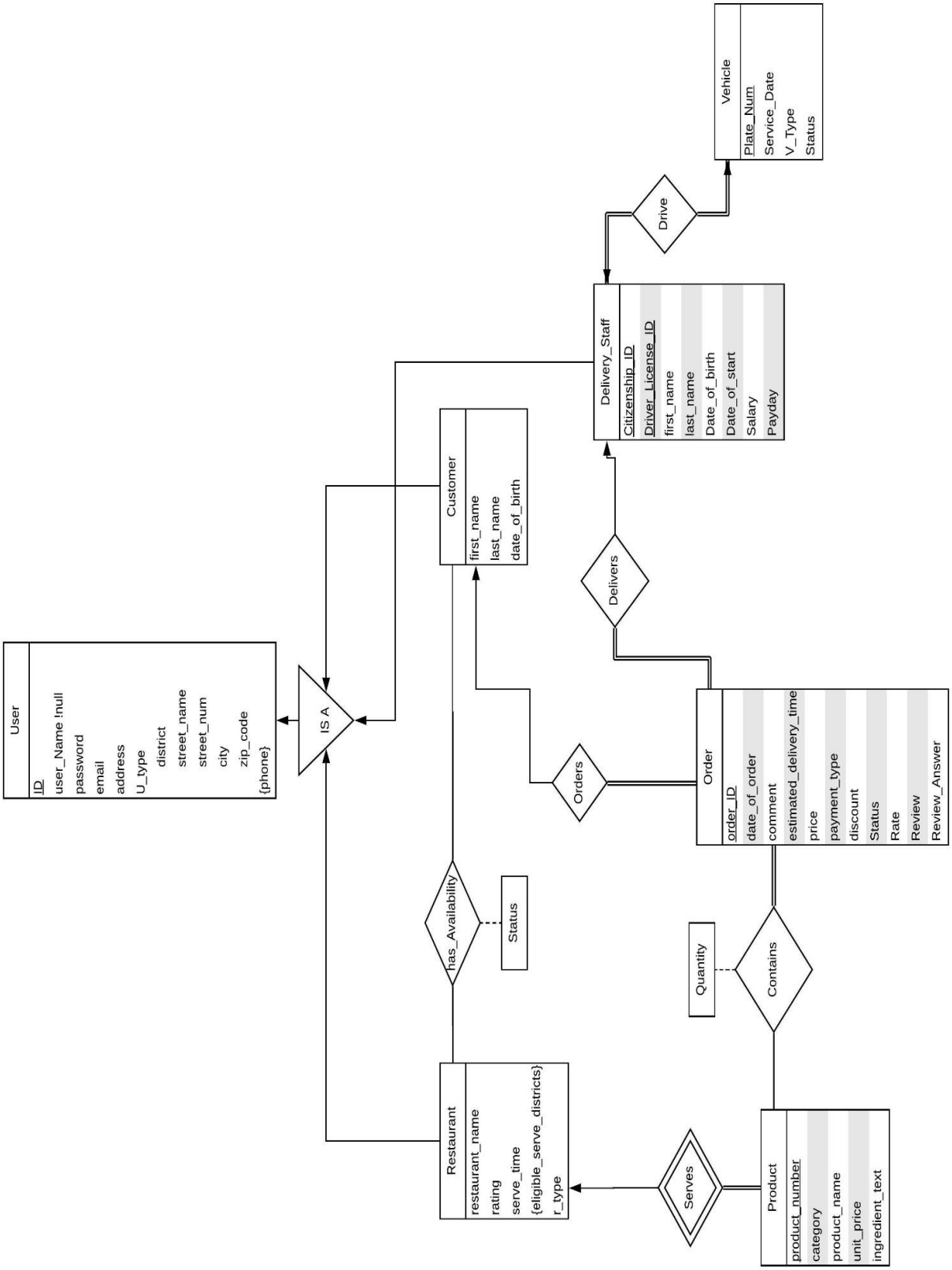
### *1.1.1 Changes*

- Changed functional attribute price() to non-functional attribute, also deleted age() attribute since it conflicts with the date_of_birth attribute.
- We changed "Order" entity's participation in 'Contains', 'Orders' and 'Delivers' relationships as total participation.
- We changed Delivery_Staff's and Vehicle's entity's participation in 'Drive' relationship as total participation.
- Changed the availability relation name to has_availability for convenience.
- Oder_time_average attribute in Restaurant entity is deleted.

### *1.1.2 Additions*

- We added the attribute "rate" in Order entity.
- We added "U_type" attribute to determine the user type for convenience in User entity.
- Driver_License_ID and Citizenship_ID are underlined in E/R diagram to specify that they are primary keys for the Delivery_Staff entity.
- Status attribute meaning closed or open in restaurant entity is added to the has_availability relation.
- "Review_Answer" attribute is added to the order entity. The restaurants can answer their reviews of the orders.

## 1.2 Revised E/R Diagram



# ***2. Relation Schemas***

## 2.1 User

**Relational Model:**

User( ID, user_Name, password, email, U_type, district, street_name, street_Num, city, zip_code)

**Functional Dependencies:**

ID → user_Name password email U_type district street_name street_Num city zip_code

**Candidate Keys:**

{ { ID } }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE User(
        ID              int PRIMARY KEY AUTO_INCREMENT,
        User_Name       varchar(30) NOT NULL,
        Password        varchar(20) NOT NULL,
        Email           varchar(40) NOT NULL,
        U_Type          varchar(50) NOT NULL,
        District        varchar(100) NOT NULL ,
        Street_name     varchar(100) NOT NULL,
        Street_num      int,
        City            varchar(50) NOT NULL,
        Zip_code        int );
```

## 2.2 Phone

**Relational Model:**

Phone(ID, phone_num)

**Functional Dependencies:**

ID→phone_num

**Candidate Keys:**

{ { ID } }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE phone(
        ID          int PRIMARY KEY,
        Phone_num   int NOT NULL,
        FOREIGN KEY (ID) references User(ID) );
```

## 2.3 Customer

**Relational Model:**

Customer( <u>customer_ID</u>, first_name, last_name, date_of_birth)

**Functional Dependencies:**

customer_ID→ first_name last_name date_of_birth

**Candidate Keys:**

{ { customer_ID } }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Customer(
        customer_ID         int PRIMARY KEY,
        First_name          varchar(50) NOT NULL,
        Last_name           varchar(50) NOT NULL,
        Date_of_birth       date NOT NULL,
        FOREIGN KEY(customer_ID) references User(ID));
```

## 2.4 Restaurant

**Relational Model:**

Restaurant ( restaurant_ID, restaurant_name, rating, serve_time, , r_type)

**Functional Dependencies:**

restaurant_ID→ restaurant_name rating serve_time r_type

**Candidate Keys:**

{{ restaurant_ID }}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Restaurant(
        Restaurant_ID           int PRIMARY KEY,
        Restaurant_name         varchar(30) NOT NULL,
        Rating                  int,
        Serve_time              int,
        R_type                  varchar(50) NOT NULL,
        FOREIGN KEY (restaurant_ID) references User(ID));
```

## 2.5 Eligible_serve_districts

**Relational Model:**

Eligible_serve_districts( restaurant_ID, districts)

**Functional Dependencies:**

restaurant_ID→ district

**Candidate Keys:**

{{ restaurant_ID }}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Eligible_serve_districts(
        restaurant_ID        int PRIMARY KEY,
        districts            varchar(200) NOT NULL,
        FOREIGN KEY (restaurant_ID) references User(ID));
```

## 2.6 Delivery_Staff

**Relational Model:**

Delivery_Staff( Staff_ID, Citizenship_ID, Driver_License_ID, first_name, last_name date_of_birth, date_of_start, salary, payday)

**Functional Dependencies:**

Staff_ID,Citizenship_ID, Driver_License_ID → date_of_birth date_of_start salary payday

**Candidate Keys:**

{{ Staff_ID,Citizenship_ID, Driver_License_ID}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Delivery_Staff(
        Staff_ID            int PRIMARY KEY,
        Citizenship_ID      int PRIMARY KEY,
        Driver_License_ID   int PRIMARY KEY,
        First_name          varchar(50) NOT NULL,
        Last_name           varchar(50) NOT NULL,
        Date_of_birth       date NOT NULL,
        Salary              int,
        Payday              int,
        FOREIGN KEY(staff_ID) references User(ID));
```

## 2.7 Order

**Relational Model:**

Order( Order_ID, date_of_order, comment, estimated_delivery_time, price, payment_type, discount, status, rate, review, review_Answer, Citizenship_ID, Driver_License_ID, customer_ID,Staff_ID)

**Functional Dependencies:**

Order_ID→ date_of_order comment estimated_delivery_time price payment_type discount status rate review review_Answer

**Candidate Keys:**

{{Order_ID}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Order(
        Order_ID                    int PRIMARY KEY AUTO_INCREMENT,
        Date_of_order               date NOT NULL,
        Comment                     varchar(200),
        Estimated_delivery_time     int,
        Price                       float NOT NULL,
        Payment_type                varchar(50) NOT NULL,
        Discount                    int,
        Status                      varchar(50),
        Review                      varchar(150),
        Review_Answer               varchar(150),
        Rate                        int,
        FOREIGN KEY( Citizenship_ID) references Delivery_Staff(Citizenship_ID),
        FOREIGN KEY( Driver_License_ID) references Delivery_Staff(Driver_License_ID),
        FOREIGN KEY( customer_ID) references User(ID));
        FOREIGN KEY( Staff_ID) references User(ID));
```

## 2.8 Product

**Relational Model:**

Product( product_number, category, product_name, unit_price, ingredient_text, restaurant_ID)

**Functional Dependencies:**

Product_number→ category product_name unit_price ingredient_text

**Candidate Keys:**

{{ product_number}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Product(
        product_number      int PRIMARY KEY AUTO_INCREMENT,
        category            varchar(50) NOT NULL,
        product_name        varchar(100) NOT NULL,
        unit_price          int NOT NULL,
        ingredient_text     varchar(200),
        FOREIGN KEY(restaurant_ID) references User(ID));
```

## 2.9 Vehicle

**Relational Model:**

Vehicle( Plate_num , service_date, V_type, status)

**Functional Dependencies:**

Plate_num→ service_date, V_type, status

**Candidate Keys:**

{{ plate_num }}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Vehicle(
        Plate_num       varchar(20) PRIMARY KEY,
        Service_date    date,
        V_Type          varchar(30) NOT NULL,
        Status          varchar(30));
```

## 2.10 Avaliability

**Relational Model:**

Availability( Restaurant_ID, Customer_ID,status)

**Functional Dependencies:**

Restaurant_ID, Customer_ID→ status

**Candidate Keys:**

{{ Restaurant_ID, Customer_ID }}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Availability(
        Customer_ID         int NOT NULL,
        Restaurant_ID       int NOT NULL,
        Status              varchar(30) NOT NULL,
        PRIMARY KEY(Customer_ID, Restaurant_ID),
        FOREIGN KEY(Customer_ID) references User(ID),
        FOREIGN KEY(Restaurant_ID) references User(ID));
```

## 2.11 Orders

**Relational Model:**

Orders( customer_ID, order_ID)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{{ customer_ID, order_ID}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Orders(
        customer_ID         int NOT NULL,
        Order_ID            int NOT NULL,
        PRIMARY KEY(customer_ID, Order_ID),
        FOREIGN KEY (customer_ID) references User(ID),
        FOREIGN KEY (order_ID) references Order(Order_ID));
```

## 2.12 Delivers

**Relational Model:**

Delivers( order_ID, staff_ID ,Citizenship_ID, Driver_License_ID)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{{ order_ID, Citizenship_ID, Driver_License_ID}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Delivers(
        Order_ID            int NOT NULL,
        Staff_ID            int NOT NULL,
        Citizenship_ID      int NOT NULL,
        Driver_License_ID   int NOT NULL,
        PRIMARY KEY(Order_ID, Staff_ID, Citizenship_ID, Driver_License_ID),
        FOREIGN KEY (Order_ID) references Order(Order_ID),
        FOREIGN KEY (Staff_ID) references User(ID),
        FOREIGN KEY (Citizenship_ID) references Delivery_Staff(Citizenship_ID),
        FOREIGN KEY (Driver_License_ID) references Delivery_Staff (Driver_License_ID));
```

## 2.13 Serves

**Relational Model:**

Serves(restaurant_ID, product_number)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{{ restaurant_ID, product_number}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Serves(
        restaurant_ID        int NOT NULL,
        Product_number       int NOT NULL,
        PRIMARY KEY(restaurant_ID, Product_number),
        FOREIGN KEY (restaurant_ID) references User(ID),
        FOREIGN KEY (Product_number) references Product(Product_number) );
```

## 2.14 Contains

**Relational Model:**

Contains( order_ID, product_number, quantity)

**Functional Dependencies:**

Order_ID,Product_number→quantity

**Candidate Keys:**

{{ Order_ID,Product_number}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Contains(
        Order_ID            int NOT NULL,
        Product_Number      int NOT NULL,
        Quantity            int NOT NULL,
        PRIMARY KEY (Order_ID ,Product_Number ),
        FOREIGN KEY (Order_ID) references Order(Order_ID),
        FOREIGN KEY (Product_Number) references Product(Product_Number) );
```

## 2.15 Drive

**Relational Model:**

Drive(Staff_ID, Citizenship_ID, Driver_License_ID, Plate_num)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{{ Citizenship_ID, Driver_License_ID, Plate_num}}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Drive(
        Citizenship_ID          int NOT NULL ,
        Citizenship_ID          int NOT NULL,
        Plate_num               varchar(20) NOT NULL,
        Staff_ID                int NOT NULL,
        PRIMARY KEY ( Citizenship_ID ,Citizenship_ID, Plate_num ,Staff_ID),
        FOREIGN KEY (Citizenship_ID) references Delivery_Staff(Citizenship_ID),
        FOREIGN KEY (Driver_License_ID) references Delivery_Staff (Driver_License_ID),
        FOREIGN KEY (Staff_ID) references User(ID),
        FOREIGN KEY (Plate_num) references Vehicle(Plate_num));
```

# ***3. Functional Dependencies and Normalization of Tables***

All functional dependencies and normal forms are indicated in Relation Schemas in Section 2 of this Project Design Report. We checked whether all relations in out design are in Boyce-Codd Normal Form (BCNF). We concluded that no decomposition is required.

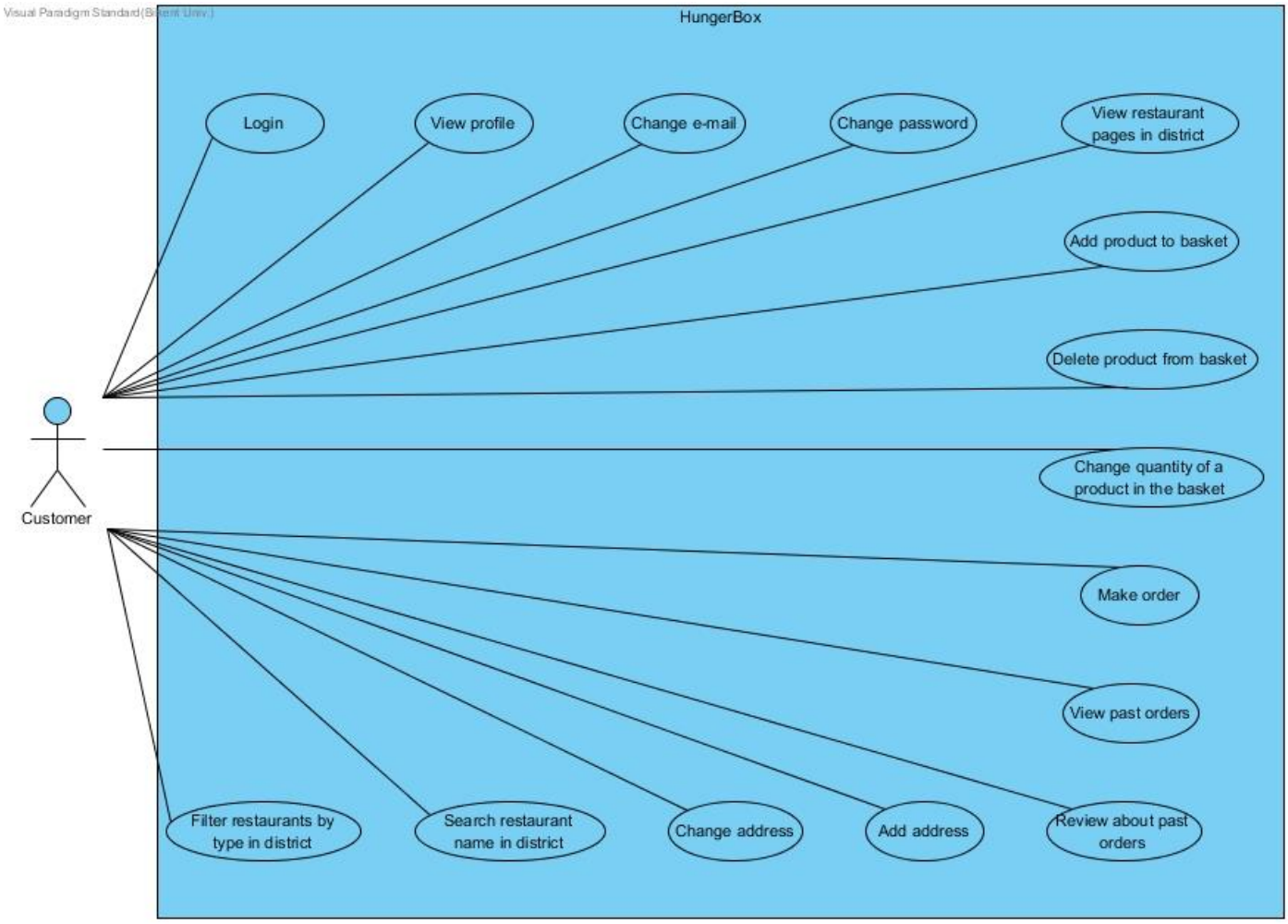
# 4. Functional Component

## 1. Use Cases / Scenarios

In HungerBox System there are three different user types: customer, restaurant and delivery staff. Roles and usable functions for each type is different than each other. In order to use the system, all type of users should sign up first then login the system. The system will provide limited features according to their user type.

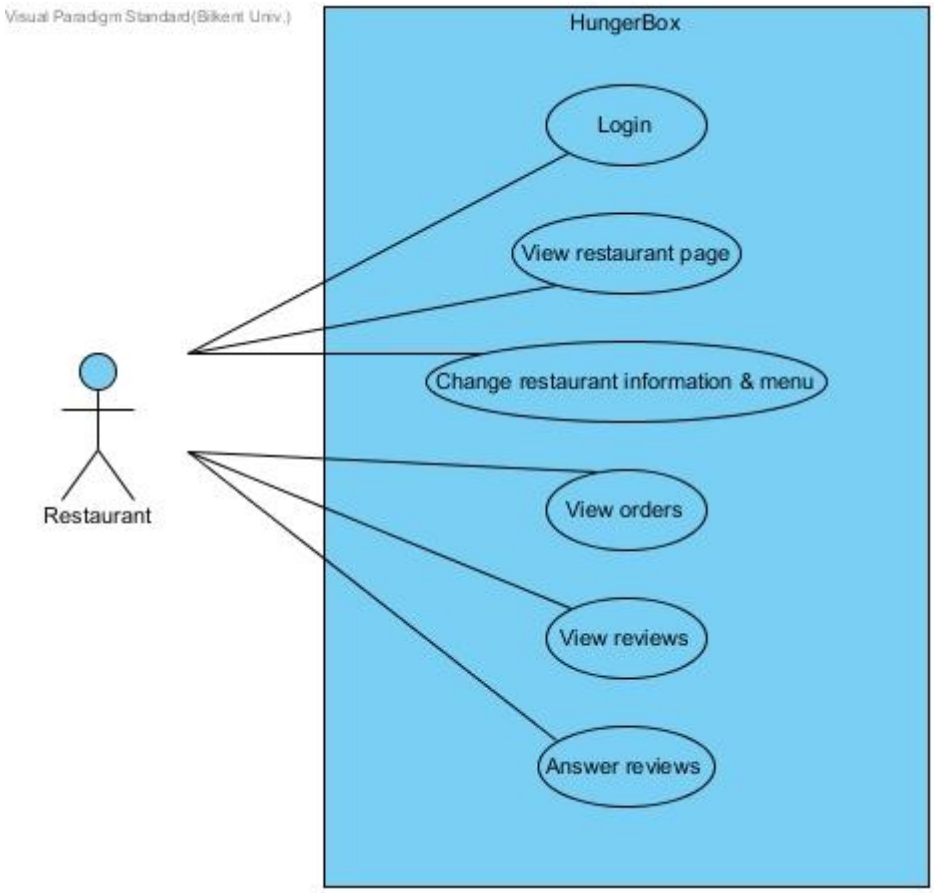
**Customer:**

- Customer can login to the system with email address or username and password.
- Customer can access his profile which includes his username, email address, name, surname, date of birth, address and phone numbers.
- Customer can change his email address, password and address.
- Customer can add or delete phone numbers.
- Customer can view restaurant pages in eligible serve districts which includes restaurant working hours, average service time, menu, reviews and rates.
- Customer can select products from restaurant's menu and add them to his basket.
- Customer can delete products from his basket.
- Customer can change quantity of products in his basket.
- Customer can select payment type.
- Customer can select delivery staff to deliver his product.
- Customer can make an order.
- Customer can view his past orders and rates he has given.
- Customer can review about his past orders.
- Customer can rate his past orders by its delivery time, service and taste.
- Customer can search restaurants by name in his district.
- Customer can filter restaurants by type in district.


Visual Paradigm Standard(Bilkent Univ.)
HungerBox
Login
View profile
Change e-mail
Change password
View restaurant pages in district
Add product to basket
Delete product from basket
Change quantity of a product in the basket
Make order
View past orders
Review about past orders
Add address
Change address
Search restaurant name in district
Filter restaurants by type in district
Customer

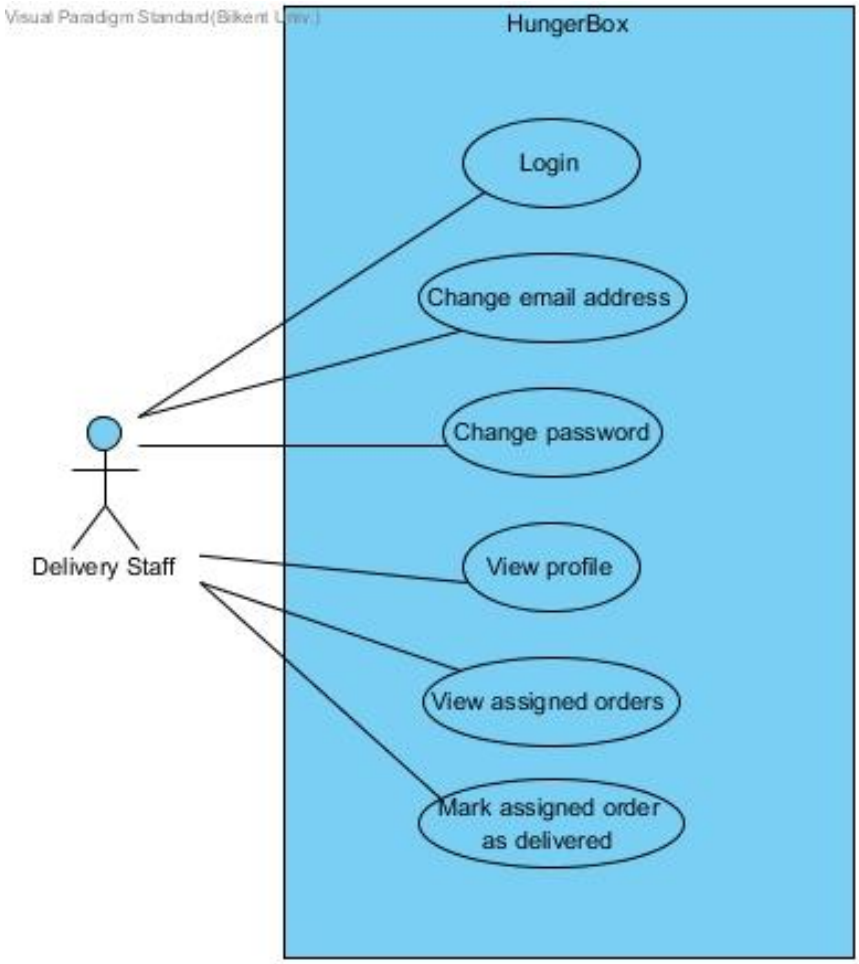
**Restaurant:**

- Restaurant can login with an email address or username and password.
- Restaurant can change email address, username and password.
- Restaurant can view its profile which includes its username, email address, password, address and phone information.
- Restaurant can view its own page which includes restaurant name, menu (it consists of products that restaurant serves and their prices), rating, average service time, status, serving districts and restaurant type.
- Restaurant can change its password, email address, average service time, status, serving districts and menu.
- Restaurant can view coming orders.
- Restaurant can view past orders that are ordered.
- Restaurant can view reviews and rates of customers for past orders.
- Restaurant can answer reviews.



**Delivery Staff:**

- Delivery Staff can login with an email address or username and password.
- Delivery Staff can view his profile which consist of his username, email address, citizenship id, driver license id, date of birth, date of start, salary and payday.
- Delivery Staff can change his email address and password.
- Delivery Staff can view orders that are assigned to him.
- Delivery Staff can mark assigned order as delivered.



### 4.2 Data Structures

For the attribute domains we use Numeric type, Date and Time and String type data types of MySQL.

# *5. Advanced Database Components*

## 5.1 Triggers

- When new rate for an order is applied by a customer, the system automatically calculate new rate average for the restaurant.
- When a customer writes a review for an order, the system displays it in reviews part for the restaurant.
- When a restaurant answers the review of a customer, the system displays this answer below the review.
- When restaurant add new product, it automatically showed in the menu of the restaurant.
- When a customer assigns a delivery staff for his order, the system automatically display the order in assigned orders for the delivery staff.
- When closing hour of the restaurant is passed, the restaurant cannot be displayed any more until opening hour.

## 5.2 Constraints

- The system cannot be used without logging in.
- Users are identified by their IDs.
- Quantity of the product in the basket cannot be zero or negative.
- Reviews cannot be longer than 150 characters.
- Rate cannot be negative or zero.
- Restaurant cannot be without any product.
- Restaurant's answer cannot be longer than 150 characters.
- Customer cannot order if the restaurant service time is not eligible.
- A customer cannot view restaurants that are not in eligible serve districts.
- Company name cannot be longer than 30 characters.
- Username cannot be longer than 30 characters.
- Delivery Staff cannot be assigned for another order when he has already been assigned for an order.

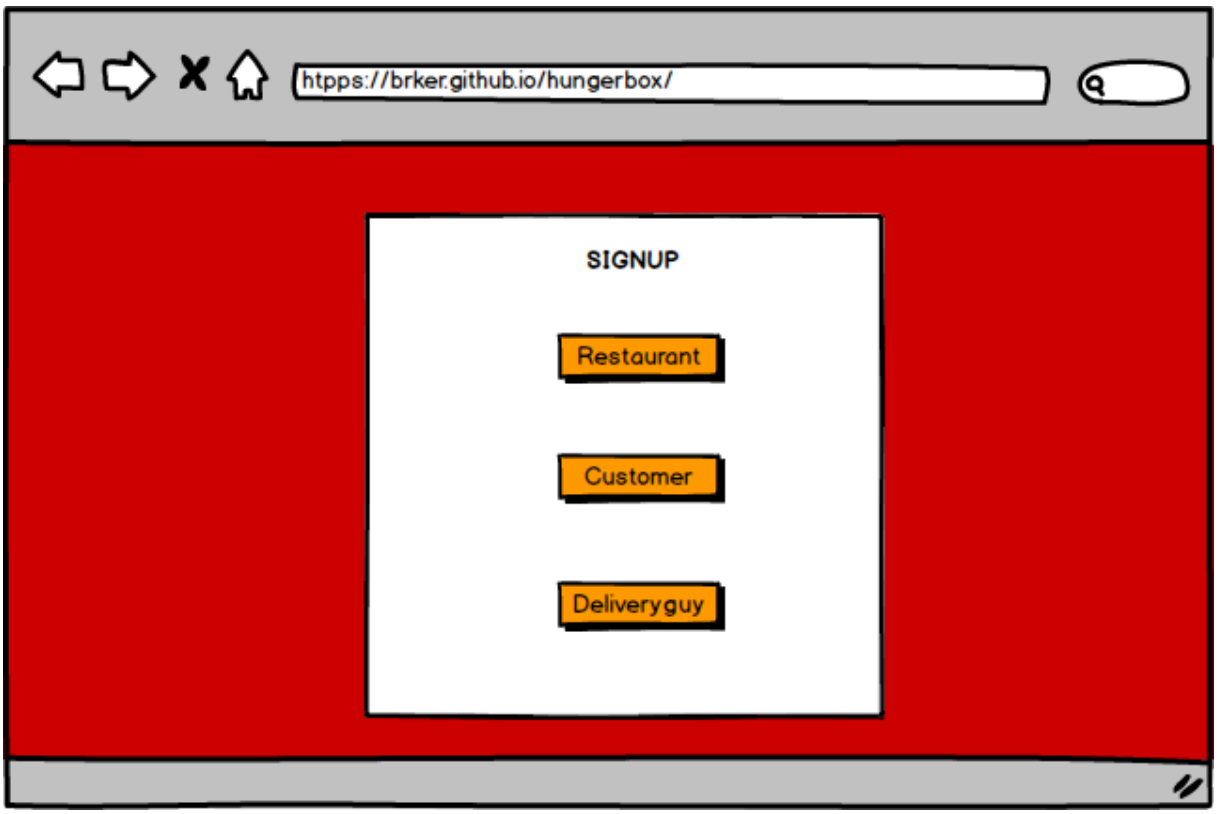
- Serve time cannot be more than 90 minutes.
- Price of a product cannot be negative.
- Vehicle cannot be used if it is under repair.

# ***6. Implementation Plan***

We will use MySQL as database implementation. For web application, we will use HTML, CSS and PHP.

# *7. User Interface Design and Corresponding SQL Statements*

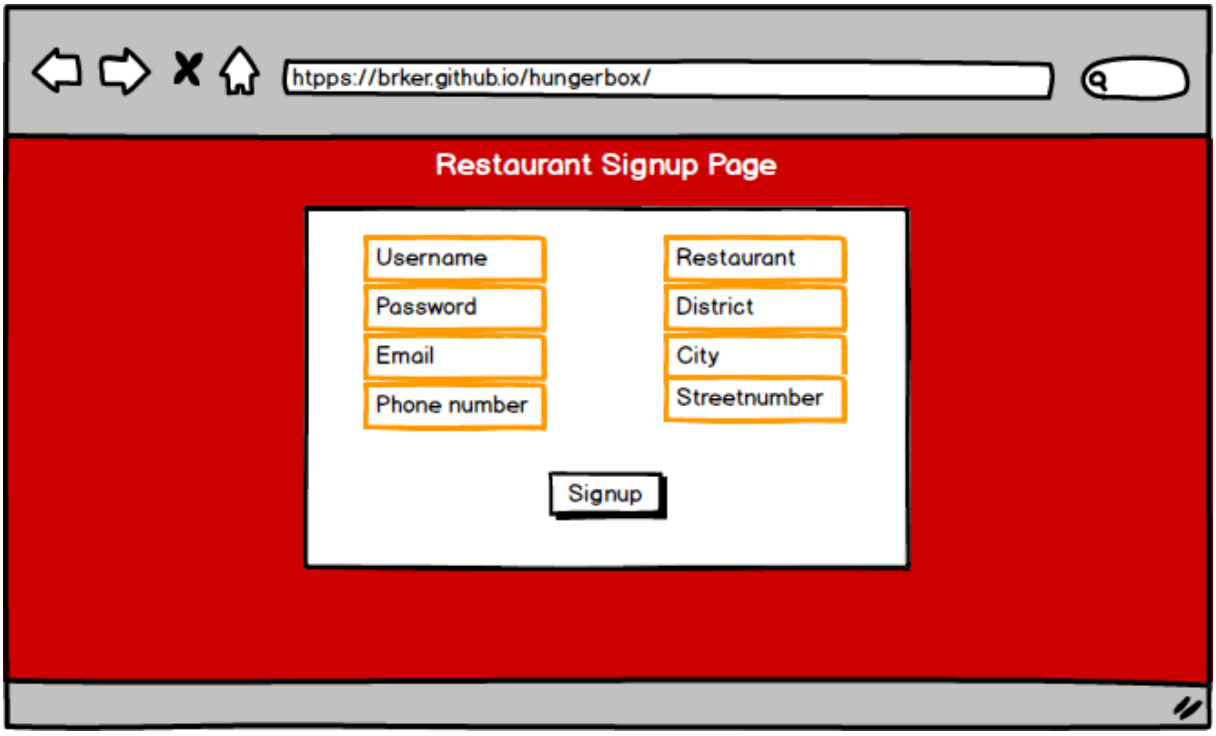
## 7.1. Signup




**Process:**

Our web application will have three different user types which are restaurant, customer, and delivery staff. Moreover, each user type needs to give different information to register into our system. Therefore, we will ask them what type of account they want to create before demanding required information.

### 7.1.1 Signup as a restaurant




**Process:**

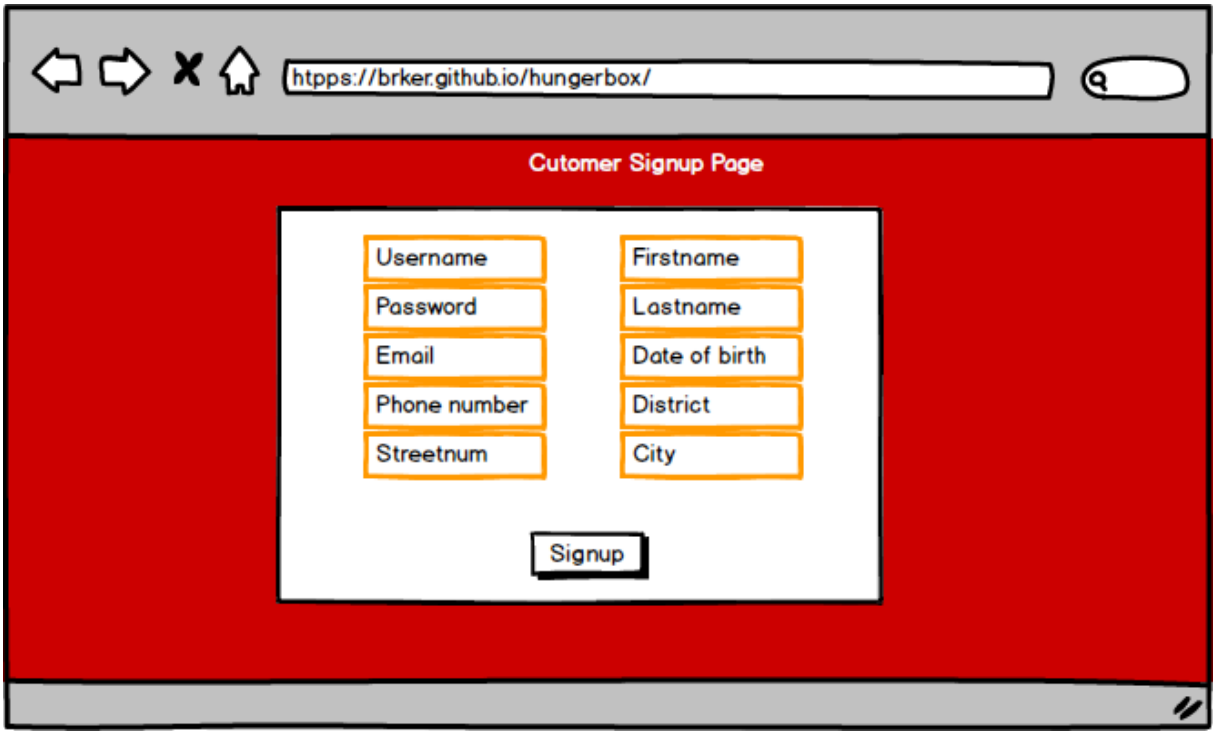
If restaurant user type is selected, user will see this signup page to register. (S)he must enter username, password, e-mail, phone number, restaurant name, district, street-number and the city as input. When sign up is clicked, a new user will be created and saved into database system.

**Inputs:** @username, @password, @email, @phonenumber, @resName, @district, @city, @streetnumber

**SQL Statements:**

```
INSERT INTO User(User-Name, Password, Email, District, Street_num, City)
SELECT @username, @password, @email, @district, @city, @streetnumber
WHERE NOT EXIST ( SELECT User-Name
                  FROM User
                  WHERE User-Name = @username)
INSERT INTO Restaurant(Restaurant-Name)
SELECT @resName
WHERE NOT EXIST ( SELECT Restaurant-Name
                  FROM Restaurant
                  WHERE Restaurant-Name = @resName)
```

### 7.1.2 Signup as a customer




**Process:**

If customer user type is selected, user will see this signup page to register. (S)he must enter username, password, e-mail, phone number, first name, last name, date of birth, district, street-number, and the city as input. When sign up is clicked, a new user will be created and saved into database system.

**Inputs:** @username, @password, @email, @phonenumber, @firstName, @lastName, @dateOfBirth, @district, @city, @streetnumber

**SQL Statements:**

```
INSERT INTO User(User-Name, Password, Email, District, Street_num, City)

SELECT @username, @password, @email, @district, @city, @streetnumber

WHERE NOT EXIST ( SELECT User-Name

                  FROM User

                  WHERE User-Name = @username )
```

INSERT INTO Customer(First_Name, Last_Name, Date_of_birth)

SELECT @firstName, @lastName, @dateOfBirth

### 7.1.3 Signup as a delivery staff


**Process:**

If customer user type is selected, user will see this signup page to register. (S)he must enter username, password, e-mail, phone number, first name, last name, city and driver license id as input. When sign up is clicked, a new user will be created and saved into database system.

**Inputs:** @username, @password, @email, @phonenumber, @firstName, @lastName, @city, @diriverLicId

**SQL Statements:**

INSERT INTO User(User-Name, Password, Email, District, City)

SELECT @username, @password, @email, @district, @city

WHERE NOT EXIST ( SELECT User-Name

FROM User

WHERE User-Name = @username)

```
INSERT INTO Delivery_Staff (First_Name, Last_Name, Date_of_birth, Driver_License_Id)
SELECT @firstName, @lastName, @dateOfBirth, @driverLicId
WHERE NOT EXIST ( SELECT Driver_License_Id
                  FROM Delivery_Staff
                  WHERE Driver_License_Id = @driverLicId)
```

### 7.2 Login


**Process:**

If users have already registered, they can login the system by entering their usernames and passwords correctly. System will check existence of entered username and password combination by using database system.

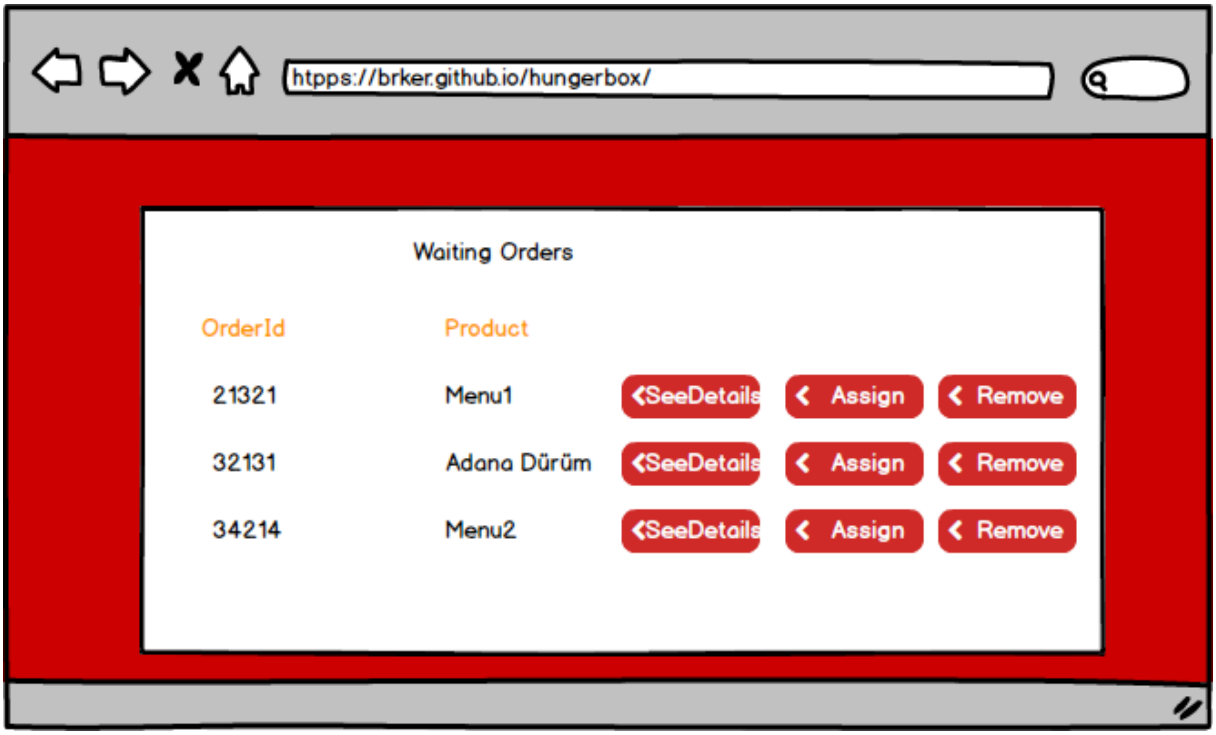
**Inputs:** @username, @password

**SQL Statements:**

```
SELECT User_Name, Password
FROM User
Where User_Name = @username AND Password = @password
```

### 7.3 Waiting orders to deliver for restaurant type users

#### 7.3.1 Main page




**Process:**

If users signed as a restaurant, they can visualize waiting orders to deliver. Moreover, they are able to see details of the order, assign a delivery staff for the order and reject the order. In addition to than when user clicked assign button (S)he will be directed a new page to select delivery staff.

**Inputs: @orderId**

**SQL Statements:**

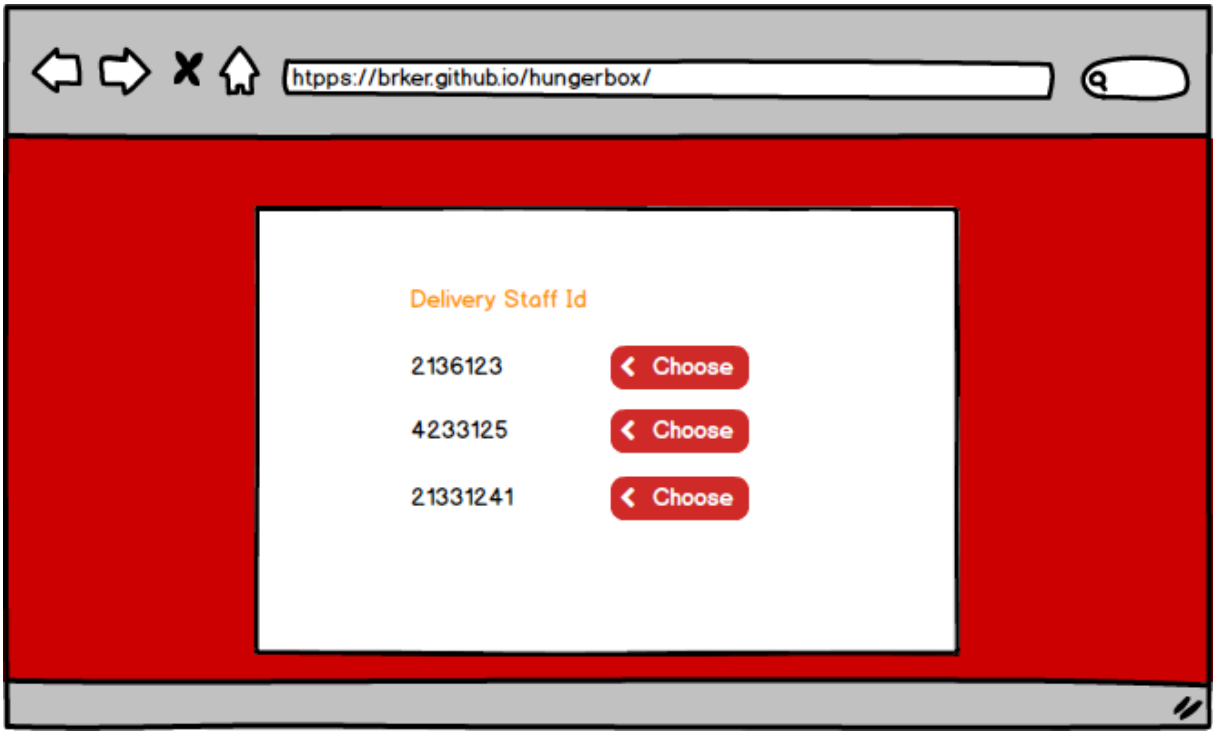
**See Details:**

```
SELECT Price, Payment-type, Comment
FROM Order
WHERE Order_Id = @orderId
```

**Assign:**

```
SELECT ID
FROM Delivery_Staff
```

### 7.3.2 Assign page




**Process:**

In this page, users are able to assign the order for specific delivery staff by clicking one of the choose buttons.

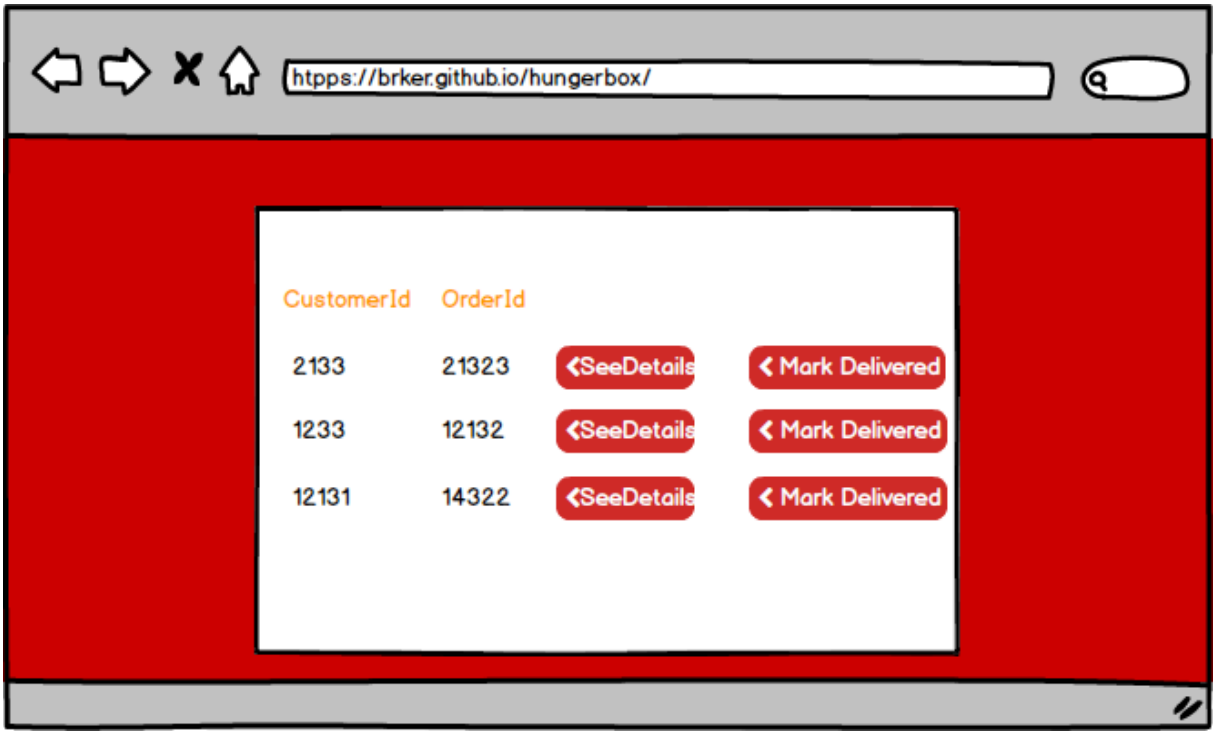
**Inputs:** @deliveryStaffId, @orderId

**SQL Statements:**

INSERT INTO Delivers (Driver_License_Id, Order_Id)

SELECT @deliveryStaffId, @orderId

### 7.4 Delivery Staff Main Page




**Process:**

In this page, Delivery Staff users can see details of orders and address of customer by clicking see details button and mark them delivered when they complete delivery job by clicking mark delivered button.

**Inputs: @orderId, @customerId, @deliveryStaffId**

**SQL Statements:**

**See Details:**

SELECT *

FROM Orders

WHERE @orderId = Order_Id

SELECT District, Street_name, Street_num

FROM Users

WHERE @customerId = Id

**Mark Delivered**

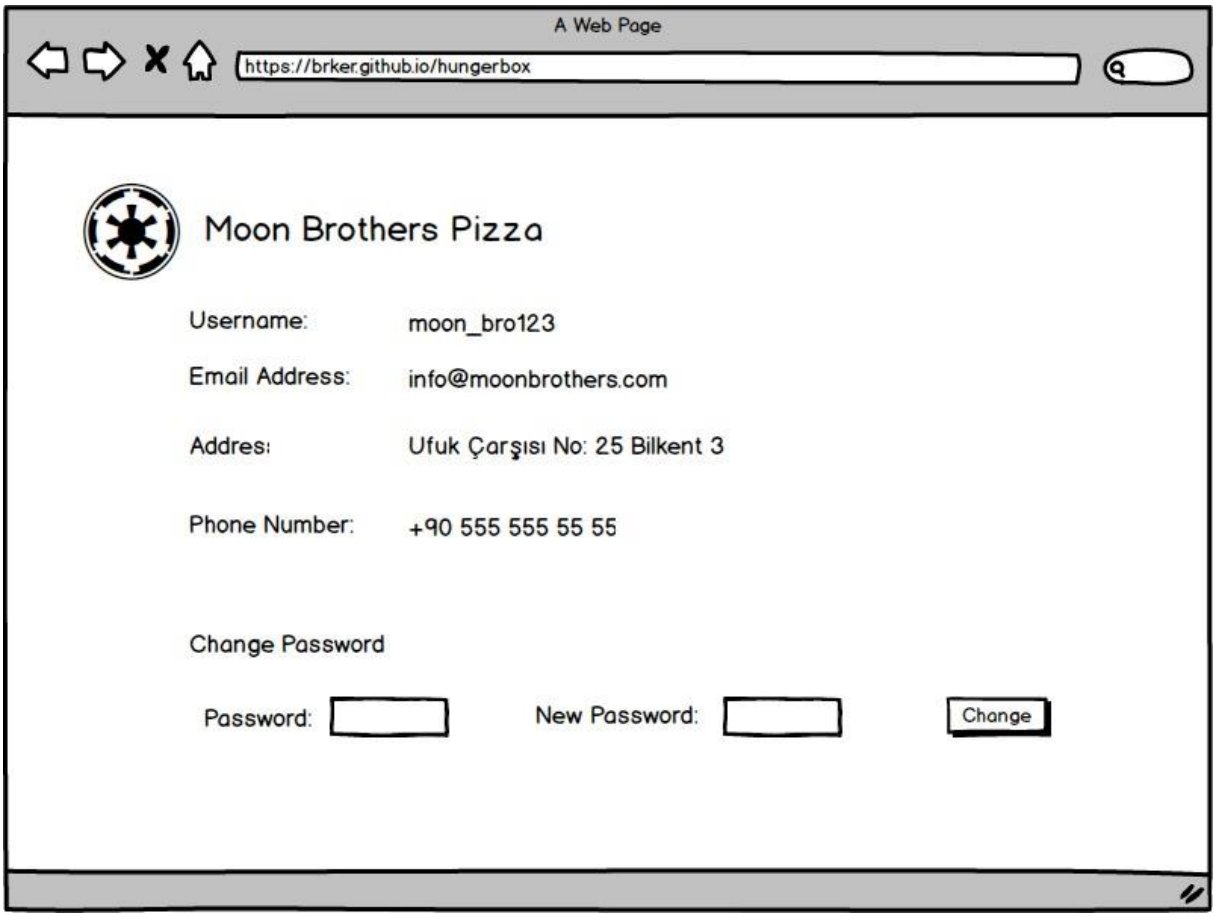
DELETE

FROM Delivers

WHERE @orderId = Order_Id and @deliveryStaffId = @Driver_License_Id

### 7.5 User Information Pages

**Process:** Process: Restaurants, customers and delivery stuff can check their information(Username, Email, Address etc.) on their pages and change their passwords.

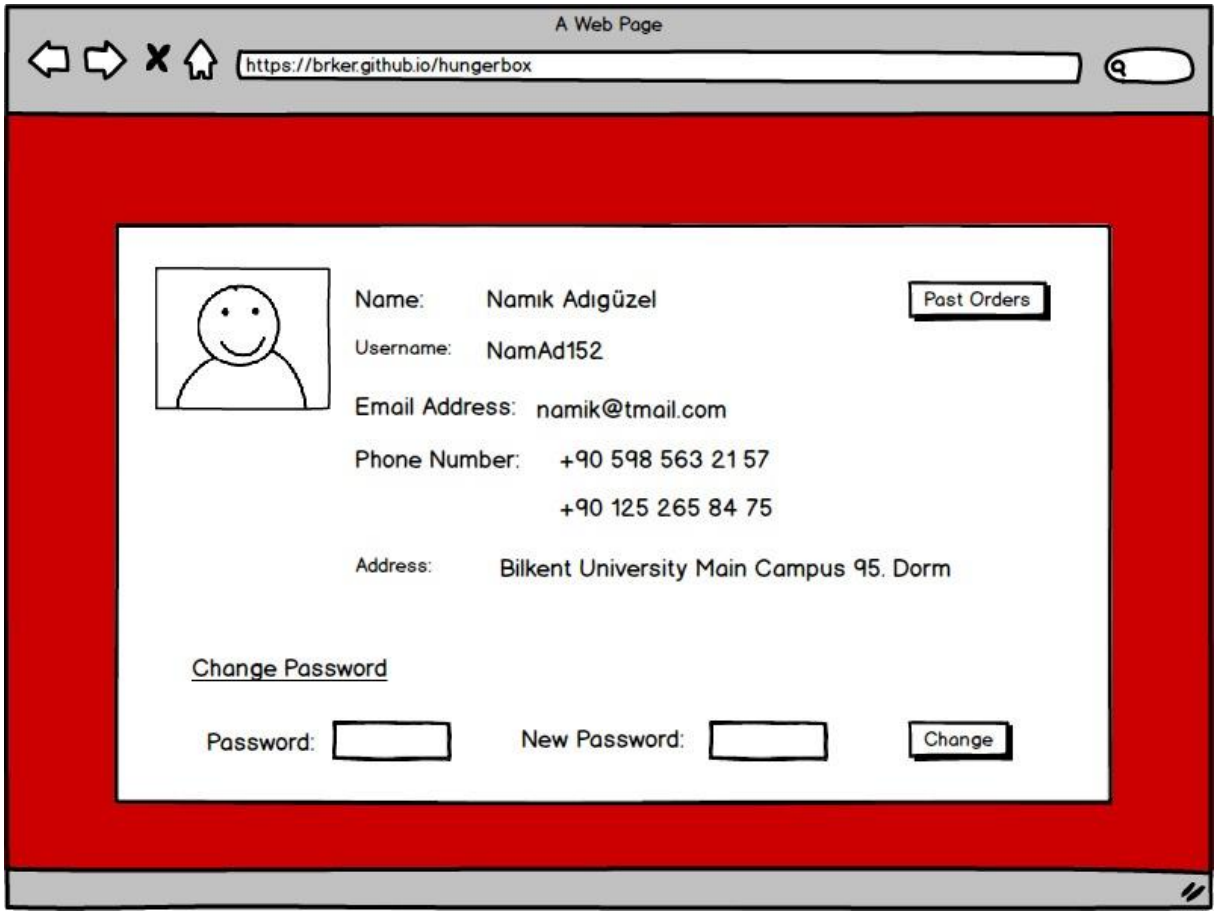



**(Restaurant Information Page)**

**SQL Statements:**
Inputs: @username, @password, @newPassword

```
SELECT U.email, U.address, P.Phone
FROM User U, Phone P
WHERE U.Username = @username AND P.ID = U.ID

UPDATE User
SET Password = @newPassword
WHERE User_Name = @username AND Password = @password
```

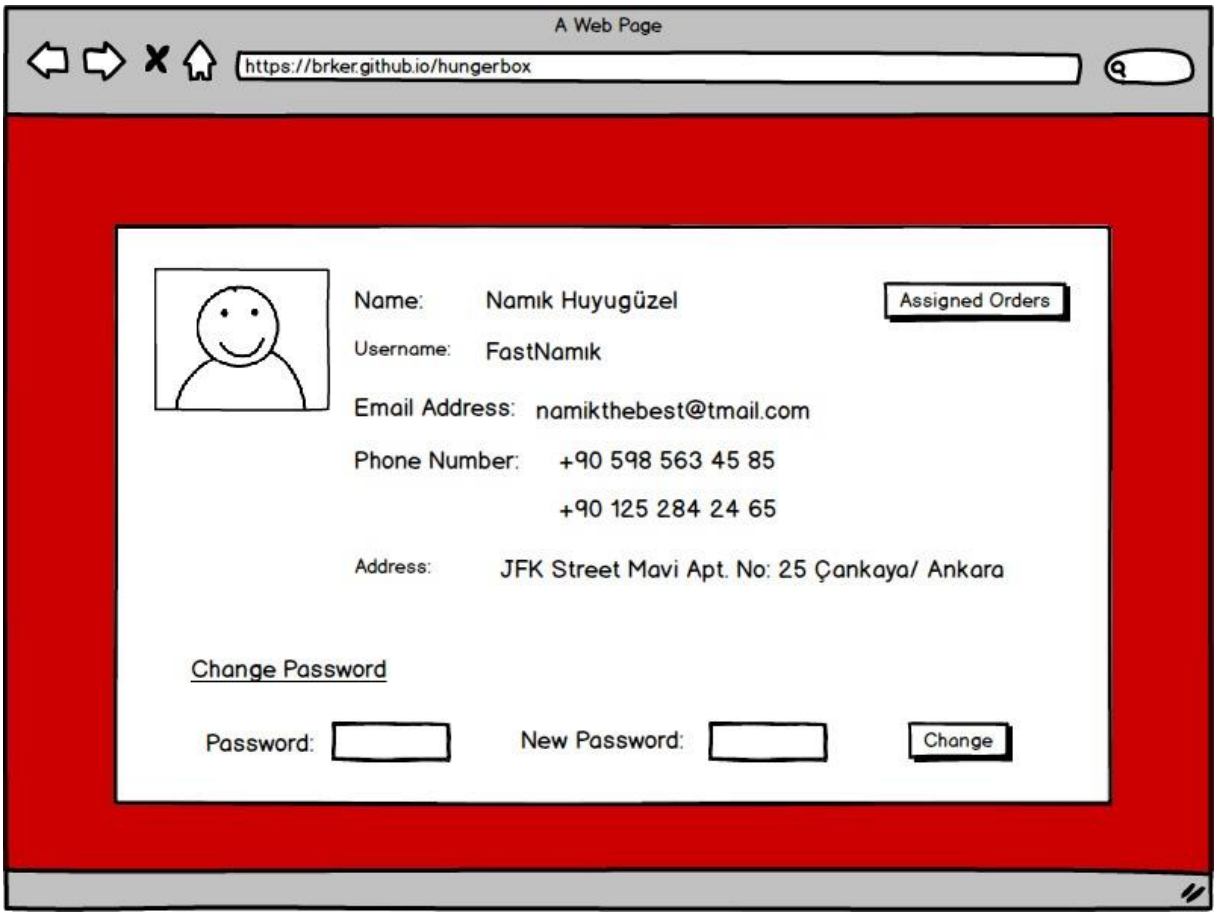



**(Customer Information Page)**

**SQL Statements:**
Inputs: @username, @password, @newPassword

```
SELECT C.first_name, C.last_name, C.email, C.address, P.Phone
FROM Customer C, Phone P
WHERE C.Username = @username AND P.ID = U.ID

UPDATE User
SET Password = @newPassword
WHERE User_Name = @username AND Password = @password
```

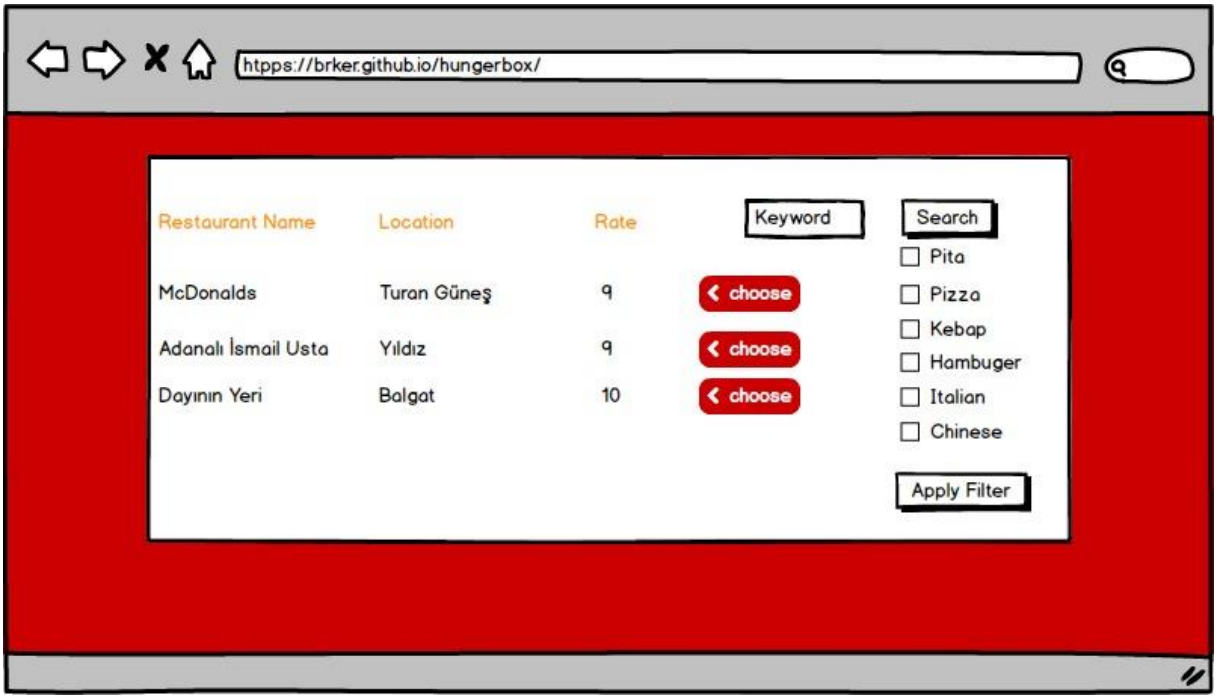



**(DeliveryStuff Information Page)**

**SQL Statements:**
Inputs: @username, @password, @newPassword

```
SELECT DS.first_name, DS.last_name, DS.email, DS.address, P.Phone
FROM Delivery_Stuff DS, Phone P
WHERE DS.Username = @username AND P.ID = U.ID

UPDATE User
SET Password = @newPassword
WHERE User_Name = @username AND Password = @password
```




**(Listing Restaurants)**

**Process:** Customers can list the restaurants that they want to order from according to their types.

SQL Statements:

Input: @type

```
SELECT restaurant_name, address, rating
FROM Restaurant
WHERE r_type = @type
```