CS 353 Database Systems

Project Final Report

Project Name: Restaurant Express

Online Restaurant Order System

Group 35

Semih Teker - 21300964

Halil İbrahim Çavdar - 21501613

Yılmaz Berkay Beken – 21400294

Bahadır Durmaz - 21502098

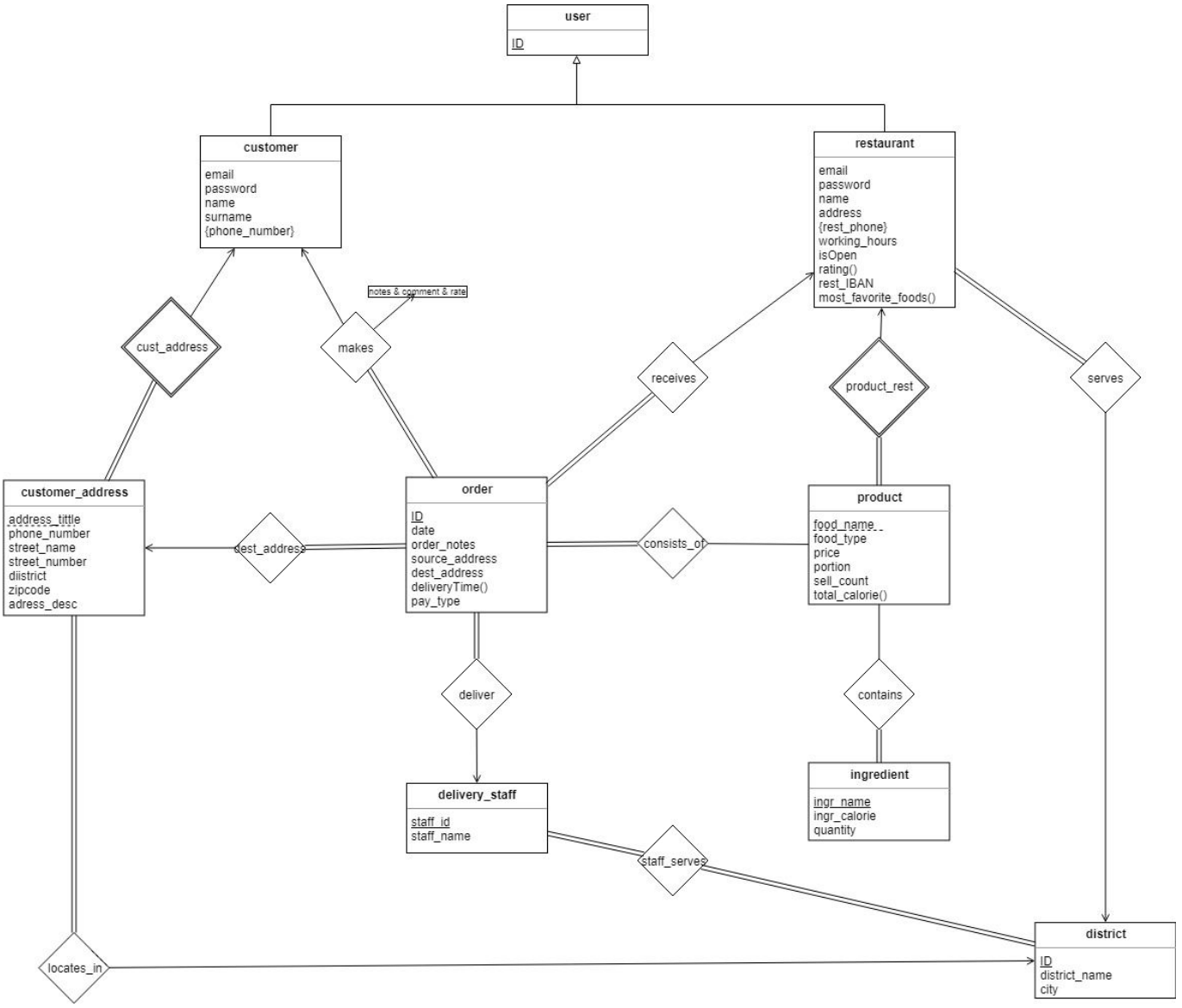
1. Overview

As our Database Project, we implement the “Restaurant Express” which is an online food service website such as Yemeksepeti.com. In our website, users can register and login, see the restaurant list located by their district, see the menus of restaurants with prices, make an order with notes, and make comment and rates for each order to get better service. Restaurants can login and register to the Restaurant Express by their names and passwords. Restaurants can display the taken orders of the customers by historical order on Restaurant Express. Restaurants are able to give the prepared orders to delivery system of the Restaurant Express. Restaurants can display the user comments of their restaurant’s to improve themselves. Thus, they will have chance to do better service and that will increase their rating. In addition, that will increase their rating so they will get more orders from the customers and will have chance to get more income thanks to comments.

2. Contributions of the Group Members

The backend of the database project is completed by Halil İbrahim Çavdar and Bahadır Durmaz; the frontend of the database is completed by Yılmaz Berkay Beken and Semih Teker. Until the implementation stage, the all sql statements are prepared by all group members and implemented by our backend team (Halil İbrahim Çavdar and Bahadır Durmaz) with fixing some errors. The overview of the pages (HTML) are prepared by frontend team (Semih Teker and Yılmaz Berkay Beken) and connections between HTML pages using Python Flask is done by Halil İbrahim Çavdar. Controller of the GUI part implemented by Bahadır Durmaz. In addition, first two reports were written by cooperation of the all group members. Final report was written by Semih Teker and Yılmaz Berkay Beken.

3. Revised E/R Diagram



4. Relation Schemas

4.1. Customer

Relational Model:

customer (c_id, email, password, name, surname, {phone_number})

4.2. Restaurant

Relational Model:

restaurant(r_id, email, password, name, address, {rest_phone}, working_hours, isOpen, rate, d_id)

PRIMARY KEY (r_id)

FOREIGN KEY (d_id) REFERENCES district(d_id)

4.3. Customer_Address

Relational Model:

customer_address(c_id, address_title, street_name, street_number, city, zipcode, adress_desc, d_id)

PRIMARY KEY (c_id, address_title)

FOREIGN KEY (c_id, name, surname) REFERENCES customer(u_id, name, surname)

FOREIGN KEY (d_id) REFERENCES district(d_id)

4.4. District

Relational Model:

district(d_id, district_name, city)

PRIMARY KEY (d_id)

4.5. Order

Relational Model:

order(ord_id, date, order_notes, source_address, dest_address, pay_type, comment, rate, cust_id, rest_id, staff_id)

PRIMARY KEY (ord_id)

FOREIGN KEY (cust_id) REFERENCES customer_address(u_id)

FOREIGN KEY (rest_id) REFERENCES restaurant(u_id)

FOREIGN KEY (staff_id) REFERENCES delivery_staff(staff_id)

FOREIGN KEY (source_address) REFERENCES restaurant(address)

FOREIGN KEY (dest_address) REFERENCES customer_address)

4.6. Delivery_staff

Relational Model:

delivery_staff(staff_id, staff_name)

4.7. Product

Relational Model:

product(rest_id, food_name, food_type, price, portion)

PRIMARY KEY (rest_id, food_name)

FOREIGN KEY (rest_id) REFERENCES restaurant(r_id)

4.8. Ingredient

Relational Model:

ingredient(ingr_name, ingr_calorie, quantity)

PRIMARY KEY (ingr_name)

4.9. Contains

Relational Model:

contains(rest_id, food_name, ingr_name)

PRIMARY KEY (rest_id, food_name, ingr_name)

FOREIGN KEY (rest_id, food_name) REFERENCES product(rest_id, food_name)

FOREIGN KEY (ingr_name) REFERENCES ingredient(ingr_name)

4.10. Consist Of

Relational Model:

consist_of(ord_id, rest_id, food_name)

PRIMARY KEY (ord_id, rest_id, food_name)

FOREIGN KEY (ord_id) REFERENCES order(ord_id)

FOREIGN KEY (rest_id, food_name) REFERENCES product(rest_id, food_name)

4.11. Staff Serves

Relational Model:

staff_serves(d_id, staff_id)

FOREIGN KEY (d_id) REFERENCES district(d_id)

FOREIGN KEY (staff_id) REFERENCES delivery_staff(staff_id)

5. Implementation Details

We utilized Python 3 with the Flask Web Framework module for our model and controller (backend). Frontend view was first implemented by Bootstrap with CSS overlay designs, the resulting HTML code was adapted to call our Python code using Flask whenever a trigger was detected to redirect to a new page and to POST and GET information from our SQLite3 database whenever required.
Our database was written in SQLite3. Though at first we planned and for a while implemented our database in MySQL we quickly found out that it was impeding practicality, as we had to create and invoke a lot of stored procedures and it wasn't as cross-platform as other systems. In the end we decided to go with SQLite3 since its integration with Python seemed seamless compared with MySQL. Unlike in MySQL we needn't setup MySQL for each computer, didn't require stored procedures and our Flask module was directly compatible with SQLite3. We used DB Browser for SQLite to visualize our database design and access the database through a GUI.

A problem we faced was that none of us were proficient in Flask let alone most of us ha d never heard of it at the start. So all of us had to learn it from scratch and progress through tutorials. Most of the later problems we faced were caused by not being proficient in Flask and it's native usage of SQLite3 through Python.
We also faced some problems while connecting our view HTML codes to our backend code and the database, specifically when adding "Form" tags to mark blocks where input's will be taken to be processed in the database and to redirect to our Python app route functions. A bug we encountered regularly at first would be that the "Form" and other tags added later to the Bootstrap design in HTML to connect to the controller would alter the original view. We fixed this issue by minimizing the blocks interfered in HTML.
Another issue we faced was sending and receiving data through indents to our HTML code. For example while implementing sign up and login features we had difficulty extracting text fields (email, password…) to be processed by our backend code or when we wanted to list a specific user's all past orders, which can differ in numbers, we had to process the necessary query in our backend code and send it to our View component to be displayed.

6. Advanced Database Features and Reports

View - Customer View of Order

SQL query of the view is:

```
CREATE VIEW cust_order AS
SELECT O.date, C.food_name, R.name, P.price, P.portion
FROM order O NATURAL JOIN consist_of C NATURAL JOIN product P, restaurant R
WHERE R.u_id=C.rest_id
```

View - Restaurant View of Order

SQL query of the view is:

```
CREATE VIEW rest_order AS
SELECT O.date, O.order_notes, O.dest_address, O.staff_id, O.pay_type, C.food_name,
P.price, P.portion
FROM order O NATURAL JOIN consist_of C NATURAL JOIN product P, restaurant R
WHERE R.u_id=C.rest_id AND R.name=@currentRestName
```

View - Delivery Staff View of Order

SQL query of the view is:

```
CREATE VIEW delivery_order AS
SELECT O.date, O.order_notes, O.dest_address, O.staff_id, O.pay_type, C.food_name,
P.price, P.portion
FROM order O NATURAL JOIN consist_of C NATURAL JOIN product P, restaurant R
WHERE R.u_id=C.rest_id AND O.staff_id=@CurrentStaffID
```

Report - Number of orders of top 3 customers

SQL query of this report is:

```
CREATE VIEW order_count_of_cust_report AS
SELECT O.cust_id, C.name, C.surname, COUNT(*)
FROM order O JOIN customer C
ON O.cust_id=C.u_id
GROUP BY O.cust_id
ORDER BY COUNT(*) DESC
LIMIT 3
```

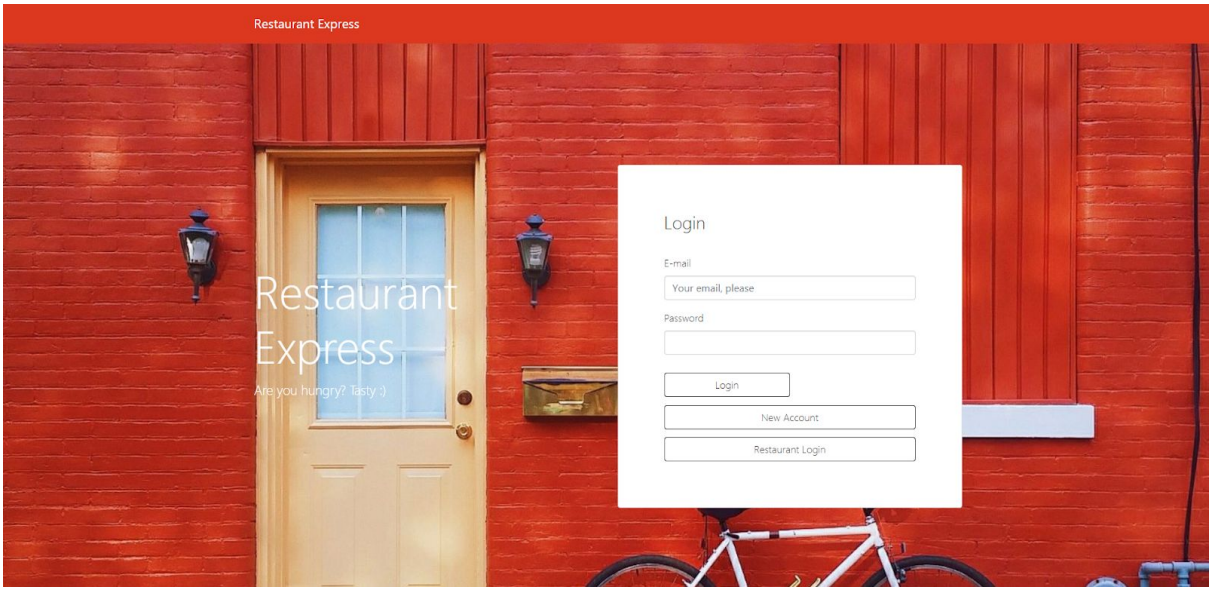
Report - Amount of money from one customer

SQL query of this report is:

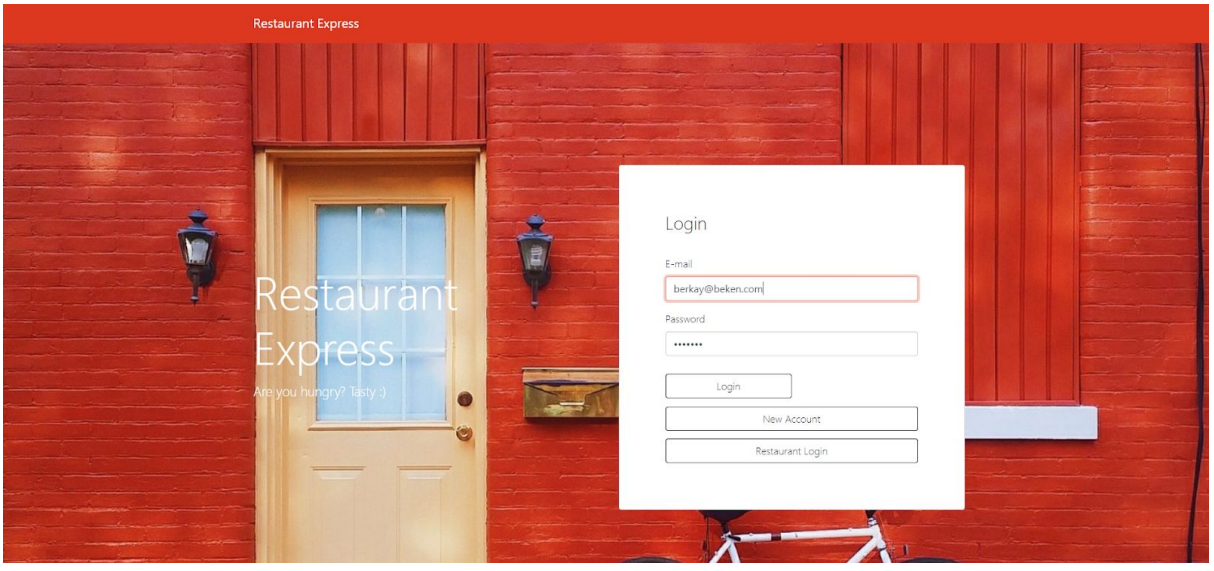
```
SELECT O.cust_id, C.name, C.surname,SUM(P.price) as amount
FROM order O NATURAL JOIN consist_of C NATURAL JOIN product P, customer C
WHERE O.cust_id=C.u_id
GROUP BY O.cust_id
```

7. User's Manual

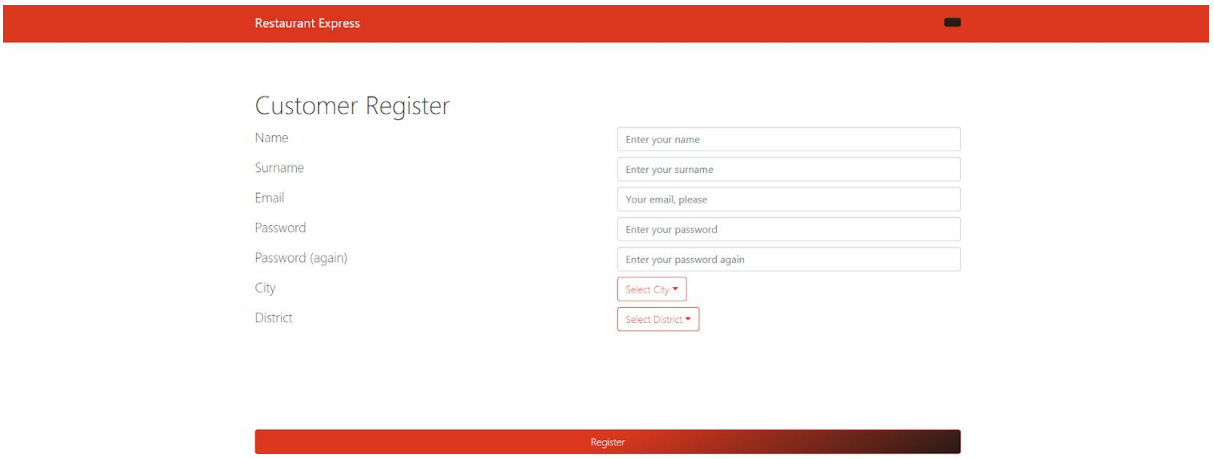
7.1. Customer Login Page

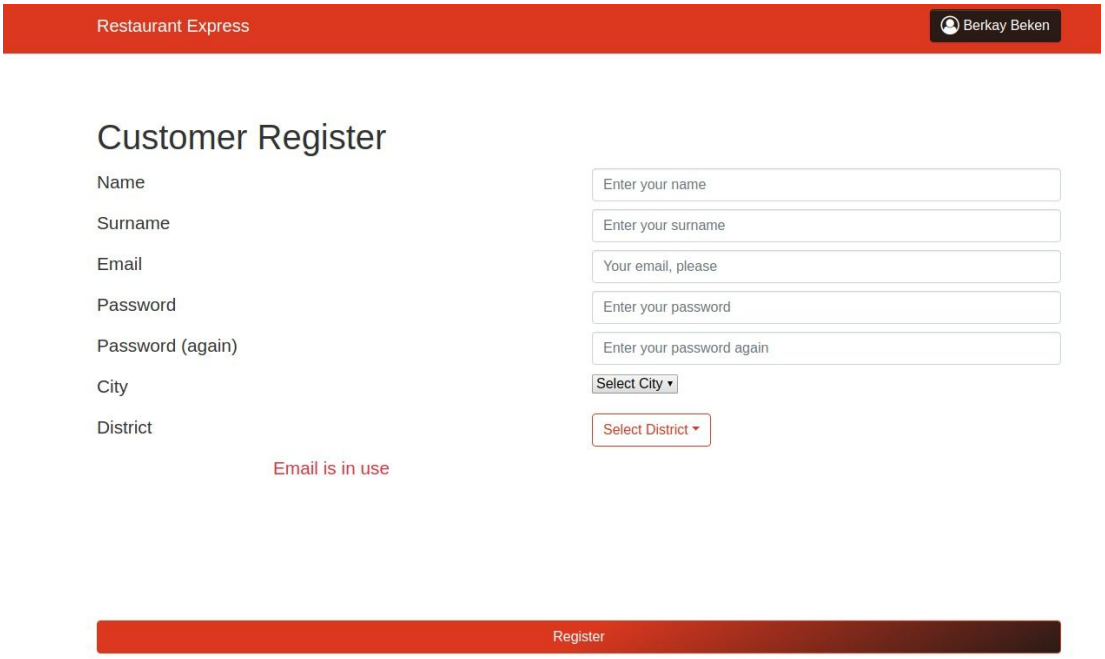


If customer has an account, he/she can enter the system by entering the email and password. Otherwise, he/she can go to register page by clicking the "Create Account" button. Restaurants can also go to login page by clicking the "Restaurant Login" link.



7.2. Customer Register Page





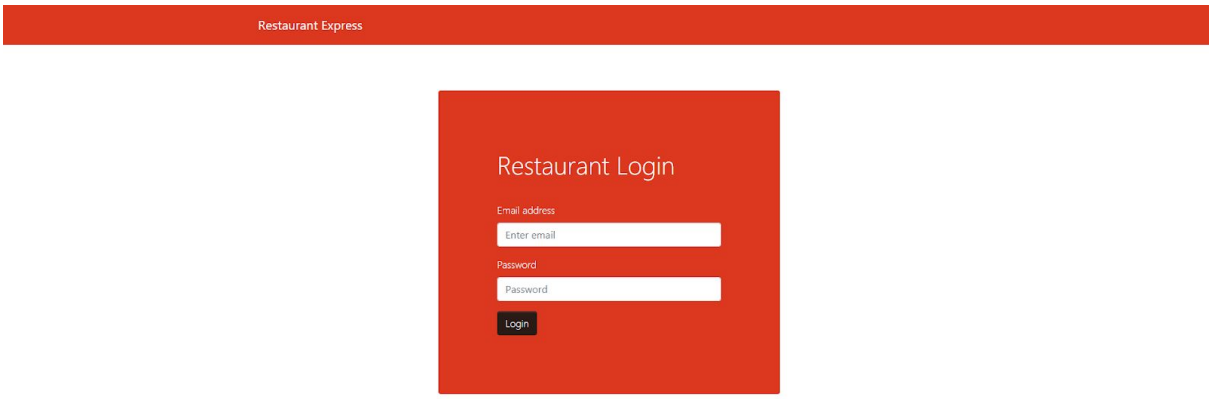
Name = Berkay

Surname = Beken

Email = berkay@beken.com

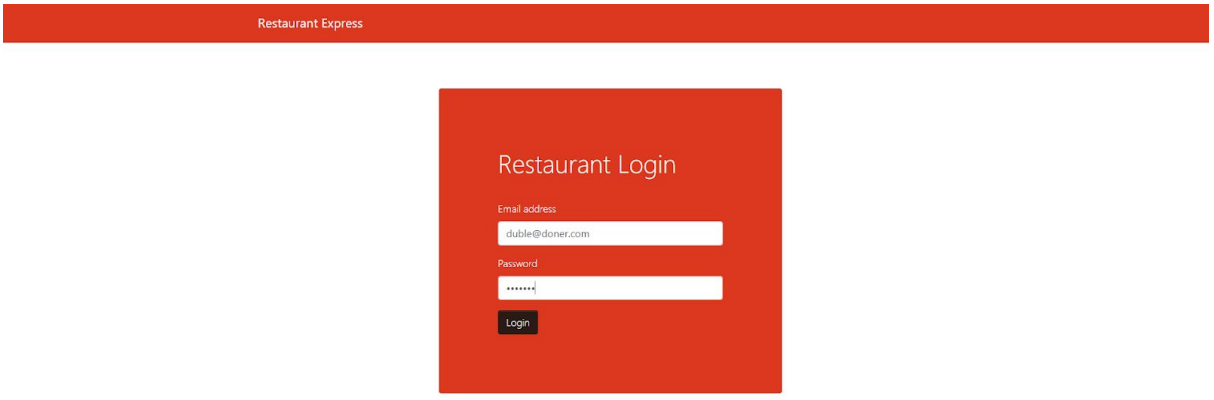
Password = CS353fn

7.3. Restaurant Login Page

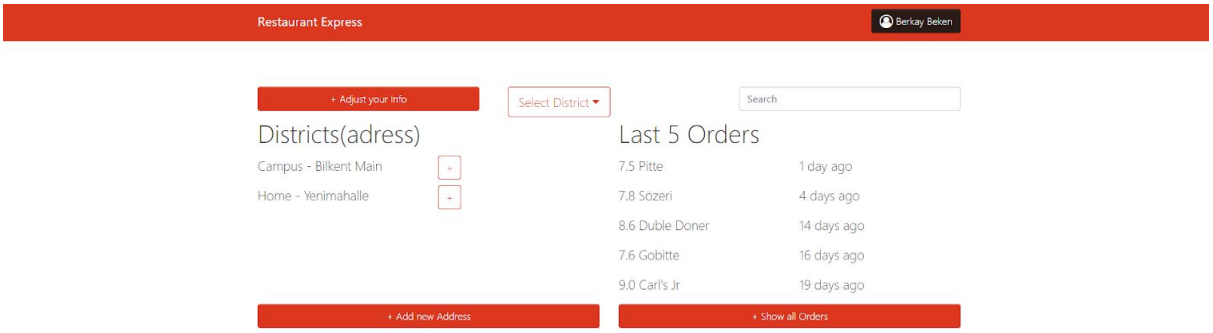


Email address = duble@doner.com

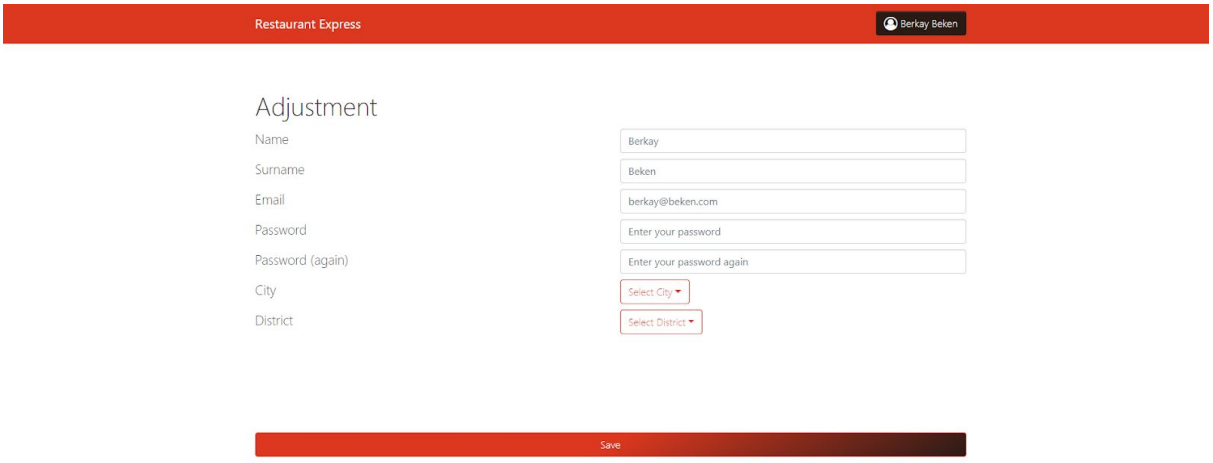
Password = dublbil



7.4. Home Page

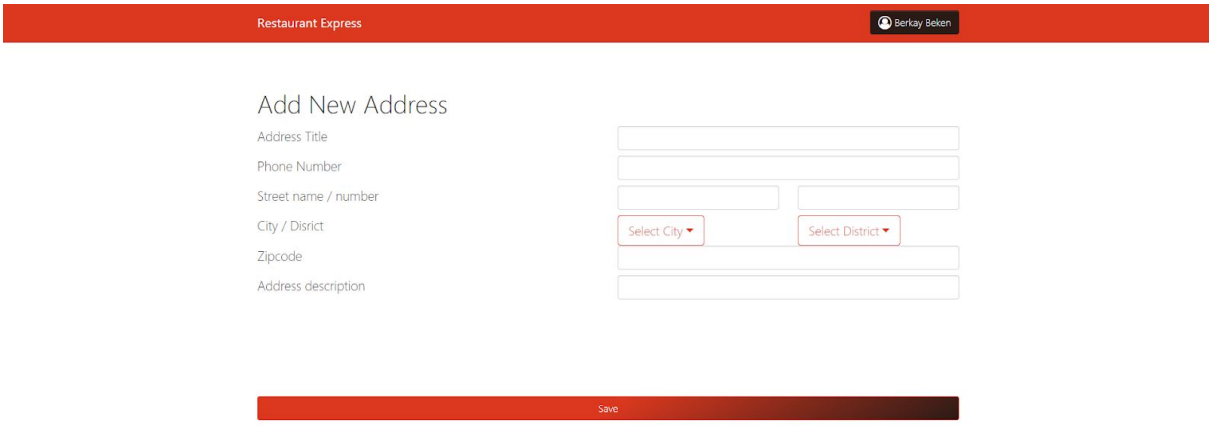


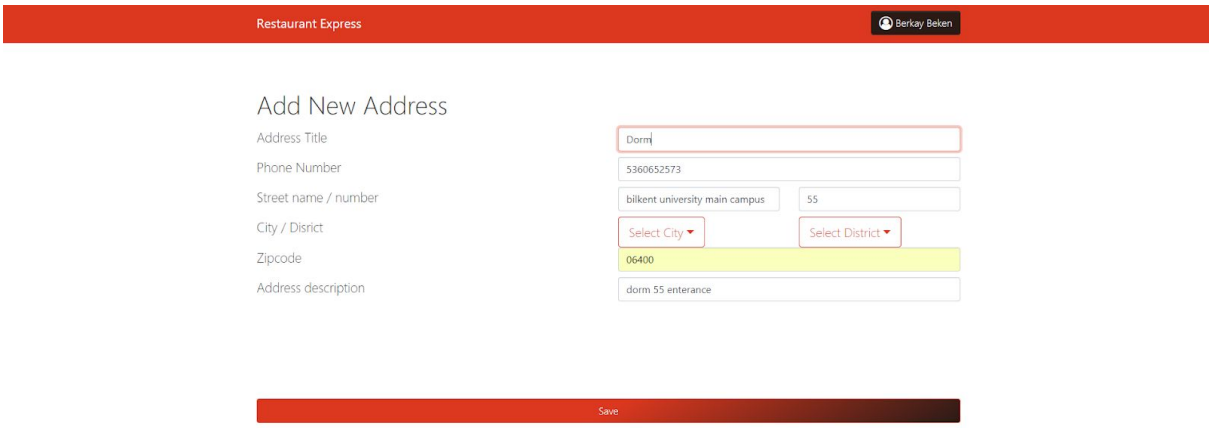
7.5. Adjustment Page



Customers can see their informations and they can adjust them. They can change their names, surnames and passwords by entering the new ones in the corresponding text fields. At the end, they can save their new informations by clicking “save” button or they can leave without any changes by clicking the “back” button.

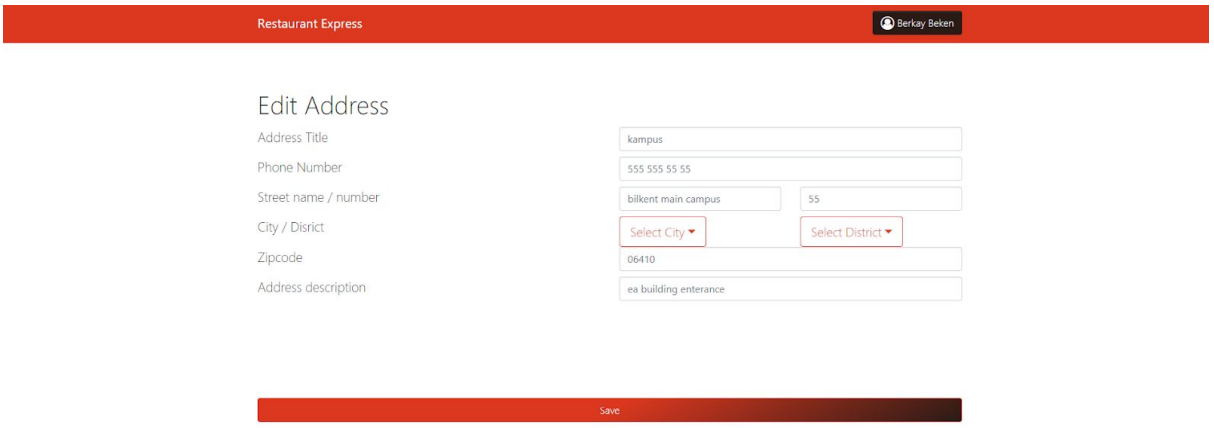
7.6. Add new Address Page





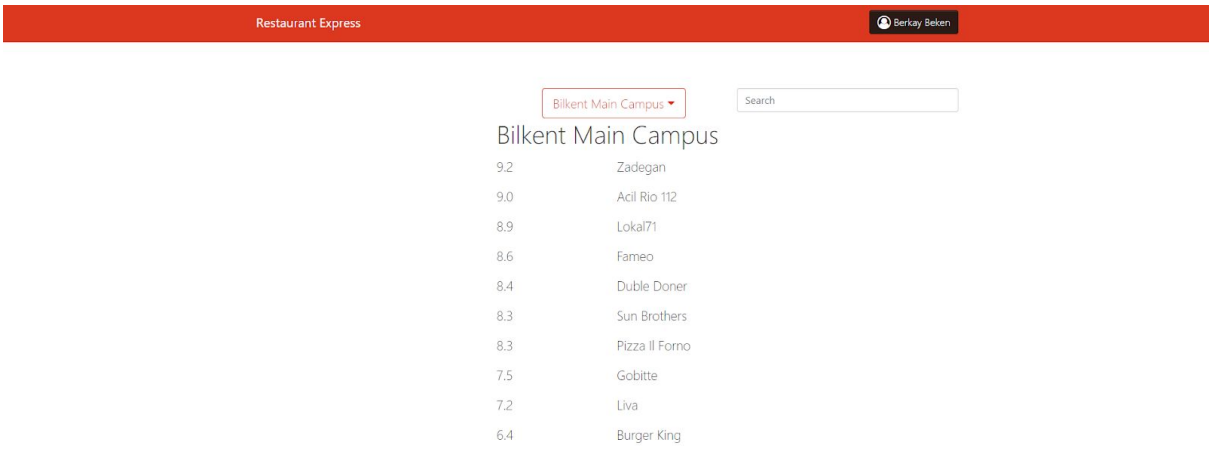
Customers can add new address by entering the address title, phone number, street-name, street-number, city, district, zip-code and address description. They can save the new address by clicking the “done” button or they can leave without any changes by clicking the “back” button.

7.7. Edit Address Info Page



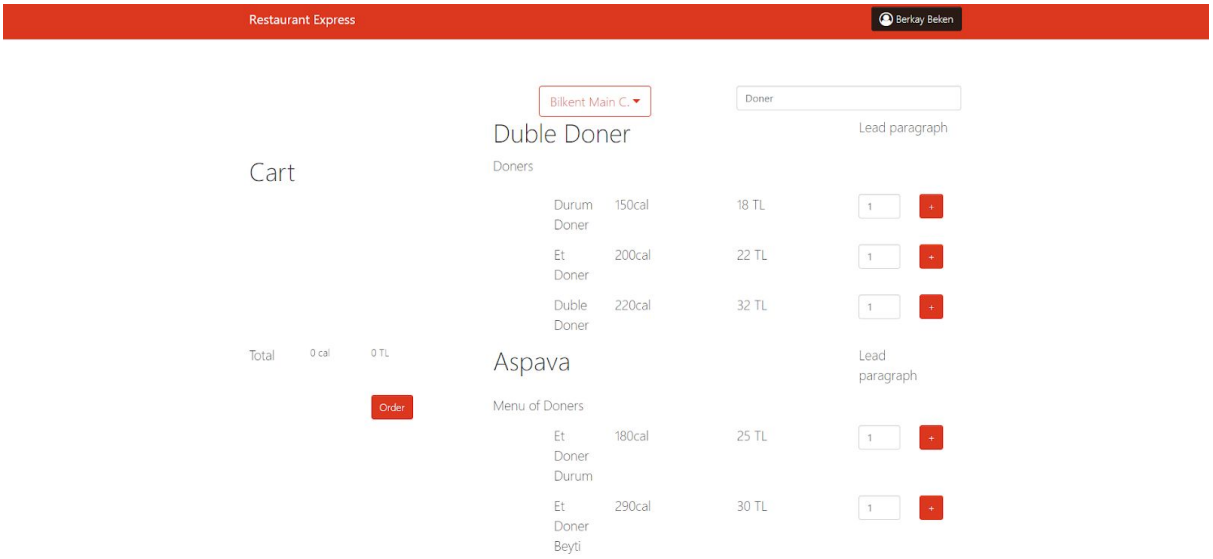
Customers can see their address informations and they can adjust them. They can change their address title, phone number, street-name, street-number, city, district, zip-code and address description by entering the new ones in the corresponding text fields. At the end, they can save their new informations by clicking “save” button or they can leave without any changes by clicking the “back” button.

7.8. List Restaurants Page



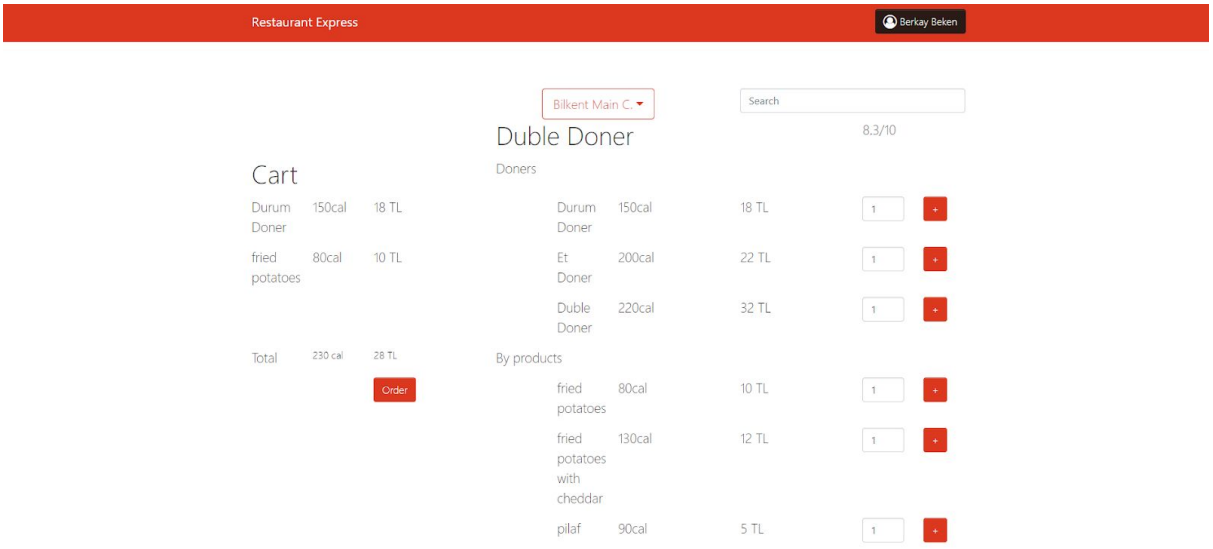
Customers can list the restaurant in the selected district. They can see the restaurant’s names and rates.

7.9. Search Page



Customers can search the meals with keywords and the district. The result of the search is that the list of the restaurants which have the desired meal on their menus. Under the each restaurant title, the related menus are listed. Customer can see the price of the menus near the menus' title and can choose the portion number. Customer can add any different menus of only one restaurant to cart by clicking the plus icon and the total price appears accordingly.

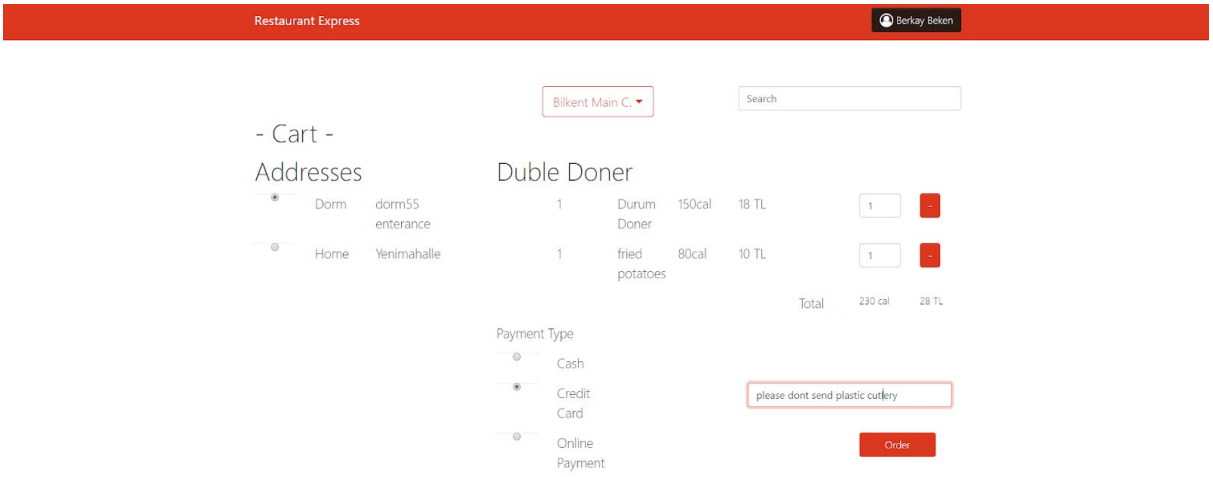
7.10. List Menus Of the Restaurant Page



Customer can list the all menus of the selected restaurant. Customer can see the price of the menus near the menus' title and can choose the portion number. The most favorite foods or best seller meals are appeared at the top of the menus (to represent it on the mock-up just one line is allocated, but the number of most favorite foods can be at most five

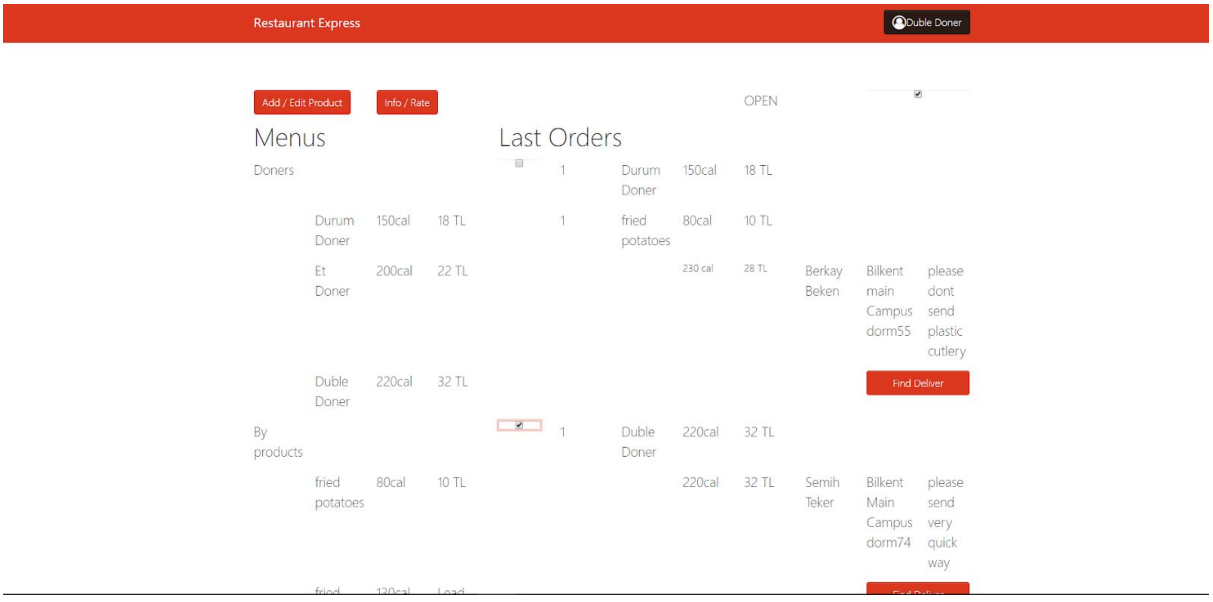
in our project). Customer can add any different menus of the restaurant to cart by clicking the plus icon and the total price appears accordingly.

7.11. Cart Page

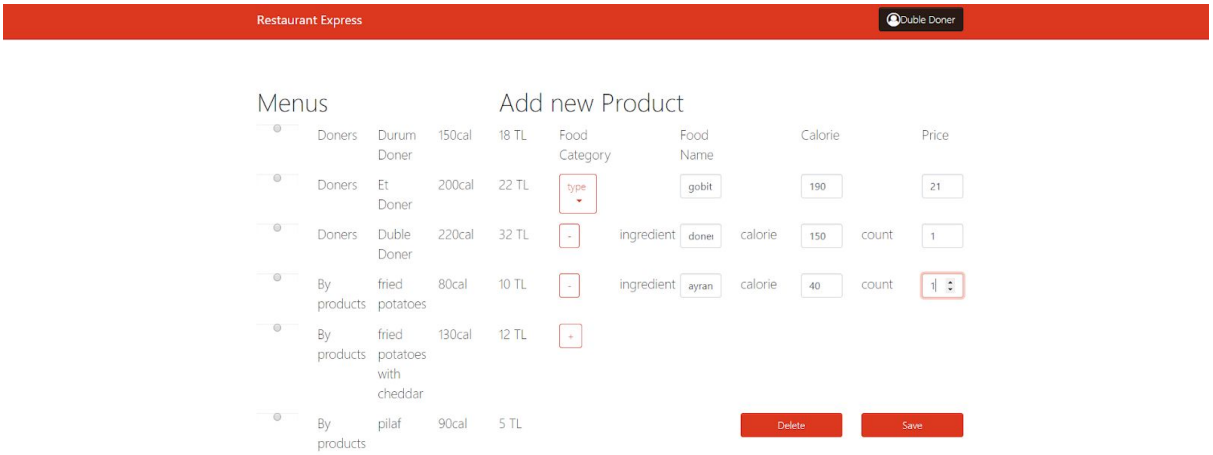


Process: Customer can list the all selected menus to buy. Customer can see the price of the menus near the menus' title and can increase or decrease the portion number by clicking the plus and minus icon, the price is updating by accordingly. Customer can discard the menus' item by decreasing the portion as zero. Customer can also see their list of address and they can choose one of these address if and only if address in the restaurant's district. Customers can choose one of the payment type and if they want to add a note about order, they can type the note box. At the end, by clicking "order" button they confirm the meal ordering.

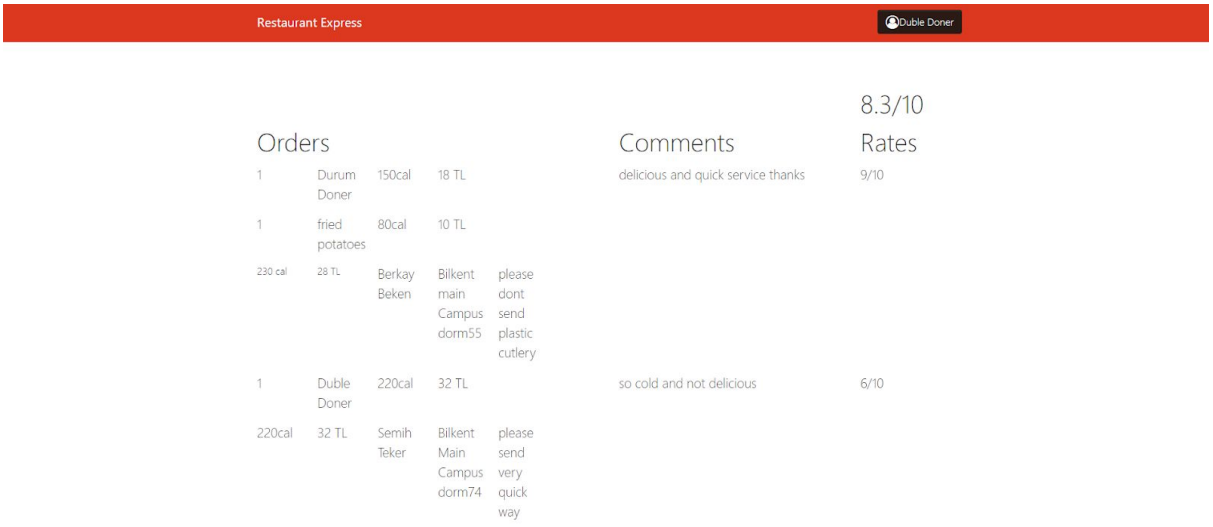
7.12. Restaurant Page



7.13. Add new Product / Edit Product Page



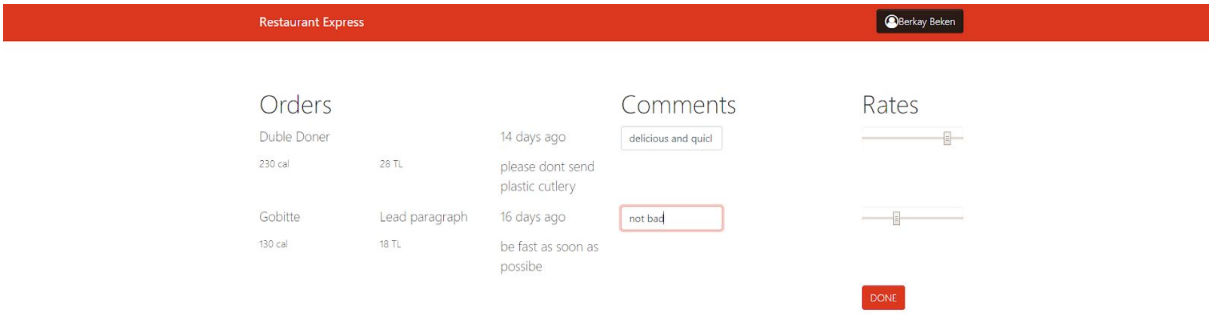
7.14. Restaurant Info Page



Restaurant can display the page by pressing Add/Edit Product button. Thus, restaurant will have a chance to change the number of remaining products. Restaurant can press info/rate button for displaying the customers' ratings and comments about their products. Thus, if their rate and comments are bad, they can better prepare their foods according to given rates and comments. If their rate and comments are good, more people will choose their restaurant. Thus, their profit will rise. Restaurant can edit their status by OPEN-CLOSE writing. If their status is open, customers can order but if it close any customer can not order from this restaurant. Restaurant can mark the box which takes part near the restaurant status, to show customers their order is preparing. Restaurant can find a delivery

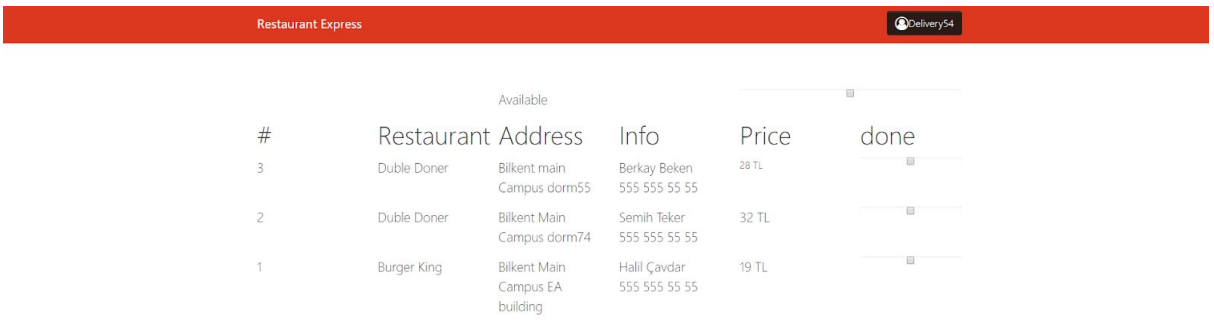
staff with pressing find deliver button after finishing the preparation of the order. Restaurant can display the notes of the customer about the orders. Restaurant can display the orders of the customers with their prices.

7.15. Make Comment(s) page



Users are able to write comments of their orders after receiving them to give feedback to restaurants.

7.16. Delivery Staff Page



Delivery stuff are able to display the orders of the users by sorting them according to order date.