CS 353 Database Management Systems Project

Design Report

Food Ordering and Delivery System

Group 19

Batuhan Özçömlekçi, 21703297

Yusuf Ziya Özgül, 21703158

Musa Ege Ünalan, 21803617

Mustafa Göktan Güdükbay, 21801740

Contents

ER Diagram

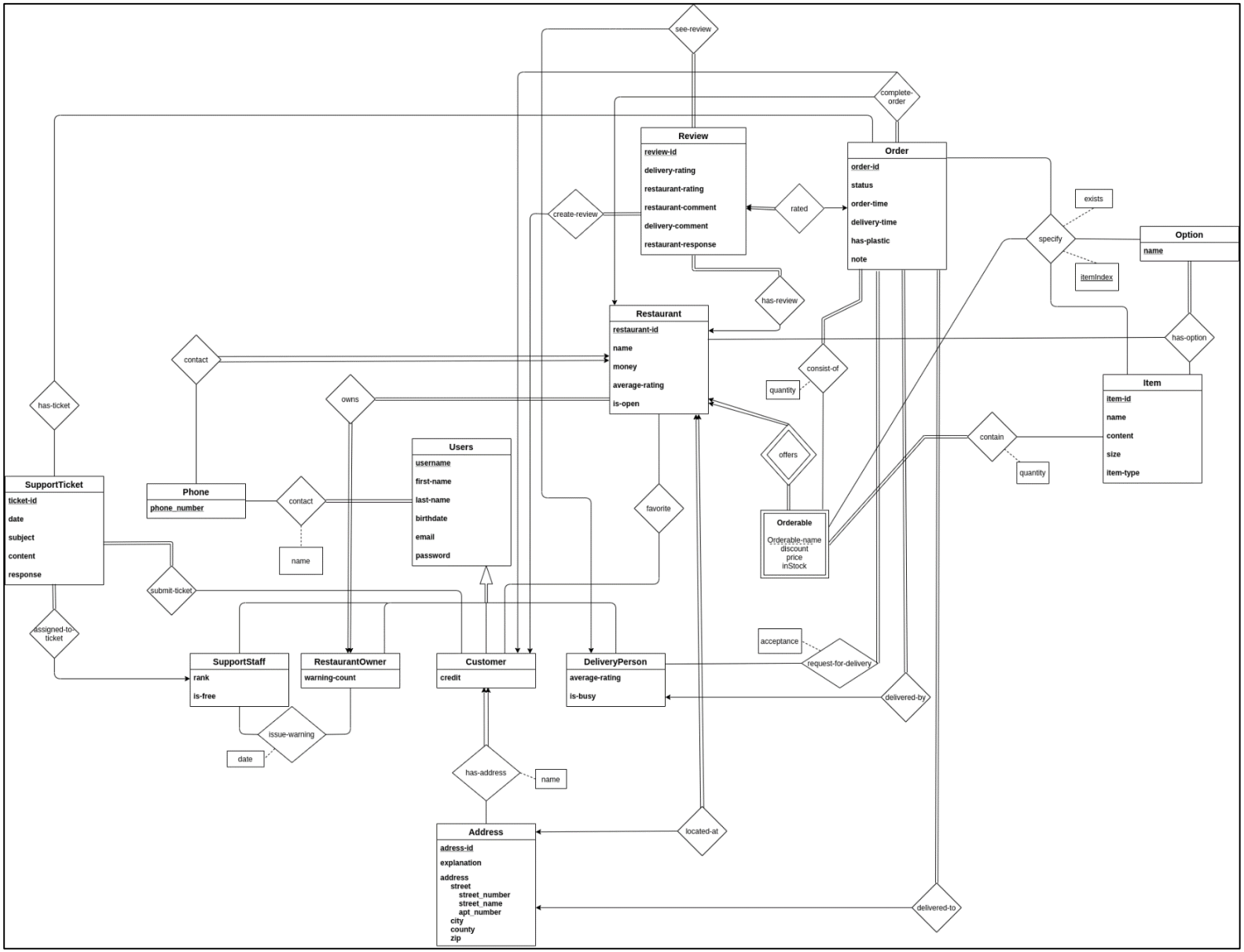

see-review
complete-order
Review
review-id
delivery-rating
restaurant-rating
restaurant-comment
delivery-comment
restaurant-response
Order
order-id
status
order-time
delivery-time
has-plastic
note
create-review
rated
exists
specify
Option
name
itemIndex
has-review
Restaurant
restaurant-id
name
money
average-rating
is-open
has-option
contact
consist-of
quantity
has-ticket
owns
Item
item-id
name
content
size
item-type
contain
quantity
offers
Users
username
first-name
last-name
birthdate
email
password
SupportTicket
ticket-id
date
subject
content
response
Phone
phone_number
contact
name
favorite
Orderable
Orderable-name
discount
price
inStock
submit-ticket
assigned-to-ticket
acceptance
request-for-delivery
SupportStaff
rank
is-free
RestaurantOwner
warning-count
Customer
credit
DeliveryPerson
average-rating
is-busy
delivered-by
issue-warning
date
has-address
name
Address
adress-id
explanation
address
street
street_number
street_name
apt_number
city
county
zip
located-at
delivered-to

Table Schemas

Item(item-id, name content, size, itemtype)

Option(name)

Order(order-id, status, order-time, delivery-time, has-plastic, note)

Review(review-id, delivery-rating, restaurant-rating, restaurant-comment, delivery-comment, restaurant-response)

Restaurant(restaurant-id, name, money, average-rating, is-open)

Orderable(restaurant-id, orderable-name, discount, price, instock)

- The attribute restaurant-id is a foreign key to Restaurant

Address(address-id, explanation, street, street_number, street_name, apt_number, city, county, zip)

Users(username, first-name, last-name, birthdate, email, password)

DeliveryPerson(username, average-rating, is-busy)

- The attribute username is a foreign key to Users

Customer(username, credit)

- The attribute username is a foreign key to Users

RestaurantOwner(username, warning-count)

- The attribute username is a foreign key to Users

SupportStaff(username, rank, is-free)

- The attribute username is a foreign key to Users

SupportTicket(ticket-id, date, subject, content, response)

Phone(phone-number)

Contain(restaurant-id, orderable-name, item-id, quantity)

- The attributes orderable-name and restaurant-id is a foreign key to Orderable
- The attribute item-id is a foreign key to Item

HasOption(restaurant-id, option-name, item-id)

- The attribute restaurant-id is a foreign key to Restaurant
- The attribute option-name is a foreign key to Option
- The attribute item-id is a foreign key to Item

Specify(item-id, option-name, order-id, restaurant-id, orderable-name, item-index, exists)

- The attribute item-id is a foreign key to Item
- The attribute option-name is a foreign key to Option
- The attribute order-id is a foreign key to Order
- The attributes orderable-name and restaurant-id is a foreign key to Orderable

Offers(restaurant-id, orderable-name)

- The attribute restaurant-id is a foreign key to Restaurant
- The attribute orderable-name is a foreign key to Orderable

ConsistOf(order-id, restaurant-id, orderable-name, quantity)

- The attribute order-id is a foreign key to Order
- The attributes orderable-name and restaurant-id is a foreign key to Orderable

HasReview(restaurant-id, review-id)

- The attribute restaurant-id is a foreign key to Restaurant
- The attribute review-id is a foreign key to Review

Rated(review-id, order-id)

- The attribute order-id is a foreign key to Rated
- The attribute review-id is a foreign key to Review

CompleteOrder(order-id, username, restaurant-id)

- The attribute username is a foreign key to Customer
- The attribute restaurant-id is a foreign key to Restaurant
- The attribute order-id is a foreign key to Order

SeeReview(review-id, username)

- The attribute username is a foreign key to DeliveryPerson
- The attribute review-id is a foreign key to Review

LocatedAt(address-id, restaurant-id)

- The attribute restaurant-id is a foreign key to Restaurant
- The attribute address-id is a foreign key to Address

HasAddress(address-id, username, name)

- The attribute username is a foreign key to Customer
- The attribute address-id is a foreign key to Address

IssueWarning(support-staff-username, restaurant-owner-username, date)

- The attribute support-staff-username is a foreign key to SupportStaff
- The attribute restaurant-owner-username is a foreign key to RestaurantOwner

AssignedToTicket(ticket-id, username)

- The attribute username is a foreign key to SupportStaff
- The attribute ticket-id is a foreign key to SupportTicket

SubmitTicket(ticket-id, username)

- The attribute username is a foreign key to Customer
- The attribute ticket-id is a foreign key to SupportTicket

HasTicket(ticket-id, order-id)

- The attribute order-id is a foreign key to Order
- The attribute ticket-id is a foreign key to SupportTicket

Favorite(username, restaurant-id)

- The attribute username is a foreign key to Customer
- The attribute restaurant-id is a foreign key to Restaurant

Contact(username, phone-number, name)

- The attribute username is a foreign key to Customer
- The attribute phone-number is a foreign key to Phone

RestaurantContact(restaurant-id, phone-number)

- The attribute restaurant-id is a foreign key to Restaurant
- The attribute phone-number is a foreign key to Phone

Owns(restaurant-id, username)

- The attribute username is a foreign key to RestaurantOwner
- The attribute restaurant-id is a foreign key to Restaurant

CreateReview(review-id, username)

- The attribute username is a foreign key to Customer
- The attribute review-id is a foreign key to Review

RequestForDelivery(username, order-id, acceptance)

- The attribute username is a foreign key to DeliveryPerson
- The attribute order-id is a foreign key to Order

DeliveredBy(order-id, username)

- The attribute username is a foreign key to DeliveryPerson
- The attribute order-id is a foreign key to Order

DeliveredTo(order-id, address-id)

- The attribute address-id is a foreign key to Address
- The attribute order-id is a foreign key to Order

User Interface Design and Corresponding Statements

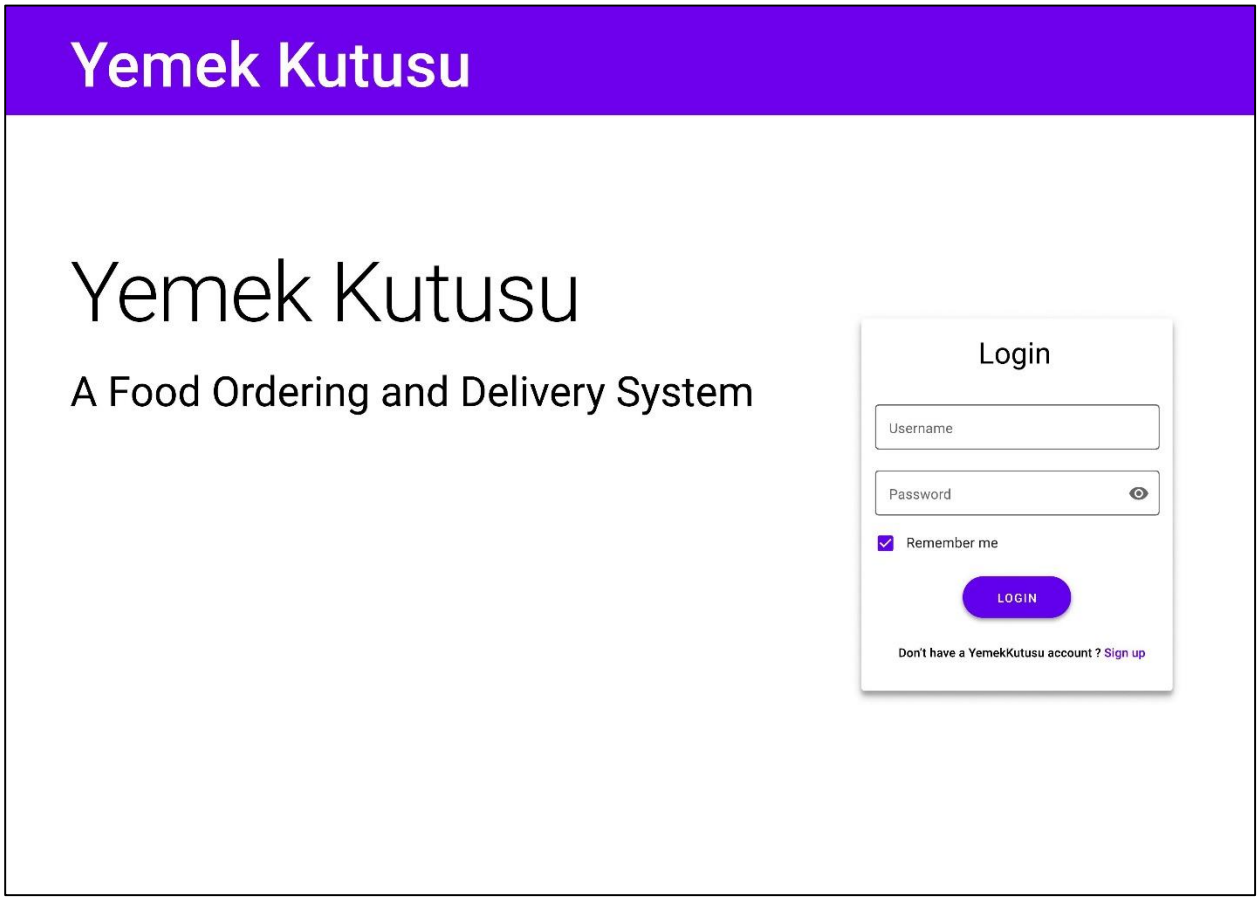
Login Page

Inputs: @username, @password

SQL Query:

SELECT count(*)
FROM Users U
WHERE U.user-name == @username AND U.password = @password;

User Interface:



Signup Page

SQL Query:

Inputs: @username, @first-name, @last-name, @birthdate, @email, @password

Shared for all users:
INSERT INTO Users VALUES(@username, @first-name, @last-name, @birthdate, @email, @password)

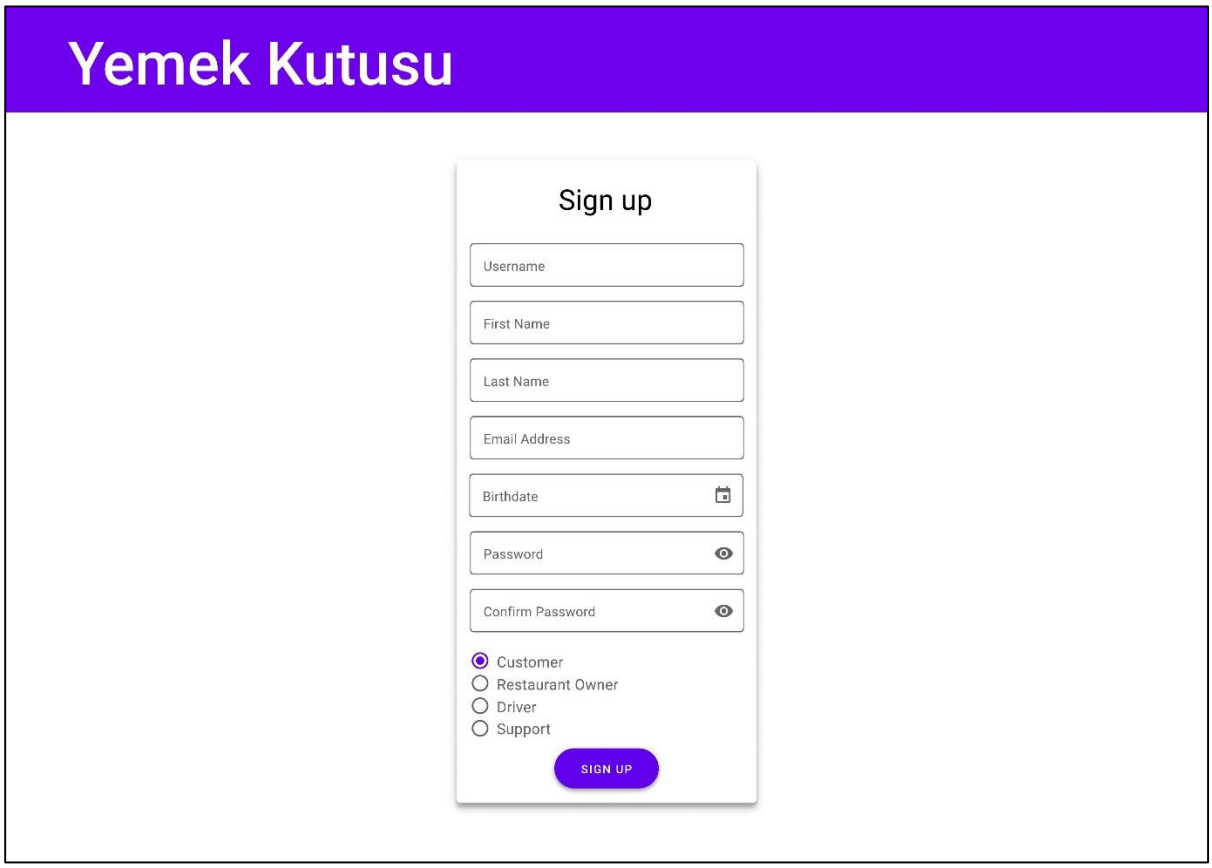
If user type is support:
INSERT INTO SupportStaff VALUES(0, true)

If user type is restaurant owner:
INSERT INTO RestaurantOwner VALUES(0)

If user type is customer:
INSERT INTO Customer VALUES(0)

If user type is delivery person:
INSERT INTO DeliveryPerson VALUES(NULL, false)

User Interface:



Homepage

Input: @username

SQL Query:

SELECT username, first-name, last-name, birthdate, email

FROM Users

WHERE @username = username;

If user type is customer:

Display Credit:

SELECT credit

FROM Customer

WHERE @username = username;

Display Addresses:

SELECT *

FROM Address NATURAL JOIN HasAddress

WHERE username = @username;

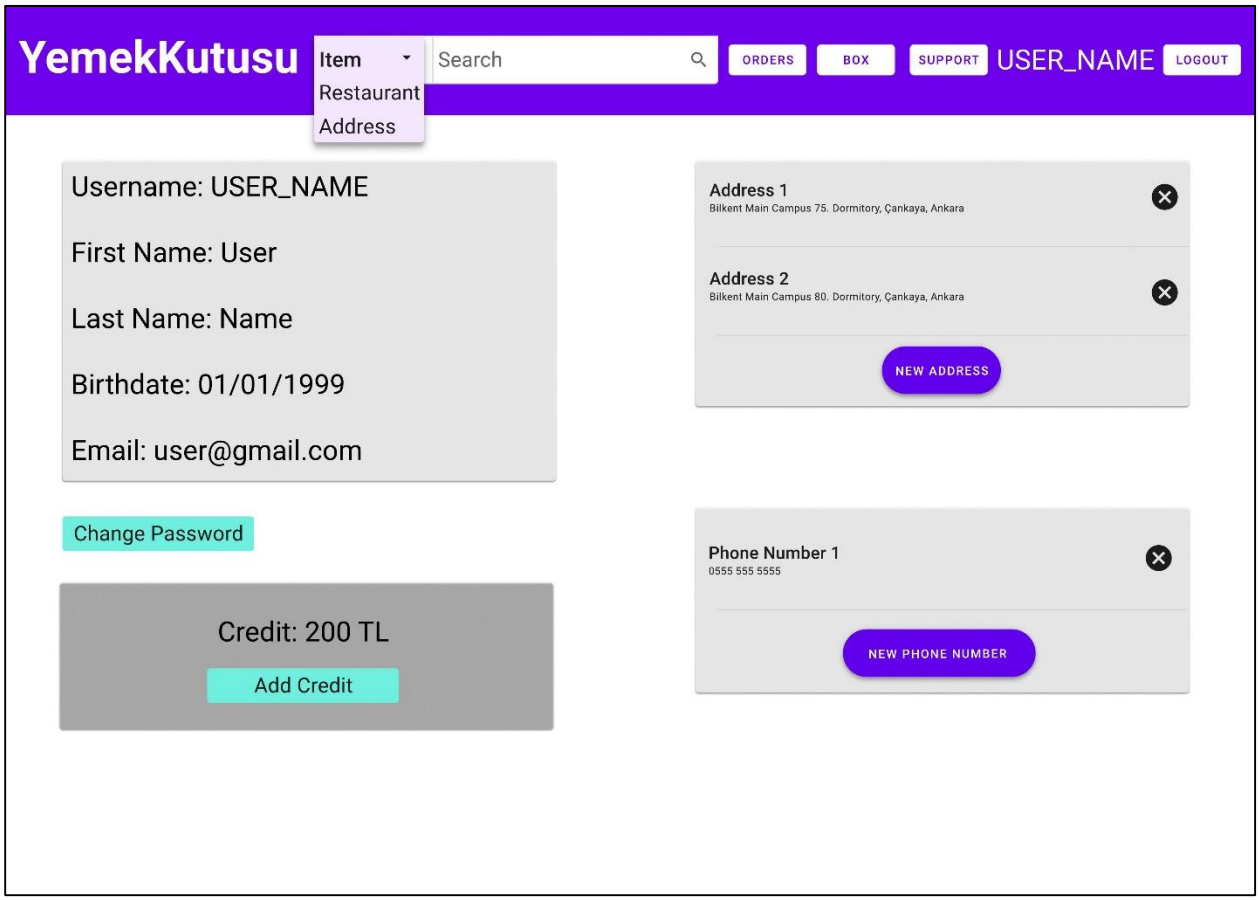
Display Phone Numbers:

SELECT *

FROM Phone NATURAL JOIN Contact

WHERE username = @username;

User Interface:



Search an Item

Input: @search_keyword

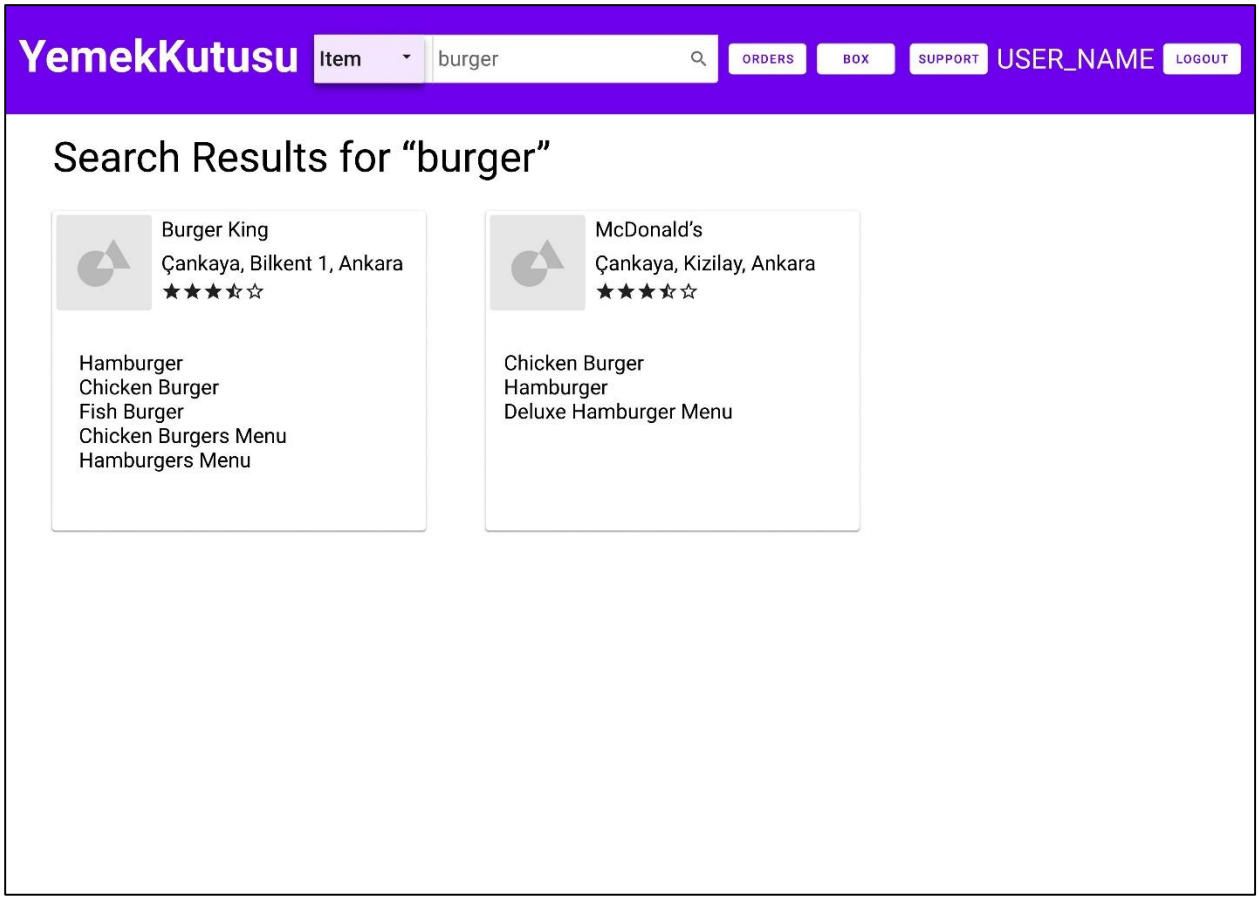
SQL Query:

SELECT R.restaurant-id, R.name, R.rating, Add.street, Add.street_number, Add.street_name, Add.apt_number, Add.city, Add.county, Add.zip, Or.orderable-name

FROM Restaurant R, offers Of, Orderable Or, locatedAt Loc , Address Add, Item I, Contain Con

WHERE R.is-open = TRUE AND R.restaurant-id = Of.restaurant-id AND Or.orderable-name = Of.orderable-name AND I.name LIKE '%@search_keyword%' AND Loc.address-id = add.address-id AND Loc.restaurant-id = R.restaurant-id AND Or.orderable-name = Con.orderable-name AND Con.item-id = I.item-id

User Interface:



Search a Restaurant by Name

Input: @search_keyword

SQL Query:

SELECT Res.name, Res.average-rating, Res.is-open

FROM Restaurant Res,

WHERE Res.name LIKE '%@search_keyword%'

Search a Restaurant by Address

Input: @search_keyword

SQL Query:

SELECT Res.name, Res.average-rating, Res.is-open

FROM Restaurant Res

WHERE add.street LIKE '%@search_keyword%' or add.street_number LIKE '%@search_keyword%' or add.street_name LIKE '%@search_keyword%' or add.city LIKE '%@search_keyword%' or add.apt_number LIKE '%@search_keyword%' or add.county LIKE '%@search_keyword%' or add.zip LIKE '%@search_keyword%'

View Restaurant Menu

Input: @restaurant_id

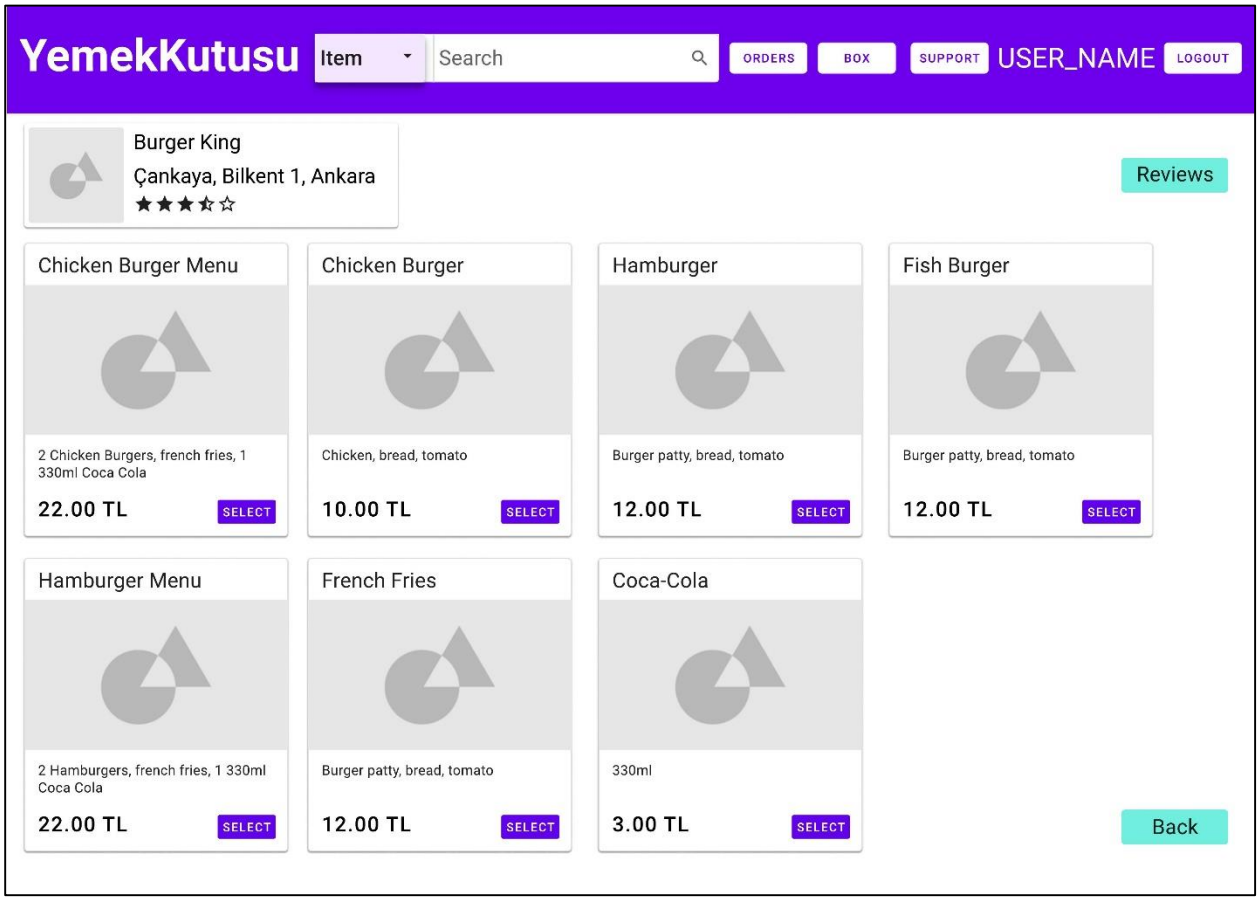
SQL Query:

SELECT O.orderable-name, O.price, Con.quantity, I.name, I.content

FROM offers Of, Orderable O, contain Con, Item I

WHERE Of.restaurant-id = "@restaurant_id" AND O.orderable-name = Of.orderable-name AND Con.orderable-name = O.orderable-name AND I.item-id = Con.item-id

User Interface:



View Options

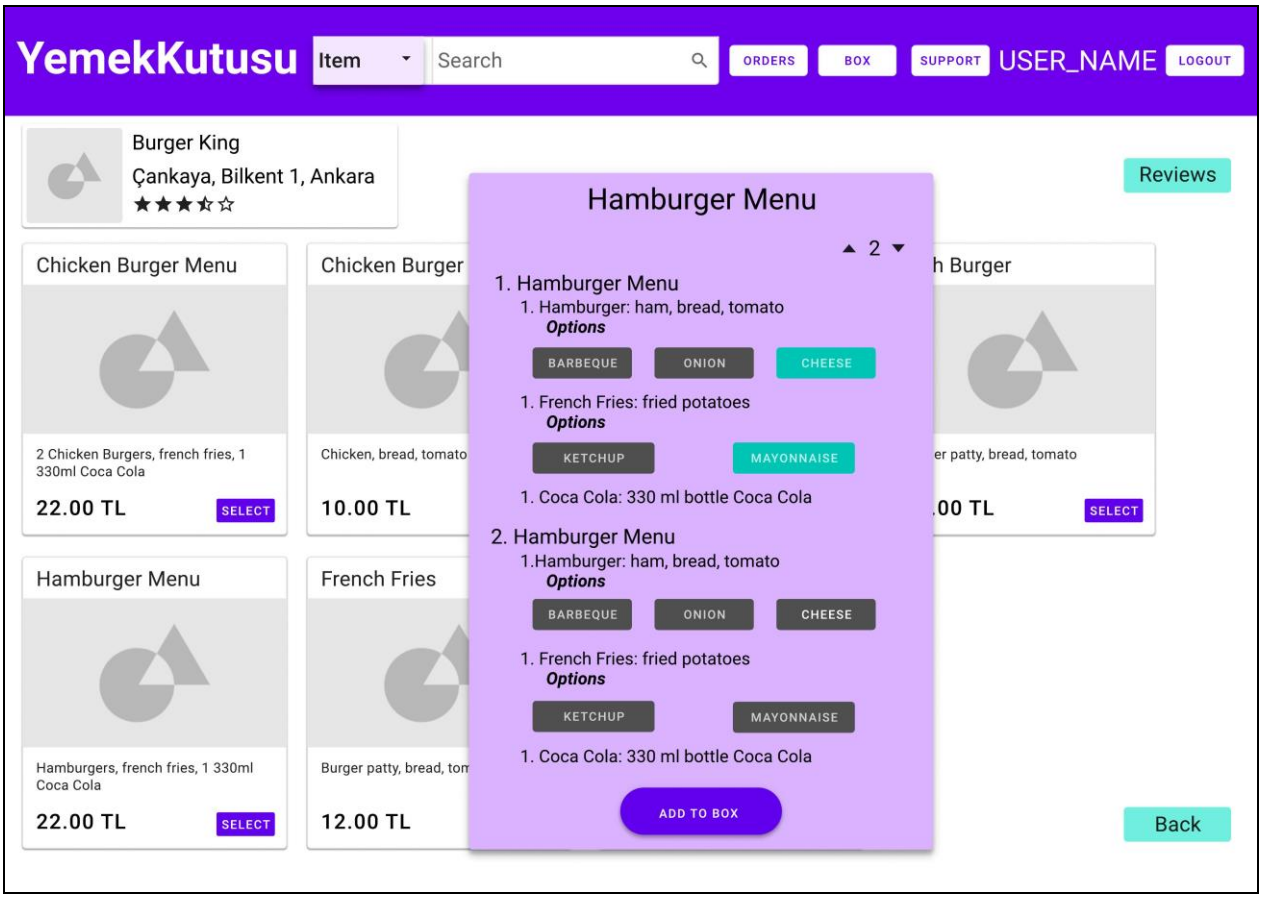
SQL Query:

SELECT I.name, Op.name

FROM Item I, Option Op, hasOption has, contain Con

WHERE Con.orderable-name = @orderable-name AND Con.item-id = I.item-id AND has.item-id = I.itemId AND Op.name = has.name

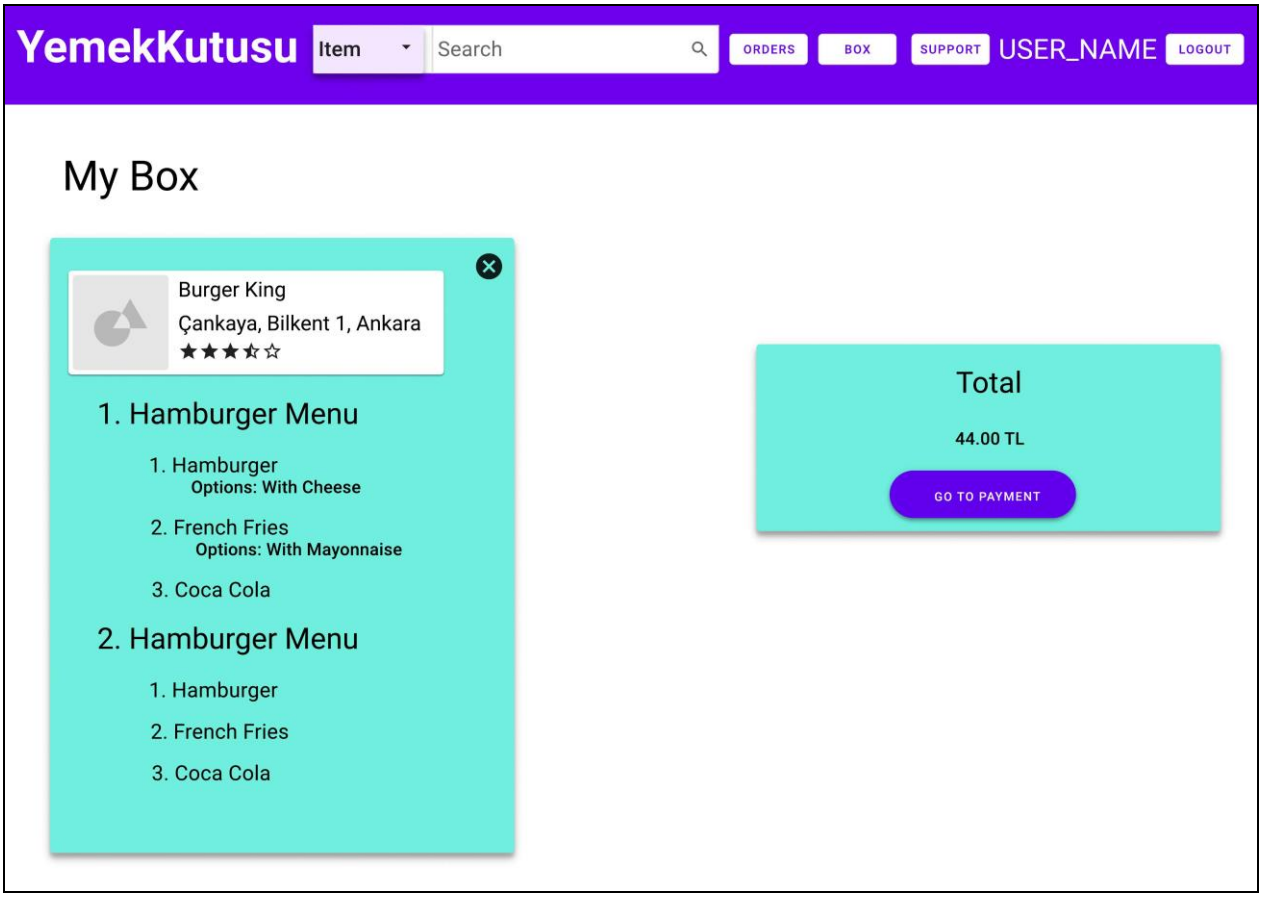
User Interface:



List Box of a Customer

No SQL query is needed for this functionality because the order is not completed yet. The customer is just listing her/his box.

User Interface:



Make Order

Inputs: @user-name, @order-id @orderable-name, @restaurant-id, @quantity, @time, @has-plastic, @note, @item-id, @option-name, @item-index, @exists

SQL Insertion Statements:

These insertions will be in a loop for each ordered menu, each item, each option. Parameters for such cases are for only one menu, item or option.

INSERT INTO ConsistOf VALUES(@order-id, @orderable-name, @restaurant-id, @quantity)

This query is for getting the item-id s of an orderable menu.

SELECT C.item-id

FROM Contain C

WHERE C.restaurant-id = @restaurant-id and orderable-name = @orderable-name

INSERT INTO Specify VALUES(@item-id, @option-name, @order-id, @restaurant-id, @orderable-name, @item-index, @exists)

INSERT INTO Order VALUES(@order-id, "Prepared", @time, NULL, @has-plastic, @note)

INSERT INTO CompleteOrder VALUES(@order-id, @username, @restaurant-id)

SQL Query:

SELECT add.street, add.street_number, add.street_name, add.apt_number, add.city,

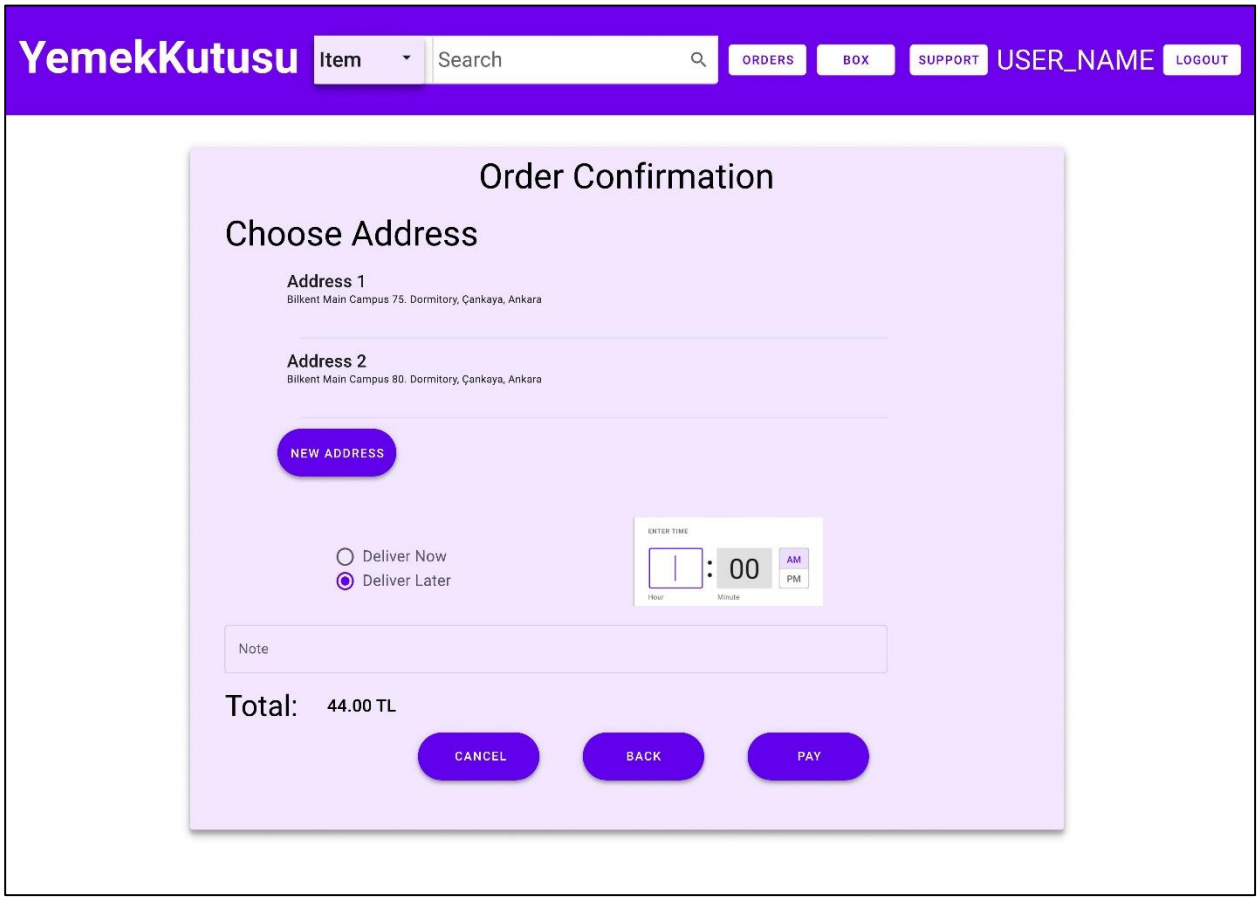
add.county, add.zip

FROM HasAddress has, Address add

WHERE has.username = @username AND has.address-id = add.address-id

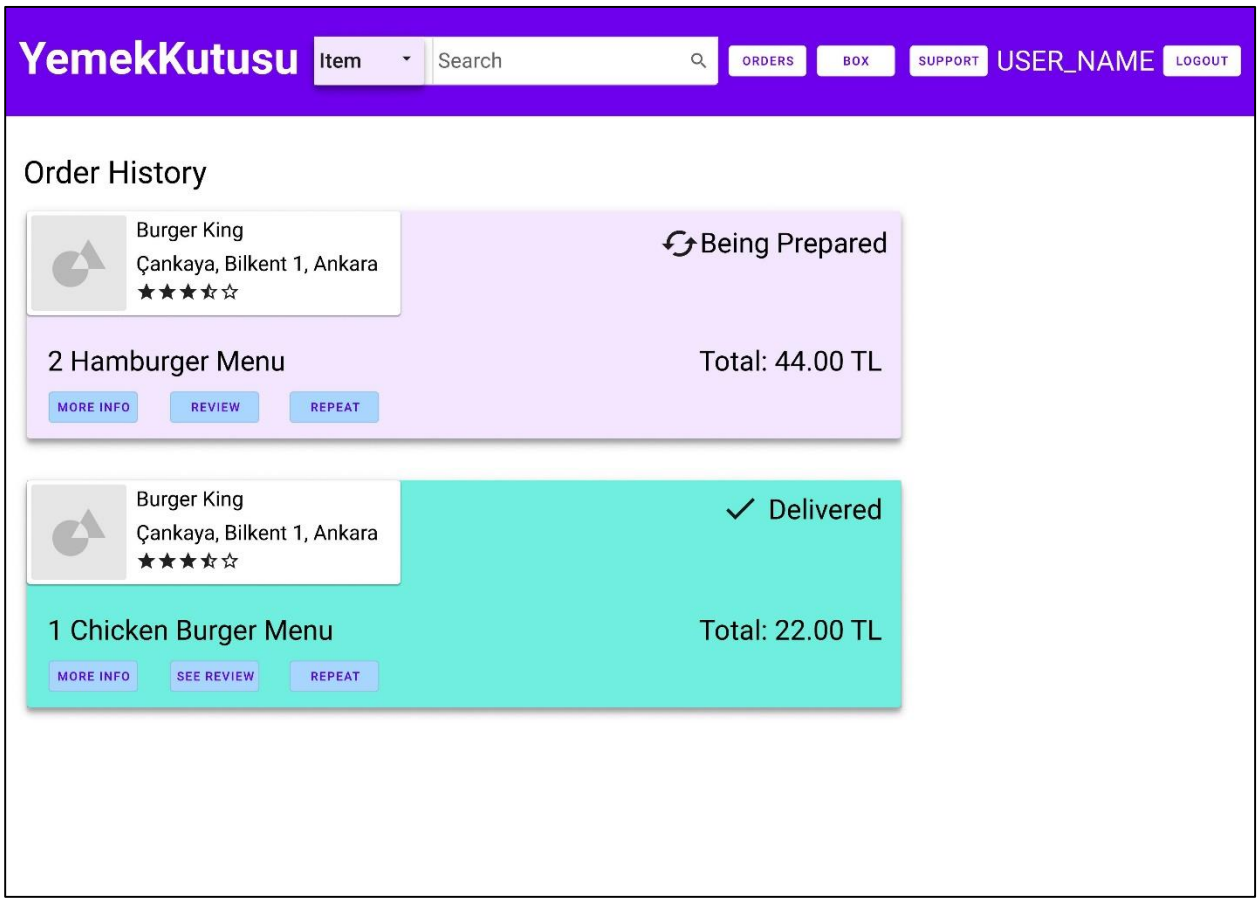
Order Confirmation

User Interface:



List Orders of a Customer

User Interface:



Check Details of an Order

Inputs: @username, @order-id

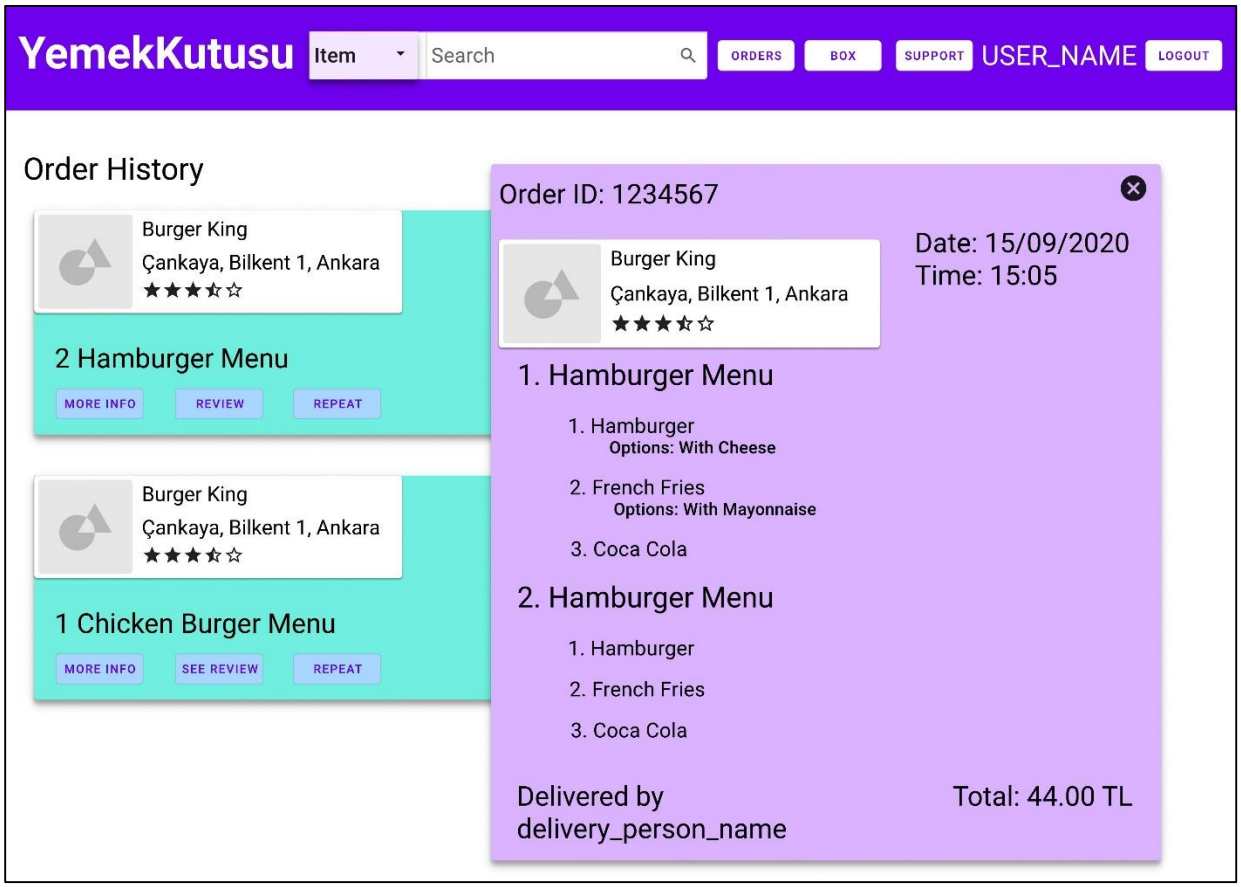
SQL Query:

SELECT Ord.name, O.order-time,R.name, add.street, add.street_number, add.street_name, add.apt_number, add.city, add.county, add.zip, R.rating, Ord.name, I.name, Op.name, Del.username

FROM Order O, Orderable Ord, ConsistOf Cof, Contain Con, CompleteOrder Ord2, Restaurant R, LocatedAt Loc, Address add, Item I, Option Op, HasOption hasO, DeliveredBy Del

WHERE @username = Ord2.username AND @order-id = O.order-.id AND Cof.order-id = O.order-id AND Ord.orderable-name = Cof.orderable-name AND R.restaurant-id = Ord2.restaurant-id AND Loc.restaurant-id = R.restaurant-id AND Loc.address-id = add.address-id AND Ord.order-id = Cof.order-id AND Del.order-id = O.order-id AND Op.name = hasO.name AND I.item-id = hasO.item-id AND O.order-id = Ord2.order-id

User Interface:



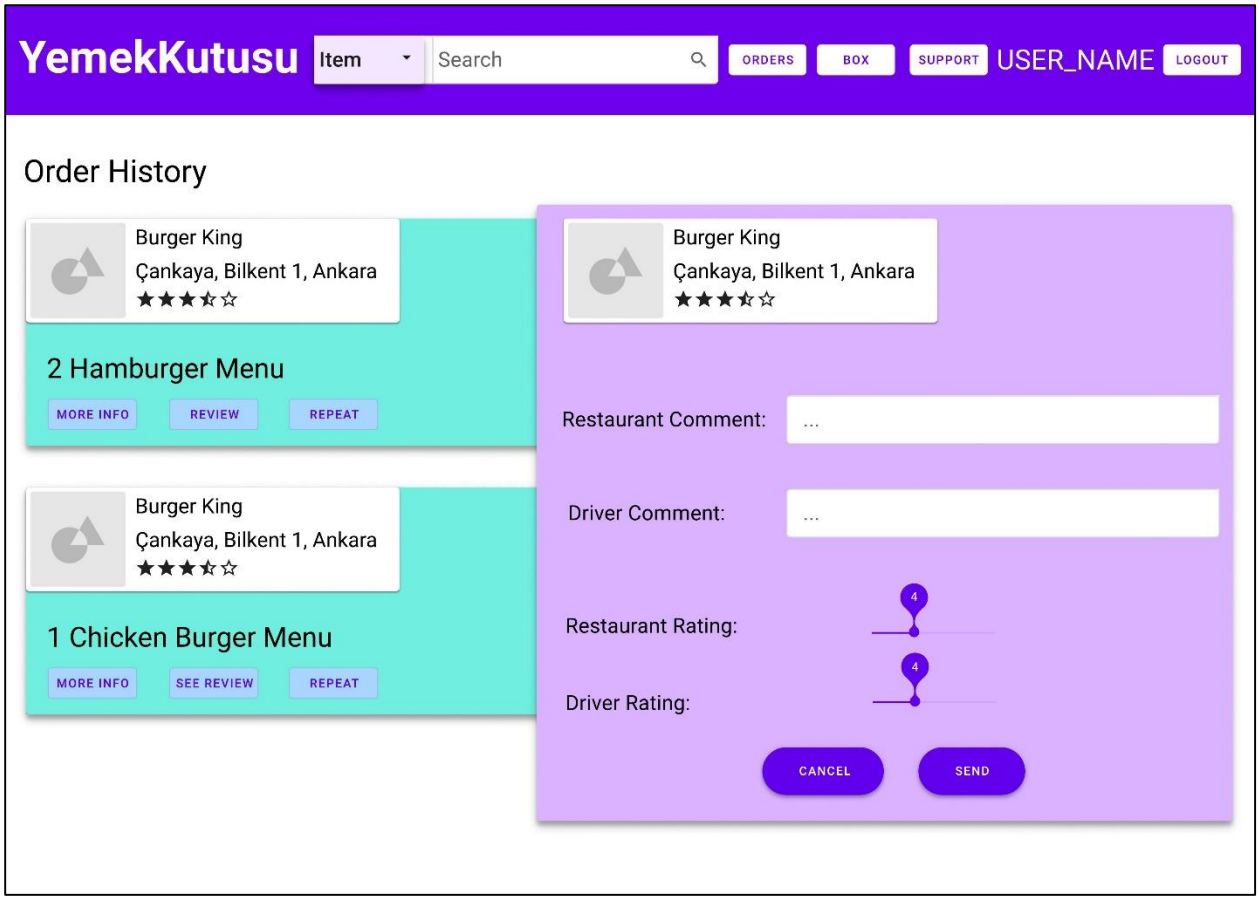
Write Comments for an Order

Inputs: @reviewID, @restaurantComment, @deliveryComment, @restaurantRating, @driverRating

SQL Insert Statement:

INSERT INTO Review
VALUES(@reviewID, @driverRating, @restaurantRating, @restaurantComment, @deliveryComment, NULL)

User Interface:



See the Comment Written for an Order

Inputs: @order_id

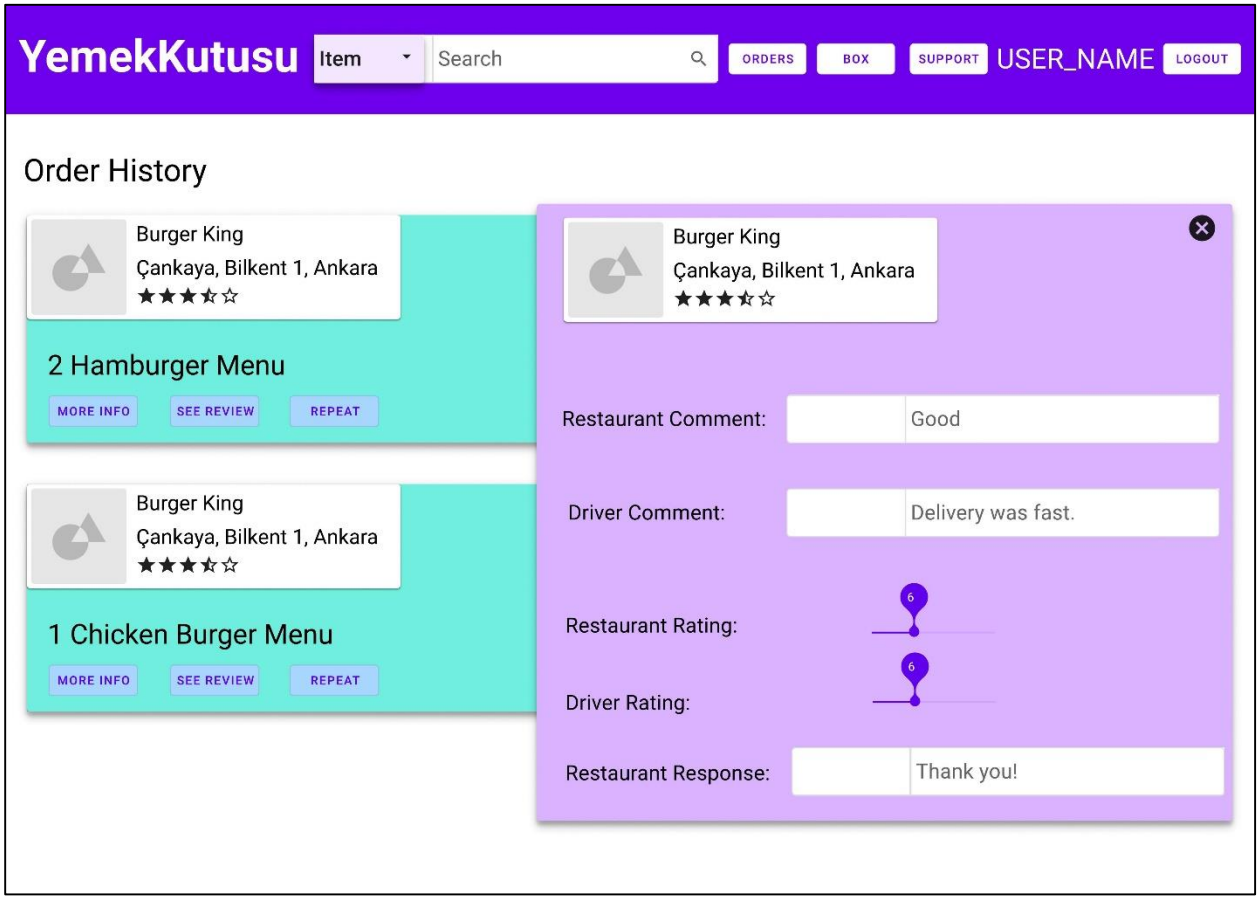
SQL Query:

SELECT R.name, R.average-rating, add.street, add.street_number, add.street_name, add.apt_number, add.city, add.county, add.zip, R.rating, Rev.restaurant-comment, Rev.driver-comment, Rev.restaurant-rating, Rev.driver-rating, Rev.restaurant-response

FROM Rated Rate, Restaurant R, LocatedAt Loc, Address add, Review Rev

WHERE @order_id = Rate.order-id AND Rev.review-id = Rate.review-id AND Loc.restaurant-id = R.restaurant-id AND Loc.address-id = add.address-id

User Interface:



List Reviews for a Restaurant

Inputs: @restaurant-id

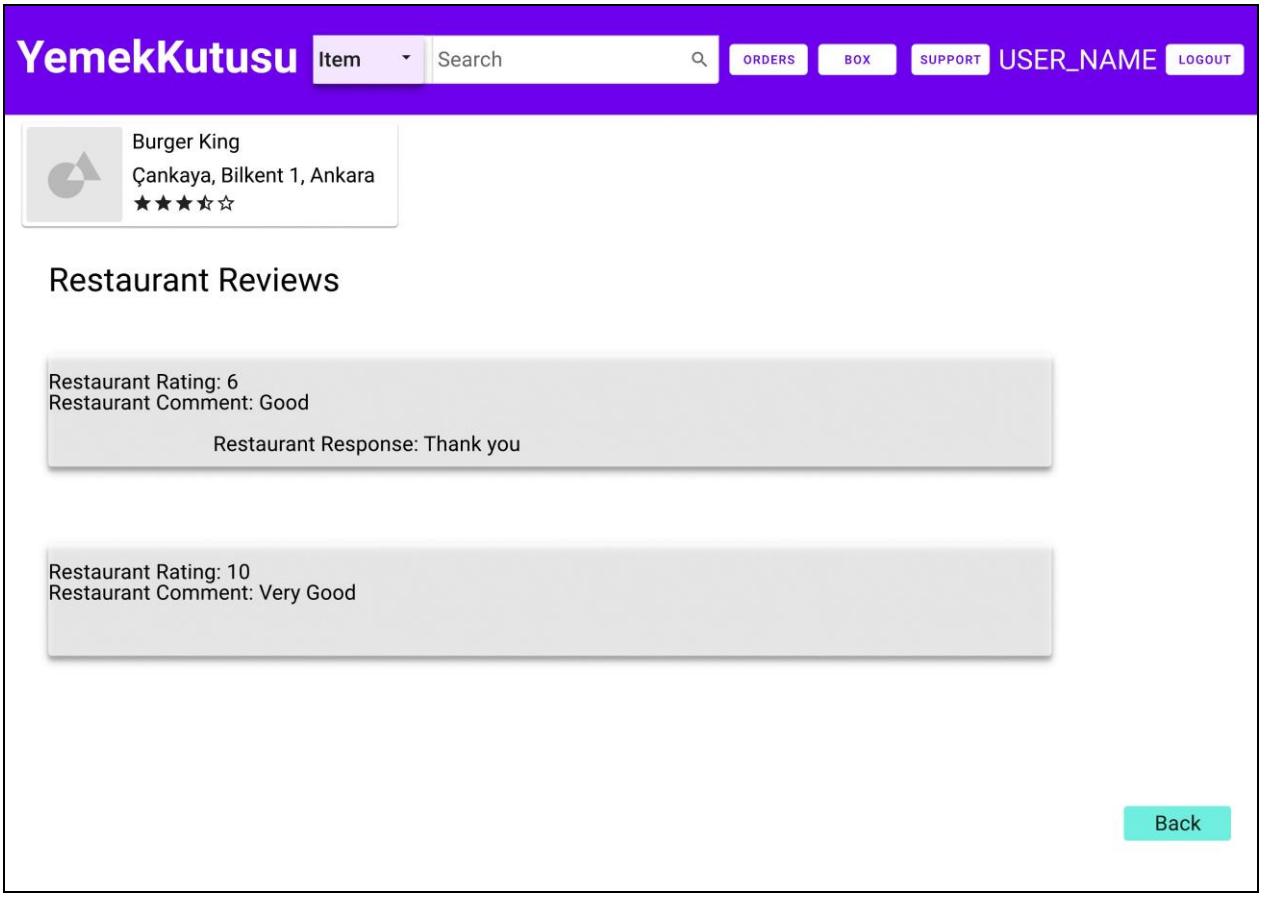
SQL Query:

SELECT Rev.review-id, Rev.delivery-rating, Rev.restaurant-rating, Rev.restaurant-comment,
Rev.delivery-comment, Rev.restaurant-response

FROM HasReview HRev NATURAL JOIN Review Rev

WHERE @restaurant-id = HasReview.restaurant-id

User Interface:



Additional Requirements

Additionally, customers of the food delivery system can submit a support ticket for any purpose which is specified by the “SupportTicket”. Therefore, two new entity types are defined which are “SupportStaff” and “SupportTicket”. Customers will be able to submit their “SupportTicket”s for any purpose. When the latter is submitted, the submitted tickets will be assigned to an available (free) “SupportStaff” and they will be responded by the “SupportStaff”. In case there is a violation of the rights of the customers is detected by “SupportStaff”s, “SupportStaff”s will be able to send a warning to the restaurant-owner. The system will take certain measures against the malicious behaviour of restaurants by inspecting their owners’ warning counts.

Send a Support Ticket

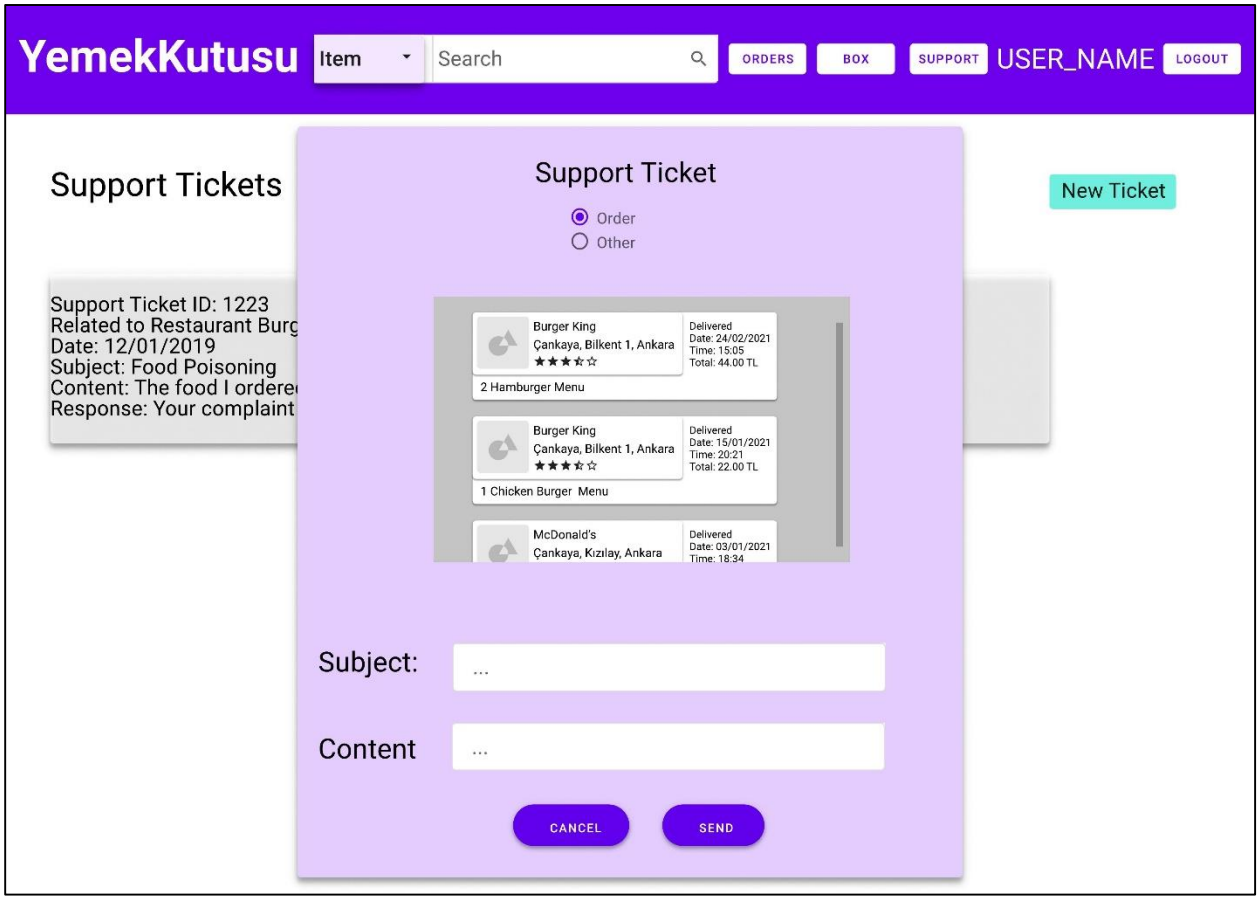
Inputs: @ticket-id, @date, @subject, @content

SQL Insert Statement:

INSERT INTO SupportTickets

VALUES(@ticket-id, @date, @subject, @content, NULL)

User Interface:



Respond to Support Tickets

Inputs: @response, @staff-username

SQL Update Statement:

UPDATE SupportTickets

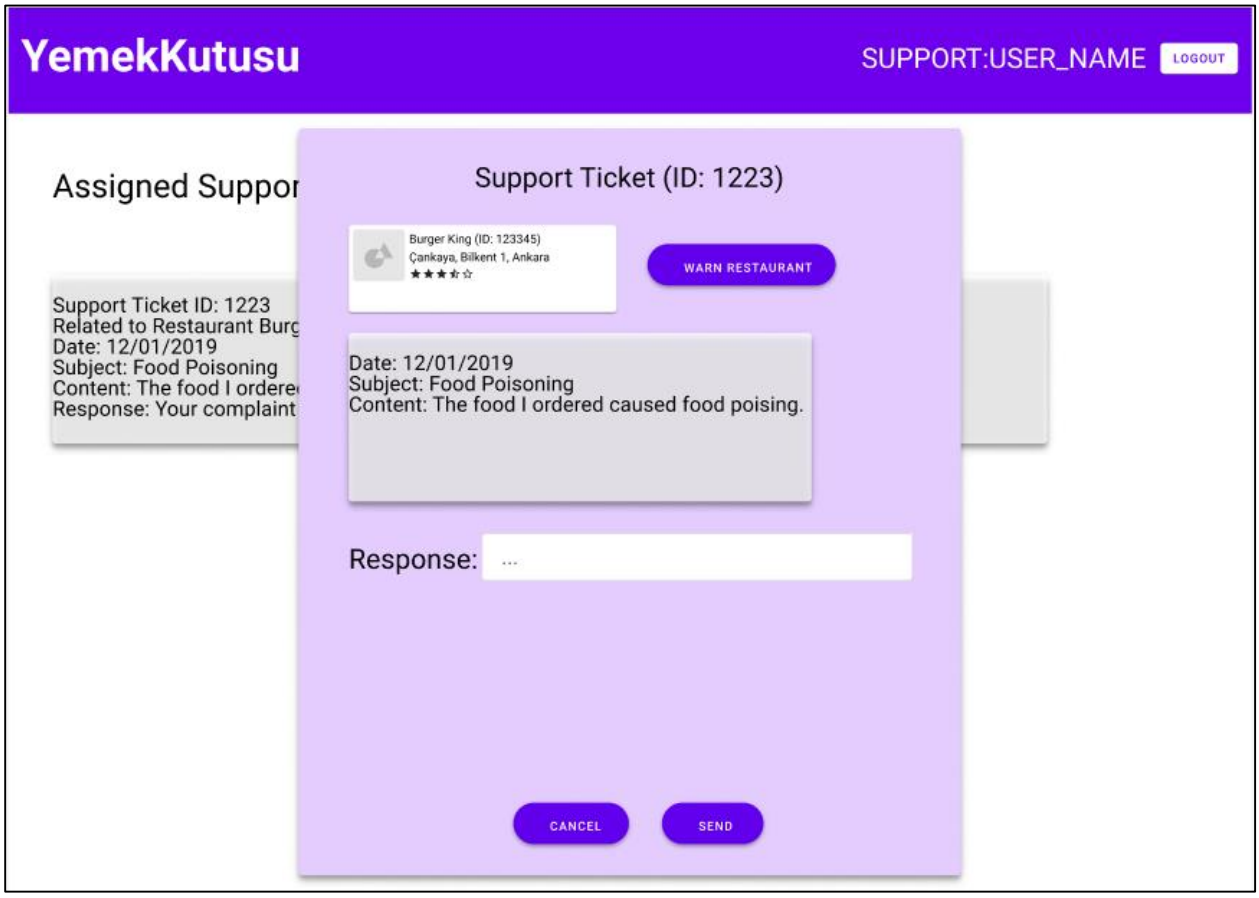
SET response = @response

WHERE EXISTS(**SELECT** *

FROM AssignedToTicket NATURAL JOIN SupportTicket

WHERE response = NULL AND username = @staff-username)

User Interface:



List Support Tickets from Customers Account

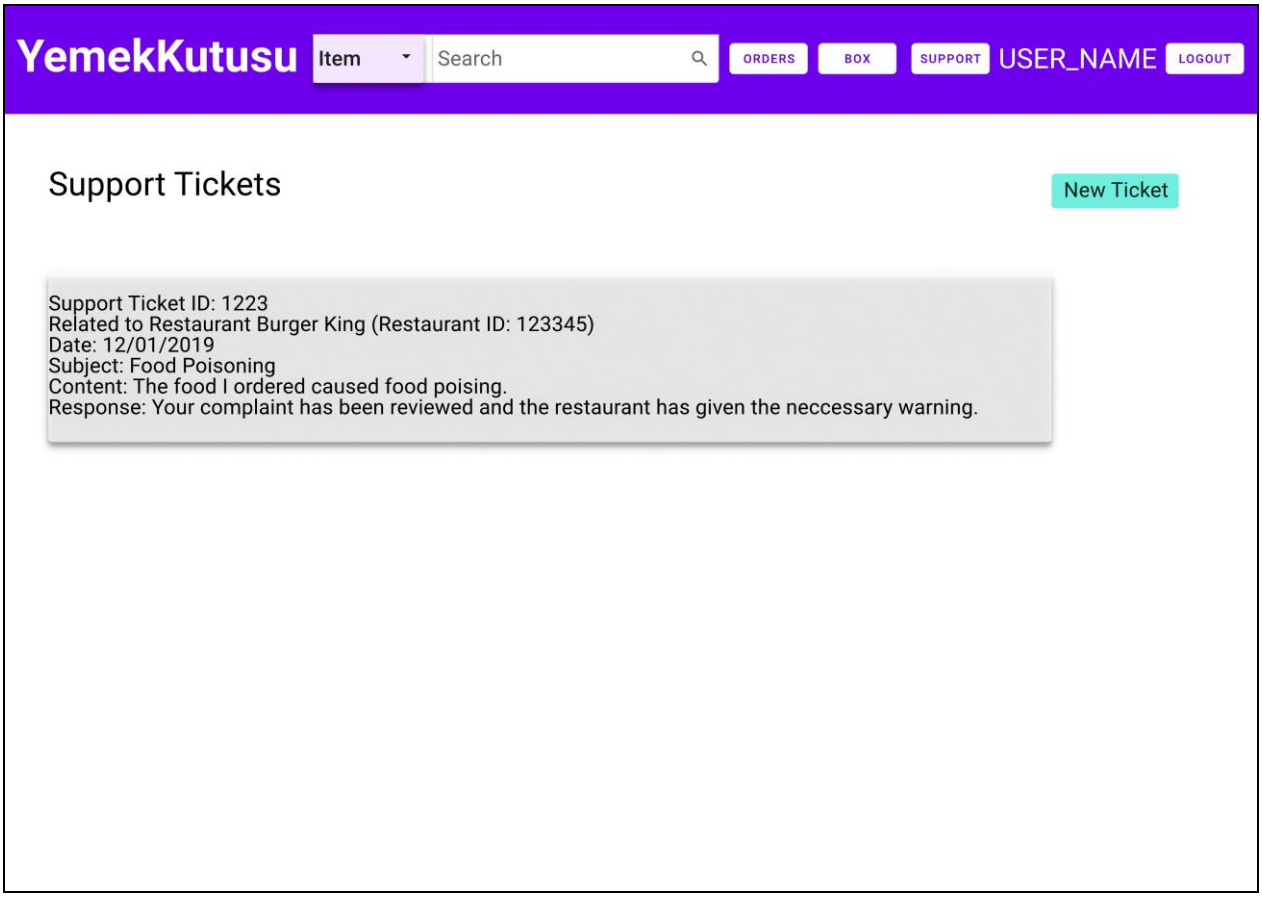
Inputs: @username

SELECT *

FROM SubmitTicket NATURAL JOIN SupportTicket

WHERE username = @username

User Interface:



List Tickets Assigned to Support Staff

Inputs: @staff-username

SELECT *

FROM AssignedToTicket NATURAL JOIN SupportTicket

WHERE username = @staff-username

Send Warning

Inputs: @owner-username

SQL Update Statement:

UPDATE RestaurantOwner

SET warning-count = warning-count+1

WHERE username = @owner-username

User Interface: