Department of Computer Engineering

CS353 - Database Systems

Spring 2021

Term Project Design Report

Food Ordering and Delivery System: e-Meal

Group 18

Teaching Assistant:

Arif Usta

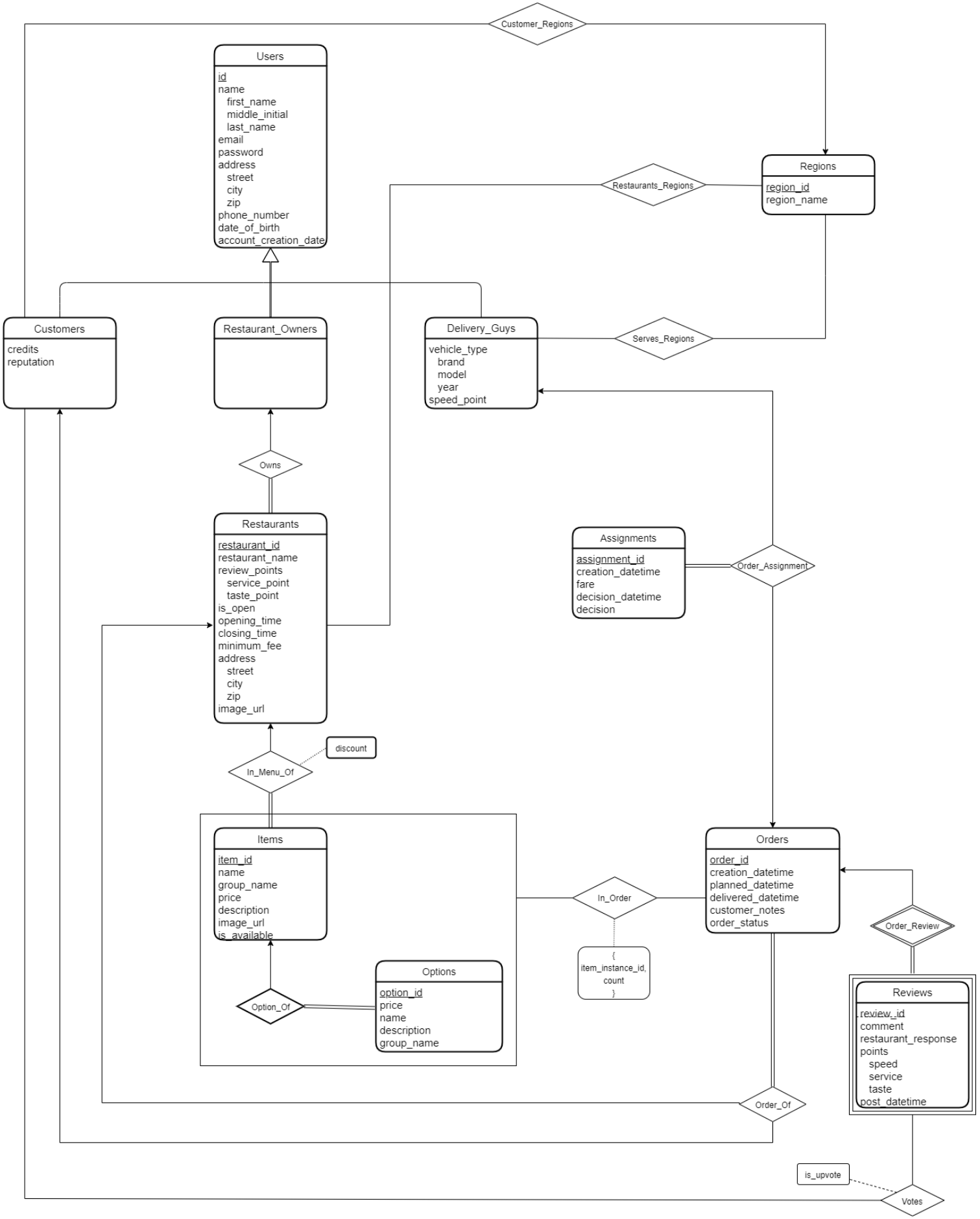
Group Members:

Melisa Taşpınar	21803668	Section 2
Yiğit Gürses	21702746	Section 1
Mustafa Hakan Kara	21703317	Section 1
Furkan Demir	21802818	Section 3

Table of Contents

1 Revised E/R Diagram

Below is our E/R diagram that we revised based on the feedback we received on our proposal.



The important changes we made are as follows.

- ➢ Specializations for Delivery_Guys (Rider, Driver) were removed.
- ➢ Attribute ‘available’ of Delivery_Guys was removed, it is now calculated with queries every time it is needed.
- ➢ Attribute ‘count_of_order’ was removed from Items, it is now calculated with queries every time it is needed.
- ➢ Regions system was added. This encapsulates 3 new relations and an entity called Regions.
- ➢ Customers, Restaurants and Delivery_Guys now have relations with Regions. These are used to determine which Assignments to give to Delivery_Guys, and which Restaurants to show to Customers.
- ➢ Reviews and Ratings were combined into Reviews to fix the ternary relationship with two weak entities.
- ➢ A new ‘Additional Requirement’ was added: Customers now have a reputation calculated as “C*(total_upvotes-total_downvotes)” where C is some arbitrary positive real number. Reputation of Customers will be shown next to their reviews to make them more/less credible. High enough reputation will also unlock special offers and discounts.
- ➢ For the reason given in the previous bullet, Customers now have a relation with Reviews. This Votes relation keeps track of which customer upvoted/downvoted which review.
- ➢ Assignment was made a strong entity to fix the ternary relation. Also its attributes were reconsidered to keep each decision of Delivery_Guys. Note that each assignment is either accepted or rejected and then stored. If an Assignment is rejected, a new Assignment is generated by the system that will be related to the same Order but a different Delivery_Guy.
- ➢ Item_Instances entity was removed and got replaced by an aggregation. In_Order now holds an attribute called ‘item_instance_ID’ which denotes different items (they can have the same item_id but their selected options are different) in an order.
- ➢ Basket entity was removed and now Order is directly connected to Customers and Restaurants.
- ➢ Advertisement entity was removed.
- ➢ Items entity now has total participation in the relation In_Menu_Of.
- ➢ Attribute ‘order_status’ of Orders is now an enum encapsulating the different stages an order can be in (being reviewed, rejected, being prepared, on the way, delivered). This allowed us to get rid of redundant attributes (isConfirmed, isDelivered).
- ➢ A new attribute called ‘group_name’ was added to the entities Items and Options. This attribute will be used to group Items/Options that have the same ‘group_name’ in the GUI. For example burgers, pizzas and beverages could be the different group names of a restaurant’s menu.

2 Set of Relations

Users(id, first_name, middle_name, last_name, email, password, street, city, zip INT, phone_number, date_of_birth, account_creation_date)

- Candidate Keys: id, email
- Primary Key: id
- Foreign Keys: -
- Table Declaration: CREATE TABLE Users(id INT, first_name VARCHAR(255), middle_name VARCHAR(255), last_name VARCHAR(255), email VARCHAR(255), password VARCHAR(255), street VARCHAR(255), city VARCHAR(255), zip INT, phone_number VARCHAR(255), date_of_birth DATE, account_creation_date DATE, PRIMARY KEY(id));

Customers(id, credits)

- Candidate Keys: id
- Primary Key: id
- Foreign Keys: id references Users(id)
- Table Declaration: CREATE TABLE Customers(id INT, credits DECIMAL(8, 2), PRIMARY KEY(id), FOREIGN KEY (id) REFERENCES Users(id) ON DELETE CASCADE);

Restaurant_Owners(id)

- Candidate Keys: id
- Primary Key: id
- Foreign Keys: id references Users(id)
- CREATE TABLE Restaurant_Owners(id INT, PRIMARY KEY(id), FOREIGN KEY (id) REFERENCES Users(id) ON DELETE CASCADE);

Delivery_Guys(id, vehicle_brand, vehicle_model, vehicle_year, speed_point)

- Candidate Keys: id
- Primary Key: id
- Foreign Keys: id references Users(id)

- Table Declaration: CREATE TABLE Delivery_Guys(id INT, vehicle_brand VARCHAR(255), vehicle_model VARCHAR(255), vehicle_year VARCHAR(255), speed_point FLOAT, PRIMARY KEY(id), FOREIGN KEY (id) REFERENCES Users(id) ON DELETE CASCADE);

Restaurants(restaurant_id, restaurant_name, owner_id, service_point, taste_point, is_open, opening_time, closing_time, minimum_fee, street, city, zip, image_url)

- Candidate Keys: restaurant_id, owner_id
- Primary Key: restaurant_id
- Foreign Keys: owner_id references Restaurant_Owners(id)
- Table Declaration: CREATE TABLE Restaurants(restaurant_id INT, restaurant_name VARCHAR(255), owner_id INT NOT NULL, service_point FLOAT, taste_point FLOAT, is_open TINYINT(1), opening_time TIME, closing_time TIME, minumum_fee DECIMAL(6, 2), street VARCHAR(255), city VARCHAR(255), zip INT, image_url VARCHAR(255), PRIMARY KEY(restaurant_id), FOREIGN KEY (owner_id) REFERENCES Restaurant_Owners(id) ON DELETE CASCADE, check (minumum_fee >= 0));

Orders(order_id, restaurant_id, customer_id, creation_datetime, planned_datetime, delivered_datetime, customer_notes, order_status)

- Candidate Keys: order_id
- Primary Key: order_id
- Foreign Keys: restaurant_id references Restaurants(restaurant_id), customer_id references Customers(id)
- Table Declaration: CREATE TABLE Orders(order_id INT, restaurant_id INT NOT NULL, customer_id INT NOT NULL, creation_datetime DATETIME, planned_datetime DATETIME, delivered_datetime DATETIME, customer_notes VARCHAR(255), order_status ENUM('being reviewed', 'rejected', 'being prepared', 'cancelled', 'on the way', 'delivered') NOT NULL, PRIMARY KEY(order_id), FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON DELETE CASCADE , FOREIGN KEY (customer_id) REFERENCES Customers(id) ON DELETE CASCADE);

Reviews(order_id, review_id, comment, restaurant_response, speed_point, service_point, taste_point, post_datetime, upvotes, downvotes)

- Candidate Keys: (order_id, review_id)
- Primary Key: (order_id, review_id)
- Foreign Keys: order_id references Orders(order_id)
- Table Declaration: CREATE TABLE Reviews(order_id INT, review_id INT, comment VARCHAR(255), restaurant_response VARCHAR(255), speed_point FLOAT, service_point FLOAT, taste_point FLOAT, post_datetime DATETIME, upvotes INT, downvotes INT, PRIMARY KEY(order_id, review_id), FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE);

Items(item_id, restaurant_id, name, group_name, price, discount, description, image_url, is_available)

- Candidate Key: item_id
- Primary Key: item_id
- Foreign Keys: restaurant_id references Restaurants(restaurant_id)
- Table Declaration: CREATE TABLE Items(item_id INT, restaurant_id INT NOT NULL, name VARCHAR(255), group_name VARCHAR(255), price DECIMAL(6, 2), discount FLOAT, description VARCHAR(255), image_url VARCHAR(255), is_available TINYINT(1), PRIMARY KEY(item_id), FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON DELETE CASCADE, check (price > 0 AND discount < 1 AND discount >= 0));

Options(option_id, item_id, group_name, name, description, price)

- Candidate Keys: option_id, (item_id, group_name, name, description, price)
- Primary Key: option_id
- Foreign Keys: item_id references Items(item_id)
- Table Declaration: CREATE TABLE Options(option_id INT, item_id INT NOT NULL, group_name VARCHAR(255), name VARCHAR(255), description VARCHAR(255), price DECIMAL(6, 2), PRIMARY KEY(option_id), FOREIGN KEY (item_id) REFERENCES Items(item_id) ON DELETE CASCADE, check (price >= 0));

In_Order(item_id, option_id, item_instance_id, order_id, count)

- Candidate Keys: (item_id, option_id, item_instance_id, order_id)
- Primary Key: (item_id, option_id, item_instance_id, order_id)
- Foreign Keys: item_id references Items(item_id), option_id references Options(option_id), order_id references Orders(order_id)
- Table Declaration: CREATE TABLE In_Order(item_id INT, option_id INT, item_instance_id INT, order_id INT, count INT, PRIMARY KEY(item_id, option_id, item_instance_id, order_id), FOREIGN KEY (item_id) REFERENCES Items(item_id) ON DELETE CASCADE, FOREIGN KEY (option_id) REFERENCES Options(option_id) ON DELETE CASCADE, FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE);

Regions(region_id, region_name)

- Candidate Keys: region_id
- Primary Key: region_id
- Foreign Keys: -
- Table Declaration: CREATE TABLE Regions(region_id INT, region_name VARCHAR(255), PRIMARY KEY(region_id));

Restaurant_Regions(restaurant_id, region_id)

- Candidate Keys: (restaurant_id, region_id)
- Primary Key: (restaurant_id, region_id)
- Foreign Keys: restaurant_id references Restaurants(restaurant_id), region_id references Regions(region_id)
- Table Declaration: CREATE TABLE Restaurant_Regions(restaurant_id INT, region_id INT, PRIMARY KEY(restaurant_id, region_id), FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON DELETE CASCADE, FOREIGN KEY (region_id) REFERENCES Regions(region_id) ON DELETE CASCADE);

Serves_Regions(delivery_guy_id, region_id)

- Candidate Keys: (delivery_guy_id, region_id)
- Primary Key: (delivery_guy_id, region_id)

- Foreign Keys: delivery_guy_id references Delivery_Guys(id), region_id references Regions(region_id)
- Table Declaration: CREATE TABLE Serves_Regions(delivery_guy_id INT, region_id INT, PRIMARY KEY(delivery_guy_id, region_id), FOREIGN KEY (delivery_guy_id) REFERENCES Delivery_Guys(id) ON DELETE CASCADE, FOREIGN KEY (region_id) REFERENCES Regions(region_id) ON DELETE CASCADE);

Assignments(assignment_id, order_id, delivery_guy_id, creation_time, fare, decision_datetime, decision)

- Candidate Keys: assignment_id
- Primary Key: assignment_id
- Foreign Keys: order_id references Orders(order_id), delivery_guy_id references Delivery_Guys(id)
- Table Declaration: CREATE TABLE Assignments(assignment_id INT, order_id INT, delivery_guy_id INT, creation_time DATETIME, fare DECIMAL(6, 2), decision_datetime DATETIME, decision TINYINT(1), PRIMARY KEY(assignment_id), FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE, FOREIGN KEY (delivery_guy_id) REFERENCES Delivery_Guys(id) ON DELETE CASCADE, check(fare > 0));

Votes(order_id, review_id, customer_id, is_upvote)

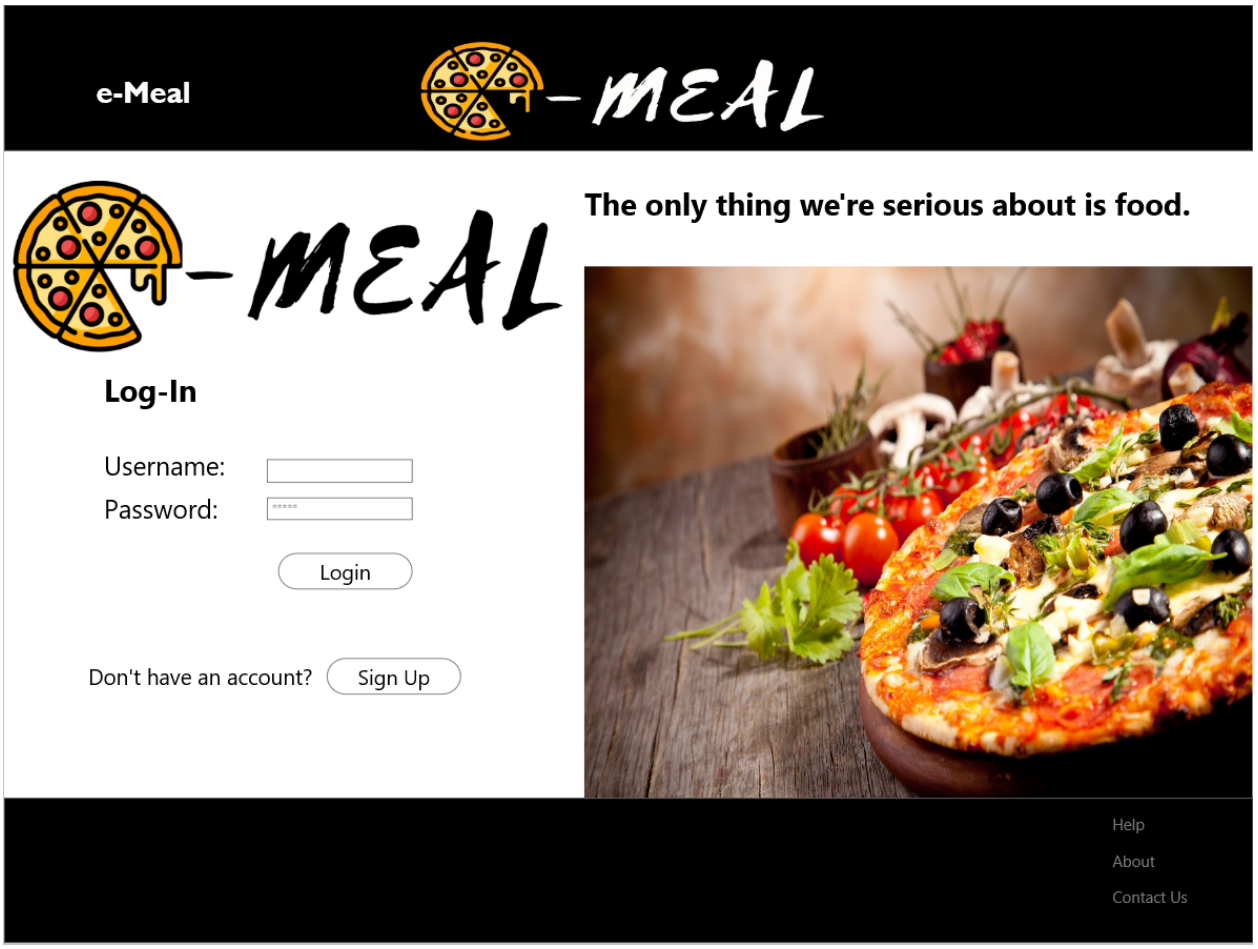
- Candidate Keys: (order_id, review_id, customer_id)
- Primary Key: (order_id, review_id, customer_id)
- Foreign Keys: (order_id, review_id) references Reviews(order_id, review_id), customer_id references Customers(id)
- Table Declaration: CREATE TABLE Votes(order_id INT, review_id INT, customer_id INT, is_upvote TINYINT(1), PRIMARY KEY (order_id, review_id, customer_id), FOREIGN KEY (order_id, review_id) REFERENCES Reviews(order_id, review_id) ON DELETE CASCADE , FOREIGN KEY (customer_id) REFERENCES Customers(id) ON DELETE CASCADE);

Customer_Regions(customer_id, region_id)

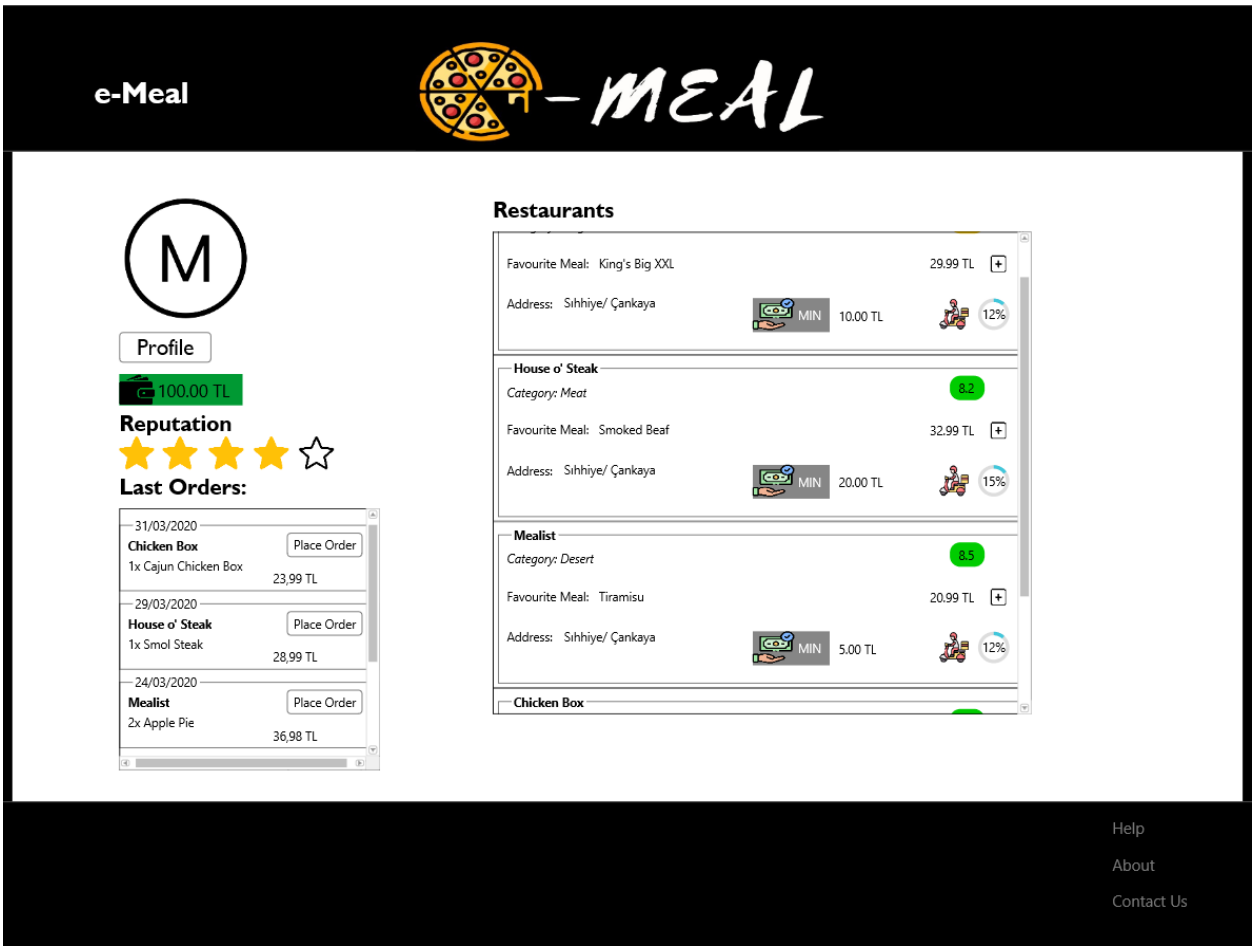
- Candidate Keys: customer_id
- Primary Key: customer_id
- Foreign Keys: customer_id references Customers(id), region_id references Regions(region_id)
- Table Declaration: CREATE TABLE Customer_Regions(customer_id INT, region_id INT, PRIMARY KEY(customer_id), FOREIGN KEY (customer_id) REFERENCES Customers(id) ON DELETE CASCADE, FOREIGN KEY (region_id) REFERENCES Regions(region_id) ON DELETE CASCADE);

3 Mockups

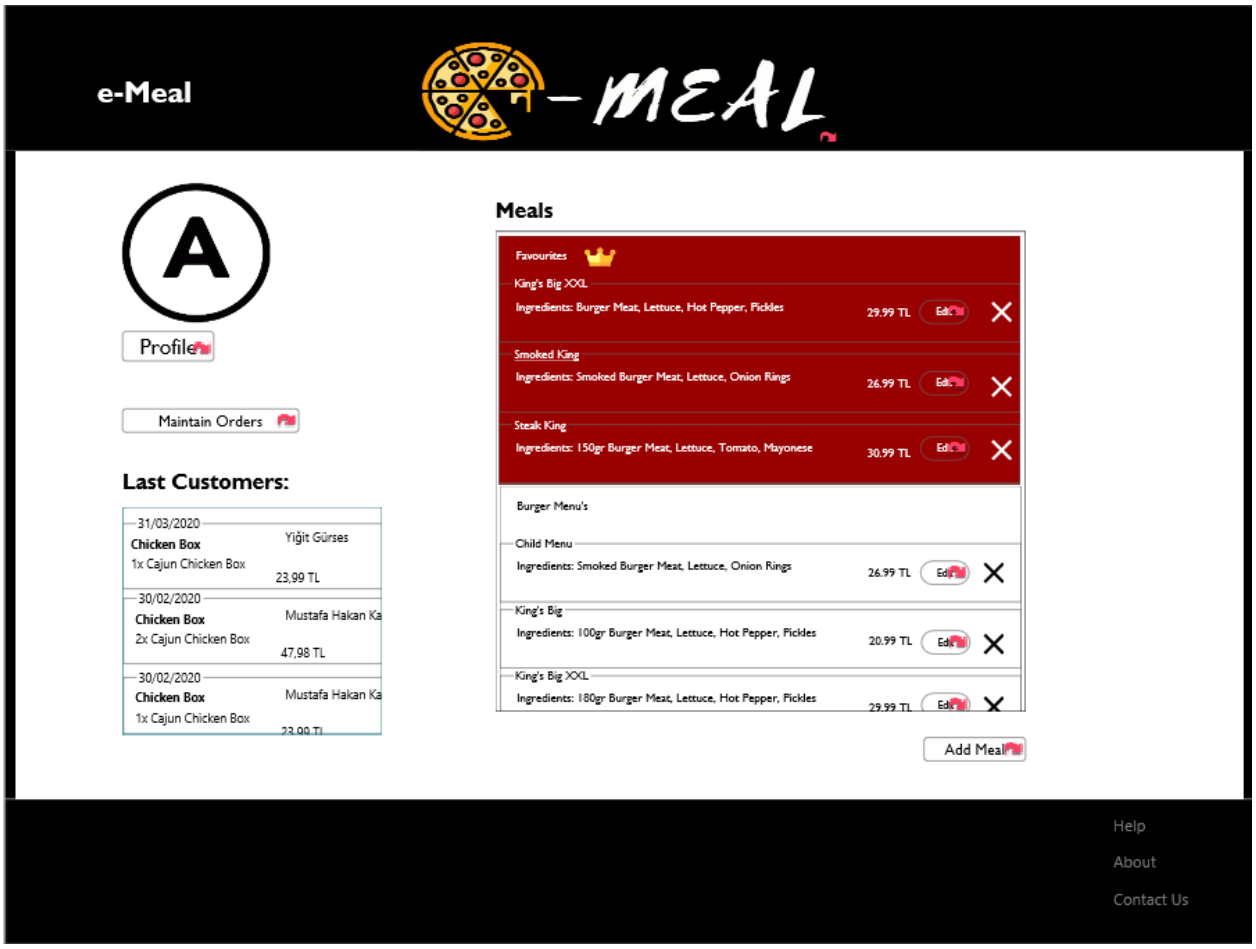
3.1 Login Page



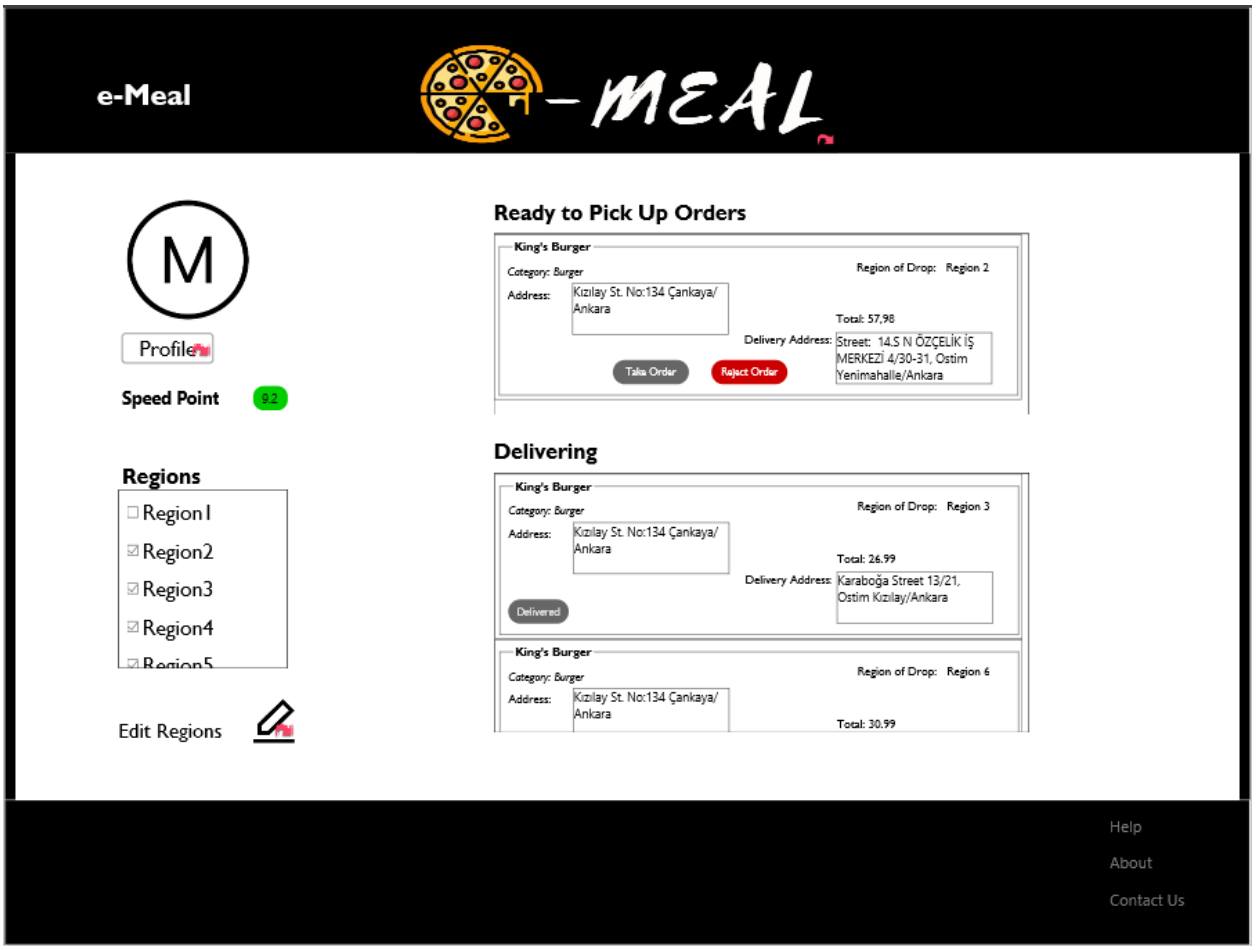
3.2 Customer’s Home Page



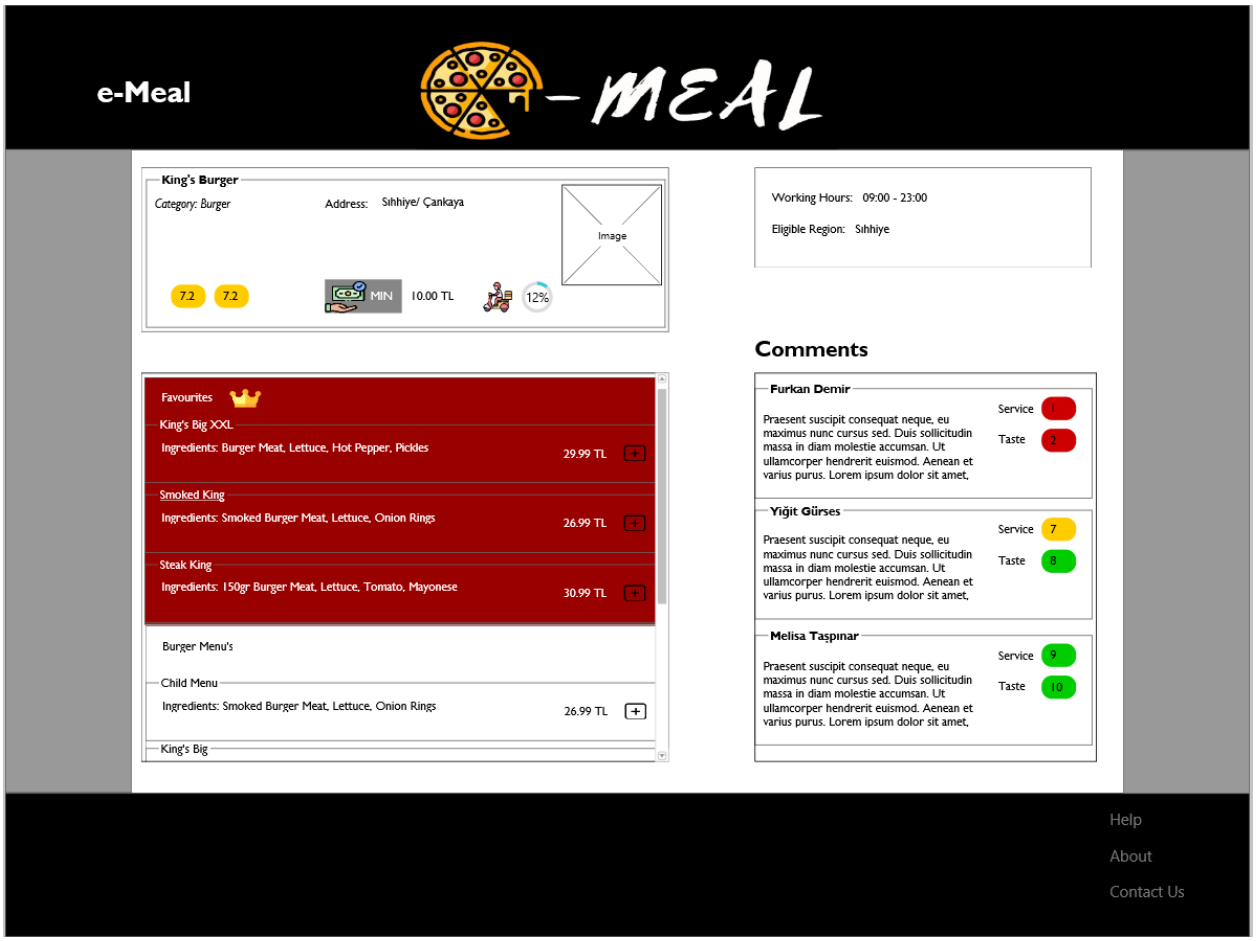
3.3 Restaurant Owner's Home Page



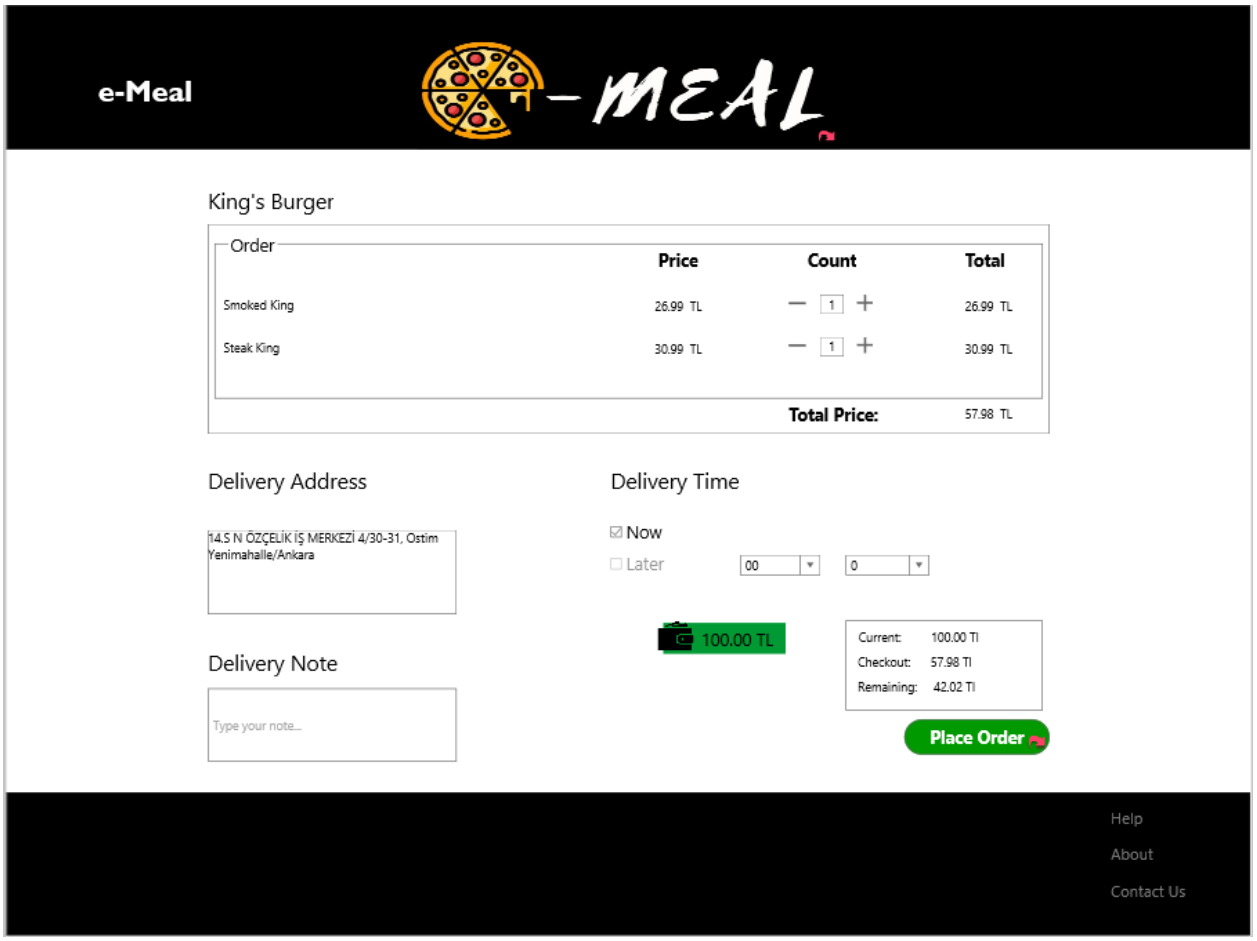
3.4 Delivery Guy's Home Page



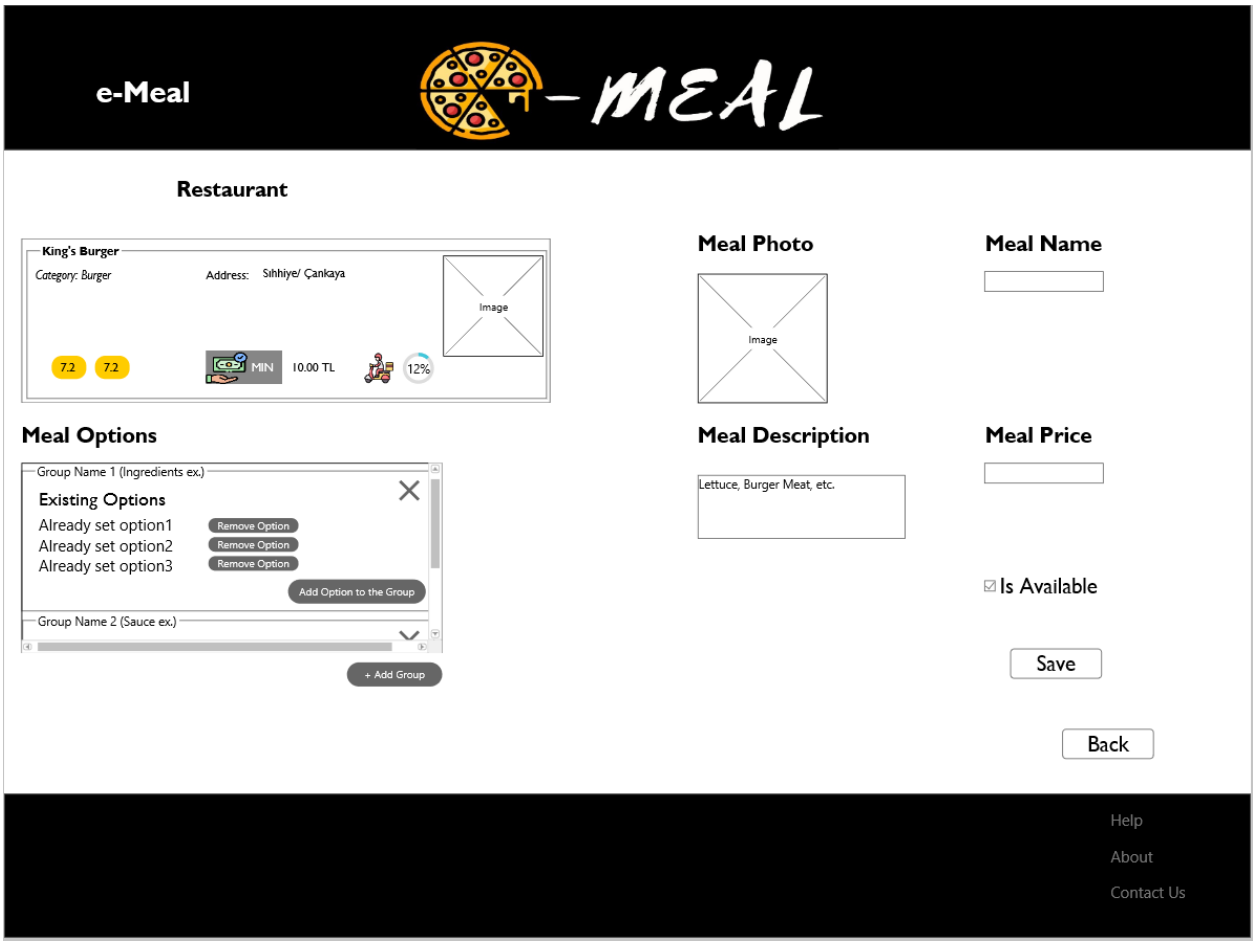
3.5 Customer's Restaurant Page



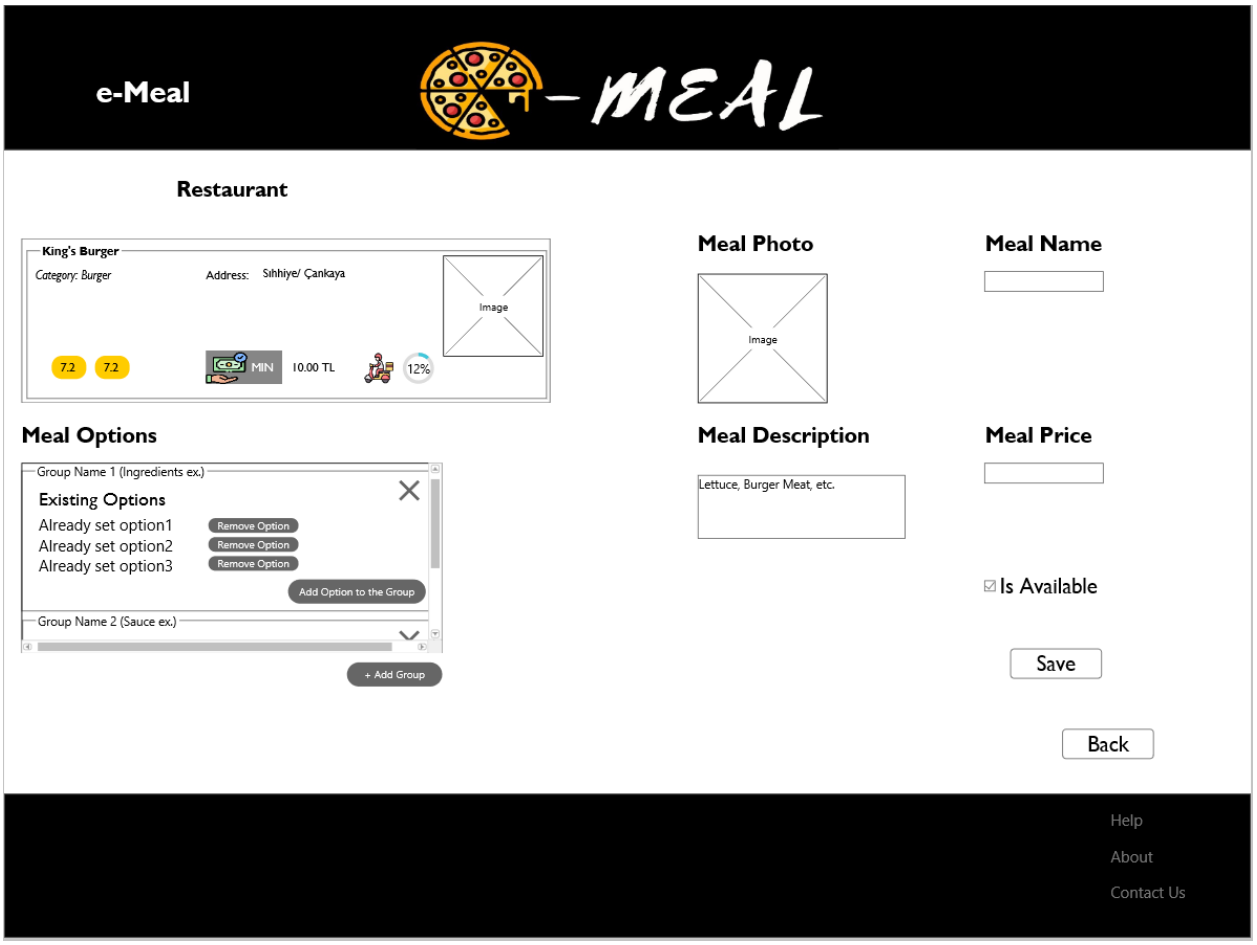
3.6 Customer’s Placing Order Page



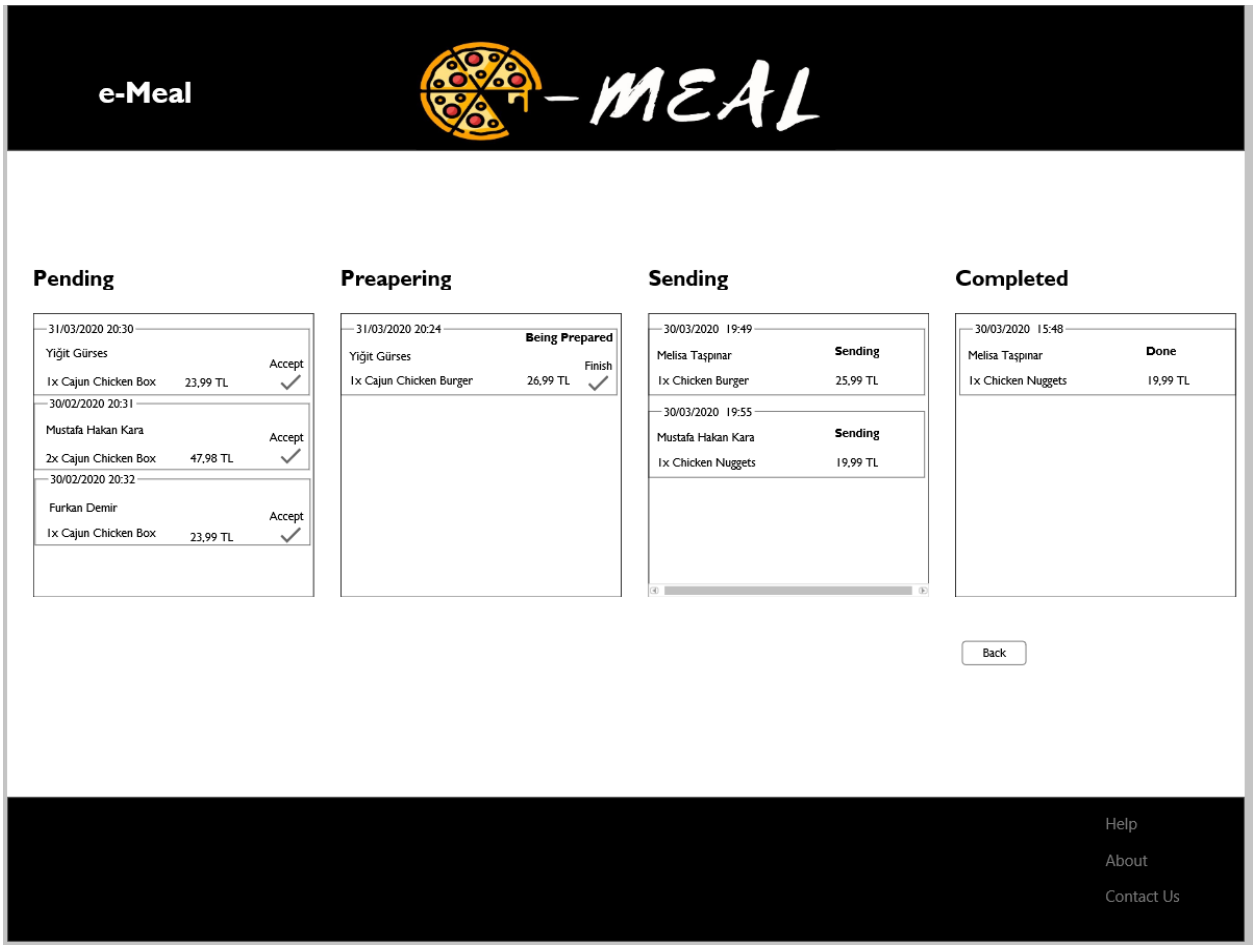
3.7 Restaurant Owner's Add Meal Page



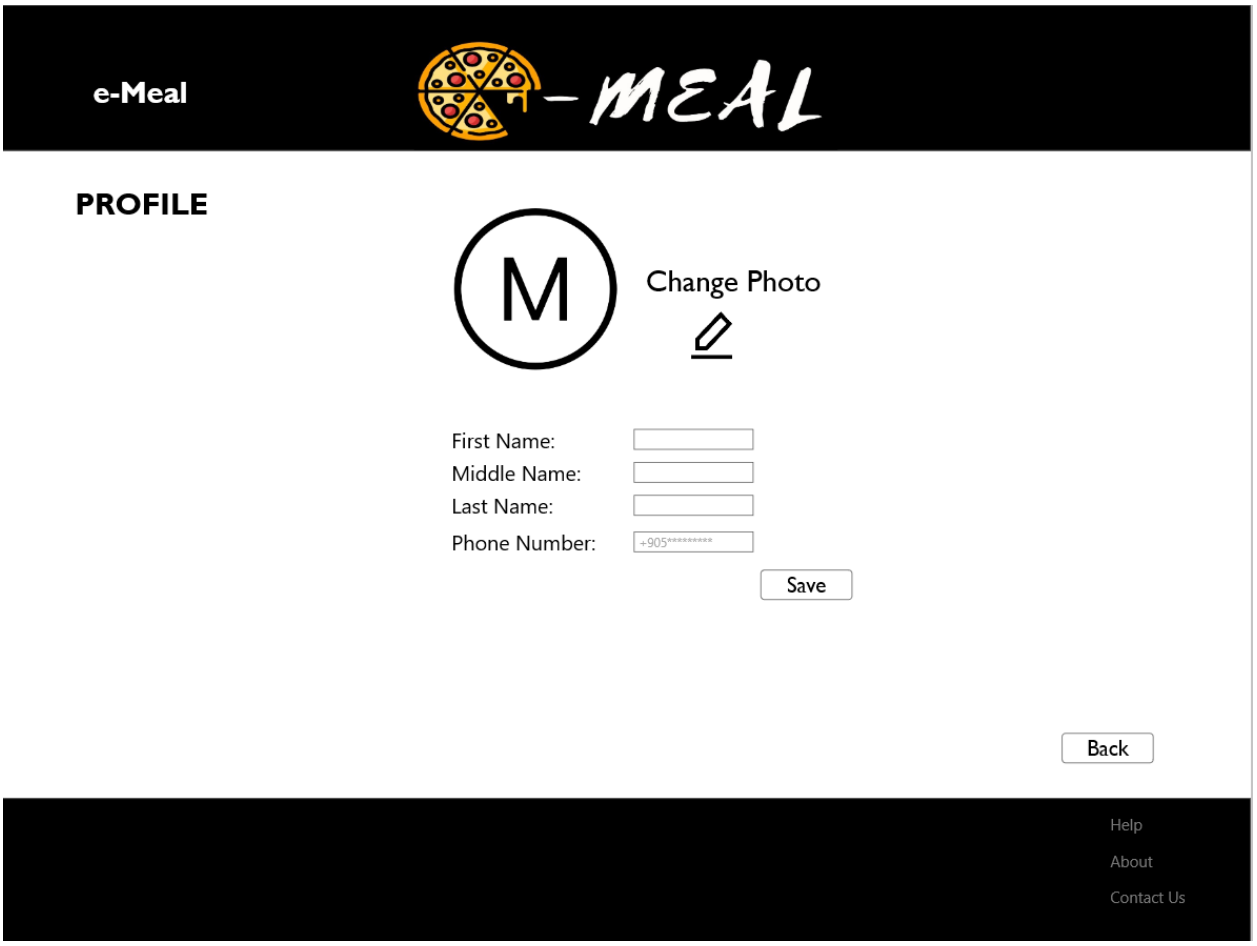
3.8 Restaurant Owner’s Edit Meal Page



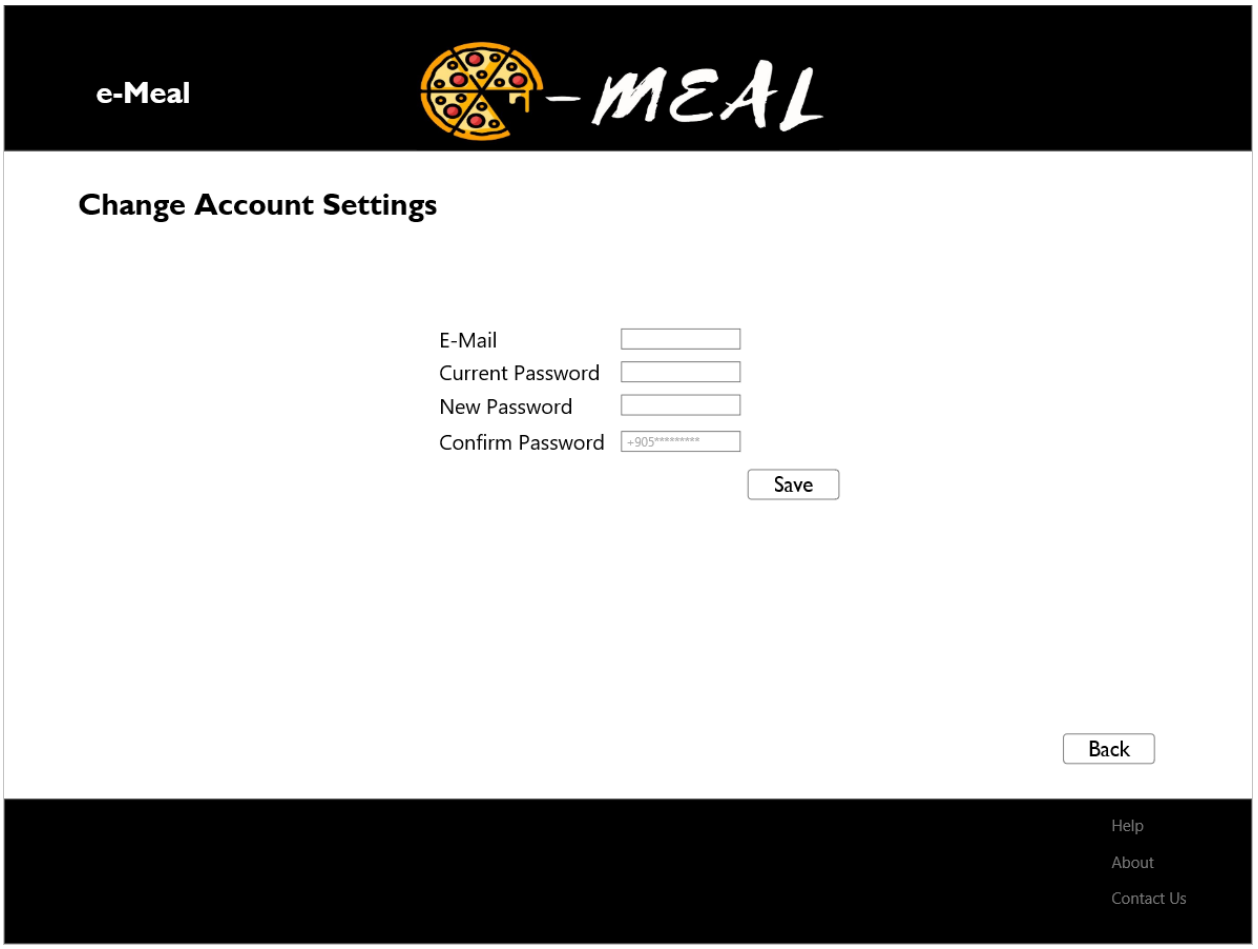
3.9 Restaurant Owner's Maintain Orders Page



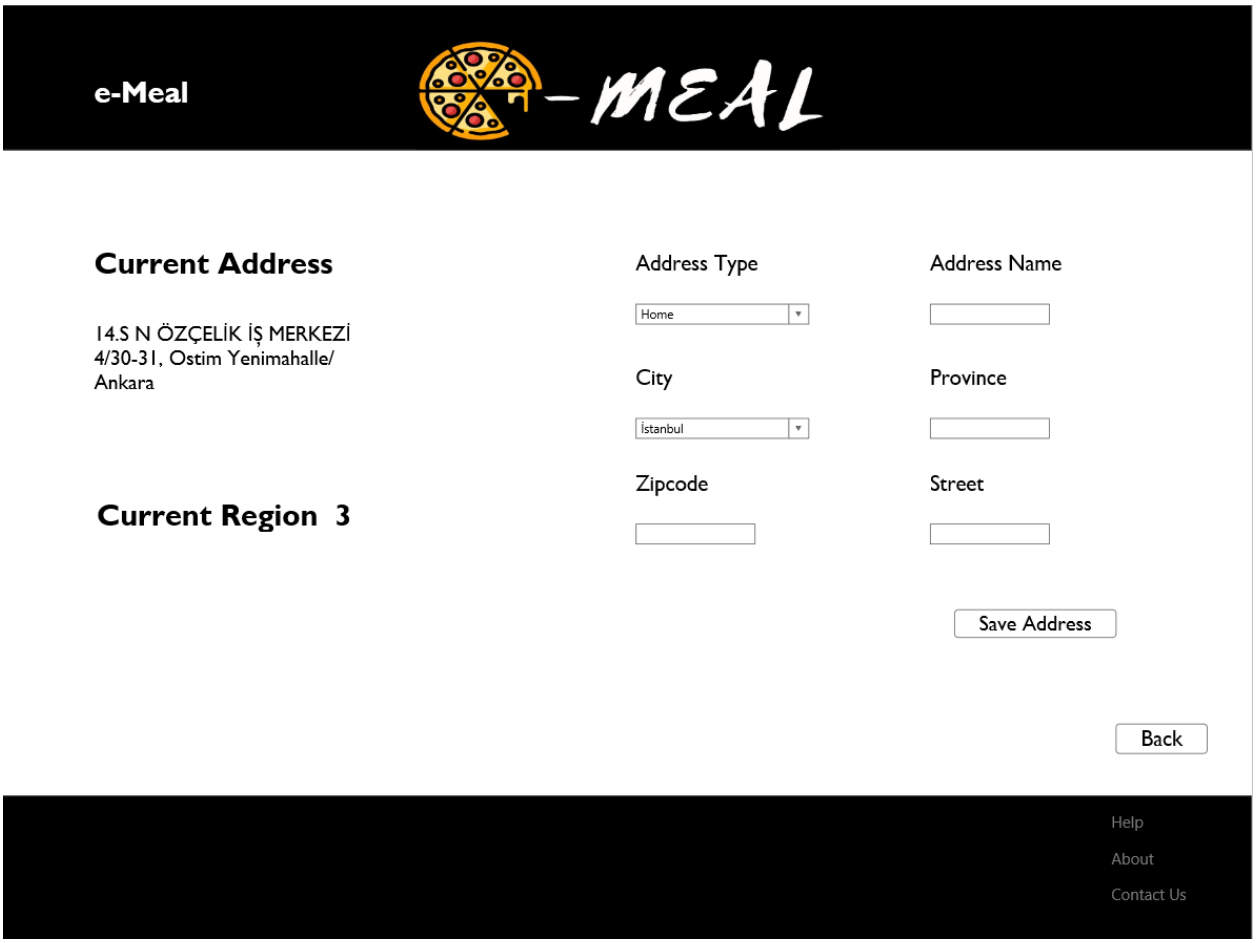
3.10 Profile Edit Page



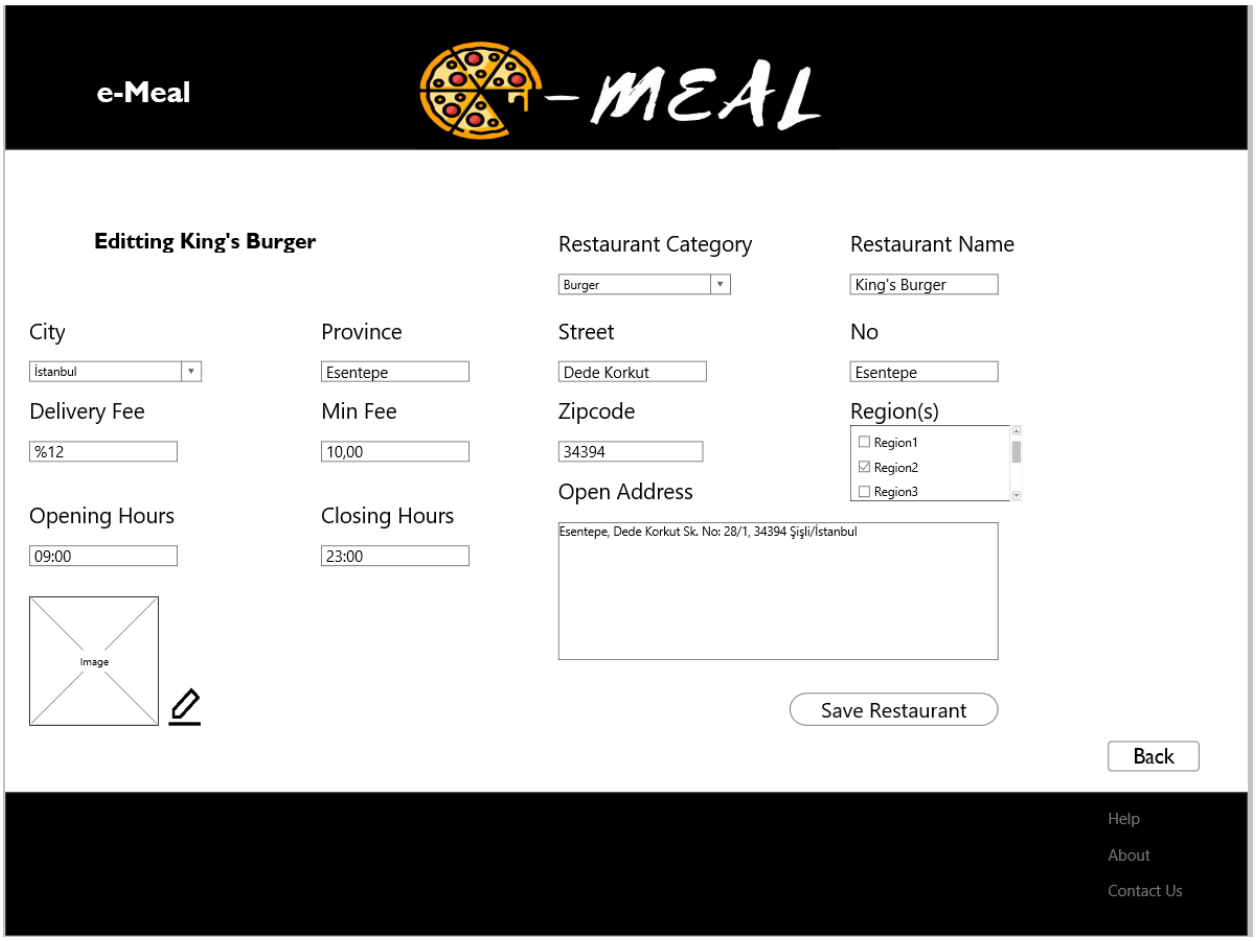
3.11 Change Account Settings Page



3.12 Change Address Page

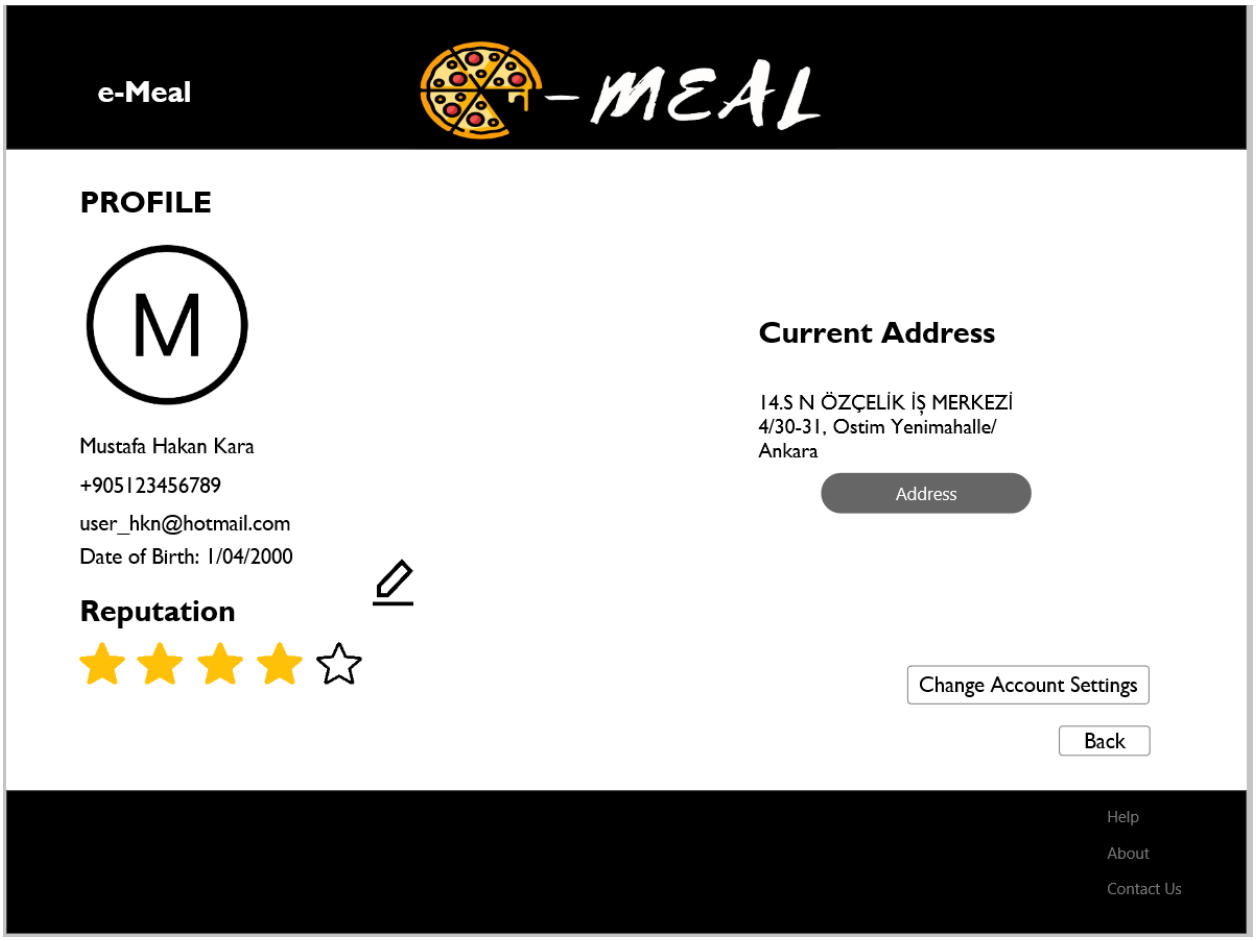


3.13 Restaurant Owner’s Edit Restaurant Page

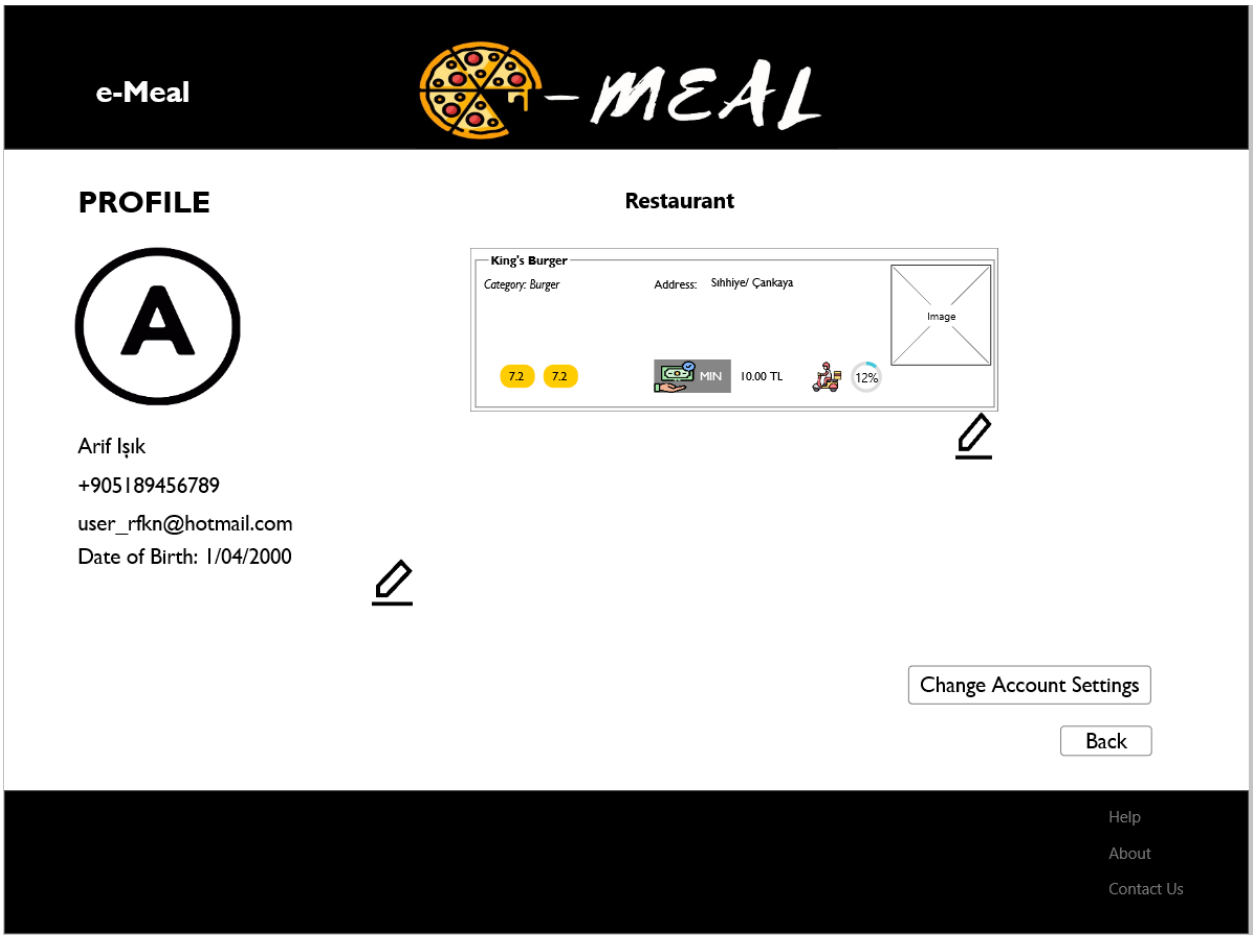


3.14 Registration Page

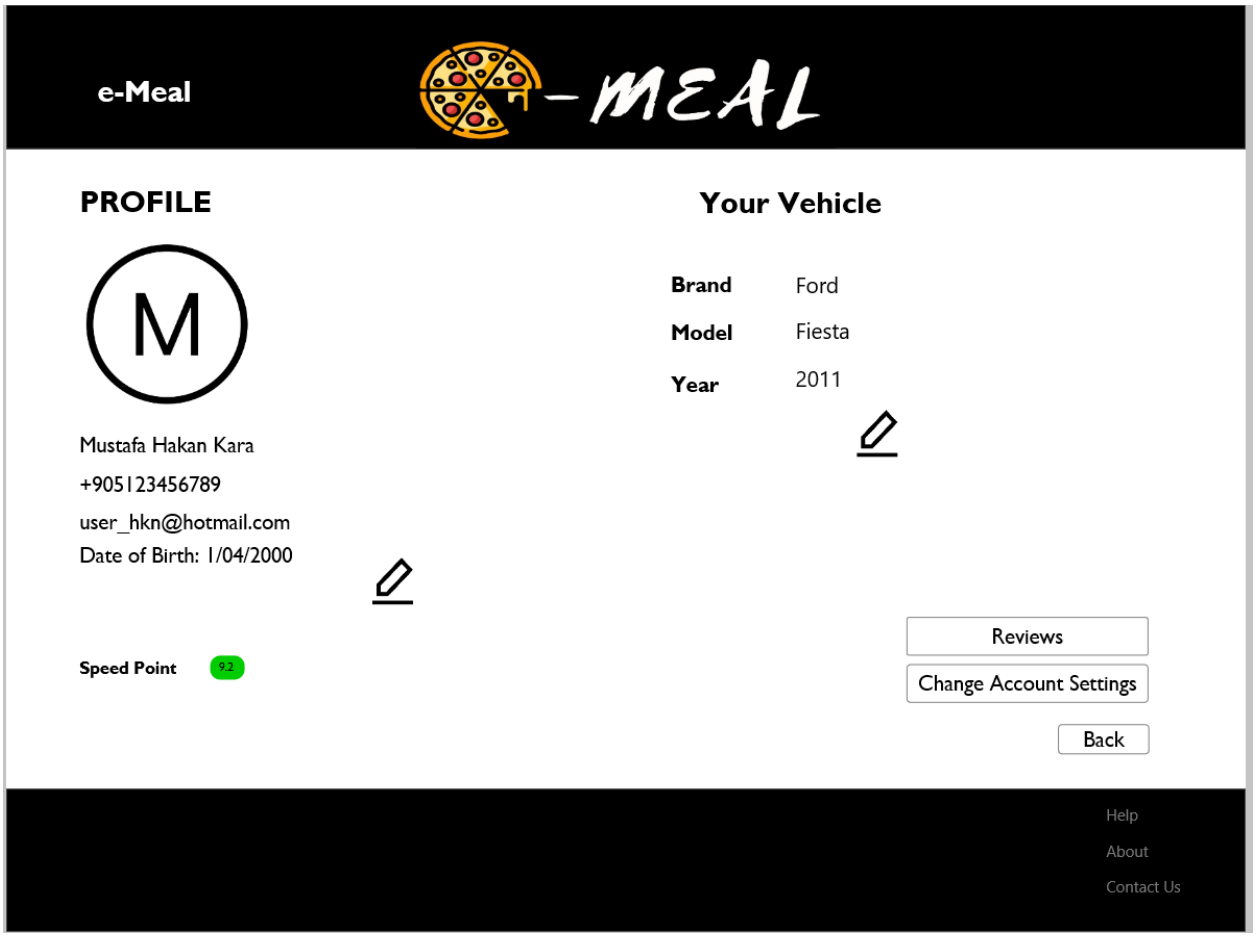
3.15 Customer’s Profile Page



3.16 Restaurant Owner's Profile Page



3.17 Delivery Guy's Profile Page



4 Queries

4.1 Login Page Queries

Check if the result of this query is not empty to validate login information:
(If login info was correct, then store user info in session in order to use later.)

```
SELECT
  *
FROM
  Users
WHERE
  (LOWER(name) LIKE '$username')
  AND (password LIKE '$password')
```

We need to run 3 queries to find the type of user (customer, owner, delivery):
(When we see one of these queries' results as non empty, we know that the user is the type of the table information was selected from. If there is any extra information other than user id, store that in the session.)

```
SELECT * FROM Customers WHERE id = '$id'

SELECT * FROM Restaurant_Owners WHERE id = '$id'

SELECT * FROM Delivery_Guys WHERE id = '$id'
```

4.2 Customer's Home Page Queries

Get previous orders of customer:

```
SELECT
  *
FROM
  Orders
WHERE
  customer_id = '$ id'
```

Get All Restaurants That is in the Customer's Region:

```
SELECT
  *
FROM
  Restaurants R
WHERE
  EXISTS (
    SELECT
      *
    FROM
      Restaurant_Regions RR
    WHERE
      RR.region_id = (
        SELECT
          C.region_id
        FROM
          Customer_Regions C
        WHERE
          C.customer_id = '$ id'
      )
      AND RR.restaurant_id = R.restaurant_id
  )
```

4.3 Restaurant Owner's Home Page Queries

Selects Customers And Their Orders Who Ordered From The Restaurant of The Owner, This Will Then be Sorted by Date and Listed:

```
SELECT
  *
FROM
  Customers C, Orders O
WHERE
   O.restaurant_id = '$restaurant_id' AND C.id = O.customer_id
```

Selects Meals Of The Restaurant and Number of Orders for Each Meal:

```
SELECT (
    SELECT
      COUNT(*) AS order_count
    FROM
      In_Order IO
    WHERE
      IO.item_id = I.item_id ),
  I.name,
  I.price,
  I.description,
I.group_name
FROM
  Items I
WHERE
  I.restaurant_id = '$restaurant_id'
```

Deletes the Selected Meal From Items Table:

```
DELETE FROM Items WHERE item_id = '$item_id'
```

Deletes the Selected Meal's Options From Options Table:

```
DELETE FROM Options WHERE item_id = '$item_id'
```

4.4 Delivery Guy's Home Page Queries

Selects Regions The Delivery Guy is Serving:

```
SELECT
  *
FROM
  Regions R
WHERE
  R.region_id IN (
    SELECT
      S.region_id
    FROM
      Serves_Region S
    WHERE
      S.delivery_guy_id = '$id'
  )
```

Selects All The Orders Delivery Guy is Currently Delivering:

```
SELECT
  *
FROM
  Assignments A,
  Orders O,
  Restaurants R,
  Customers C
WHERE
  C.id = O.customer_id
  AND R.restaurant_id = O.restaurant_id
  AND O.order_id = A.order_id
  AND O.delivery_datetime = NULL
  AND A.decision = 1
  AND A.delivery_guy_id = '$id'
```

Selects The Current Pending Assignment (returns empty is there is none):

```
SELECT
  *
FROM
  Assignments A,
  Orders O,
  Restaurants R,
  Customers C
WHERE
  C.id = O.customer_id
  AND R.restaurant_id = O.restaurant_id
  AND O.order_id = A.order_id
  AND O.order_status = ‘on the way’
  AND A.decision = NULL
  AND A.delivery_guy_id = '$id'
```

Selects The Speed Point to Display:

```
SELECT speed_point FROM Delivery_Guys WHERE id = '$id'
```

Sets order status to delivered:

```
UPDATE Orders SET order_status = 'delivered', delivered_datetime = '$current_date_time'
WHERE order_id = '$order_id'
```

4.5 Customer's Restaurant Page Queries

Selects Meals of The Restaurant Along With Their Order Counts:

```
SELECT (
    SELECT
      COUNT(*) AS order_count
    FROM
      In_Order IO
    WHERE
      IO.item_id = I.item_id
  ),
  I.name,
  I.price,
  I.description,
  I.group_name
FROM
  Items I
WHERE
  I.restaurant_id = '$restaurant_id'
```

Selects Reviews of The Restaurant:

```
 SELECT
  *
FROM
  Reviews
WHERE
  order_id IN (
    SELECT
      order_id
    FROM
      Orders
    WHERE
      restaurant_id = '$restaurant_id' )
```

4.6 Customer’s Placing Order Page Queries

When Order is Placed, Inserts it into Orders table:

```
INSERT INTO
  Orders
VALUES
  (
    '$order_id',
    '$restaurant_id',
    '$id',
    '$creation_datetime',
    '$planned_datetime',
    NULL,
    '$customer_notes',
    ‘pending’
  )
```

Then, inserts each item for the order in the In_Order table (This will be done in a for loop for each item and its options):

```
INSERT INTO
  In_Order
VALUES
  (
    '$item_id',
    '$option_id',
    '$item_instance_id',
    '$order_id',
    '$count'
  )
```

4.7 Restaurant Owner’s Add Meal Page Queries

Inserts the new Item to the items table:

```
INSERT INTO
  Items
VALUES
  (
    '$item_id',
    '$restaurant_id',
    '$name',
    '$group_name',
    '$price',
    '$discount',
    '$description',
    '$image_url',
    '$is_available'
  )
```

Inserts the options of the Item to the options table (will be ran in a for loop to insert each option one by one):

```
INSERT INTO
  Options
VALUES
  (
    '$option_id',
    '$item_id',
    '$group_name',
    '$name',
    '$description',
    '$price'
  )
```

4.8 Restaurant Owner’s Edit Meal Page Queries

Updates the Selected Item:

```
UPDATE
  Items
SET
  name = '$name',
  Group_name = '$group_name','
  price = '$price',
  discount = '$discount',
  description = '$description',
  image_url = '$image_url',
  is_available = '$is_available'
WHERE
  item_id = '$item_id'
```

Updates the Options of the Selected Item (will be done in a for loop for changed options):

```
UPDATE
  Options
SET
  group_name = '$group_name',
  name = '$name',
  description = '$description',
  price = '$price'
WHERE
  item_id = '$item_id'
```

Updates the Options of the Selected Item (will be done for deleted options):

```
DELETE FROM
  Options
WHERE
  option_id = '$option_id'
```

4.9 Restaurant Owner’s Maintain Orders Page

Select pending orders to display:

```
SELECT
  *
FROM
  Orders
WHERE
  order_status = 'being reviewed'
```

Select preparing orders to display:

```
SELECT
  *
FROM
  Orders
WHERE
  order_status = 'being prepared'
```

Select orders that are on the way to the customer:

```
SELECT
  *
FROM
  Orders
WHERE
  order_status = 'on the way'
```

Select orders that are delivered:

```
SELECT
  *
FROM
  Orders
WHERE
  order_status = 'delivered'
```

If owner accepts a waiting order:

```
UPDATE
  Orders
SET
  order_status = 'being prepared'
WHERE
  order_id = '$order_id'
```

If owner finishes preparing the order

```
UPDATE
  Orders
SET
  order_status = 'on the way'
WHERE
  order_id = '$order_id'
```

4.10 Profile Edit Page Queries

Select user’s image_url to display:

```
SELECT
  image_url
FROM
  Users
WHERE
  id = '$user_id'
```

Update user’s profile photo:

```
UPDATE
  Users
SET
  image_url = '$image_url'
WHERE
  id = '$user_id'
```

Update user’s profile information:

```
UPDATE
  Users
SET
  first_name = '$first_name',
  middle_name = '$middle_name',
  last_name = '$last_name',
  phone_number = '$phone_number'
WHERE
  id = '$user_id'
```

4.11 Change Account Settings Page Queries

Select user's image_url to display

```
SELECT
  image_url
FROM
  Users
WHERE
  id = '$user_id'
```

Update user's account information if user entered old password correctly

```
UPDATE
  Users
SET
  email = '$e_mail',
  password = '$password'
WHERE
  id = '$user_id'
  AND password = '$old_password'
```

4.12 Change Address Page Queries

Selects user's address to display:

```
SELECT
  street,
  city,
  zip
FROM
  Users
WHERE
  id = '$user_id'
```

Selects user's region to display:

```
SELECT
  region_id
FROM
  Customer_Regions
WHERE
  customer_id = '$user_id'
```

Users change their address:

```
UPDATE
  Users
SET
  street = '$street',
  city = '$city',
  zip = '$zip_code'
WHERE
  id = '$user_id'
```

4.13 Restaurant Owner's Edit Restaurant Page Queries

Existence of a Restaurant will be checked before editing:

```
SELECT
  COUNT(*)
FROM
  Restaurants,
  Restaurant_Owners
WHERE
  id = owner_id
  and id = '$user_id'
```

Editing Existing Restaurant:

```
UPDATE
  Restaurants
SET
  opening_time = '$open_time',
  closing_time = '$close_time',
  minumum_fee = '$min_fee',
  street = '$street',
  city = '$city',
  zip = '$zip_code',
  image_url = '$image_url'
```

For every region checkbox checked, following will be evaluated:

```
INSERT INTO
  Restaurant_Regions
VALUES('$restaurant_id', '$region_id')
```

For every region checkbox unchecked(region deletion), following will be evaluated:

```
DELETE FROM
  Restaurant_Regions
WHERE
  restaurant_id = '$restaurant_id'
  and region_id = '$region_id'
```

Adding a New Restaurant:

```
INSERT INTO
  Restaurants
VALUES(
    '$restaurant_id',
    '$owner_id',
    NULL,
    NULL,
    '$is_open',
    '$opening_hours',
    '$closing_hours',
    '$street',
    '$min_fee',
    '$street',
    '$zip_code',
    '$image_url'
  )
```

For every region checkbox checked, following will be evaluated:

```
INSERT INTO
  Restaurant_Regions
VALUES('$restaurant_id', '$region_id')
```

4.14 Registration Page Queries

Sign Up to the Website:

```
INSERT INTO
  Users
values(
    '$id',
    '$name',
    '$m_name',
    '$l_name',
    '$mail',
    '$password',
    '$phone_n',
    '$date_of_b',
    '$acc_creation'
  );
```

4.15 Customer’s Profile Page Query

Retrieves the profile information of customers:

```
SELECT
  first_name,
  middle_name,
  last_name,
  email,
  street,
  city,
  zip,
  phone_number,
  date_of_birth
FROM
  Users U,
  Customers C
WHERE
  U.id = C.id
```

4.16 Restaurant Owner's Profile Page Query

Retrieves the profile information of restaurant owners:

```
SELECT
  first_name,
  middle_name,
  last_name,
  email,
  phone_number,
  date_of_birth,
  restaurant_name,
  service_point,
  taste_point,
  opening_time,
  closing_time,
  minumum_fee,
  R.street,
  R.city,
  R.zip,
  image_url
FROM
  Users U,
  Restaurant_Owners RO,
  Restaurants R
WHERE
  U.id = RO.id
  AND RO.id = R.owner_id
```

4.17 Delivery Guy's Profile Page Query

Retrieves the profile information of delivery guys:

```
SELECT
  first_name,
  middle_name,
  last_name,
  email,
  street,
  city,
  zip,
  phone_number,
  date_of_birth,
  speed_point,
  vehicle_year,
  vehicle_brand,
  vehicle_model
FROM
  Users U,
  Delivery_Guys D
WHERE
  U.id = D.id
```

5 Implementation Plan

We will be using MySQL for database management and PHP for development technologies. We will be using the software Navicat to check our schema visually. The program will be tested during development on local servers created with XAMPP. After the development phase, extensive testing will be conducted on Dijkstra servers of Bilkent University. The UI will be designed with the mockup tool called Mockplus. Then, GUI will be implemented with HTML, Javascript and PHP.

6 Websites

Here is the link to the repository for our project, where we publish our reports and source code: https://github.com/Z4R1Z/CS353-Group18

Here is the link to our actual project website: https://e-meal.github.io/