CS 353 Database Systems

Project Proposal

Group - 20

Group Members

Rüzgar Ayan

Kamil Kaan Erkan

Cankat Anday Kadim

Ege Türker

Table of Contents

Introduction

As defined by our guidelines, our group was selected to create a database application for an online food delivery platform. The goal was to create a 3 way system with the customer, the restaurant and the courier being able to interact with the system in various ways. One of the starting assumptions we made was that a courier could deliver orders from any restaurant as we assumed that the platform would connect the individual courier to the restaurant.

Description

A. Why use a database?

As described above our system will need to handle many users and their interactions with one another. Even the most basic functionality our system would offer such as ordering food from a restaurant, will require quite a lot of information being stored. As we are designing a food delivery platform, there will be a lot of traffic and the system will need to group the information and use it as necessary. In addition to the aforementioned requirements, the system will also need to quickly update itself in accordance with each user input. Therefore a database is the most efficient solution to our system.

B. How is it a part of the system?

In this system, everything related to information storage will take place in the database. The information about users such as names and addresses will be stored in the database. Besides that, information about functions such as reading and writing comments, ordering food and favoriting restaurants will be provided by the database.

Requirements

In order to use our system each user must have a registered account. User accounts can belong to four categories: customers, couriers, admins and restaurant owners. Each of these account types have different interactions with the system which will be implemented according to the functional requirements. This section includes functional and non-functional requirements of our system and also the limitations and constraints of it.

Functional Requirements

Any User:

can do the following:

- Register with an email and a password.
- Set a profile name.
- Change email and password.
- Change their profile name.

Customers:

can do the following:

- Place an order for now or for a later date.
- Cancel an order before it is delivered to the courier.
- Tip the delivery personnel or the restaurant independently.
- Review and make comments for the order.
- Search by food name
- Search by the amount they are willing to pay
- List the most popular restaurants and foods.
- Add restaurants to their favorites.
- Use discount coupons issued by the restaurants
- See the status of their current order.
- Choose how to pay for their order beforehand.
- See their old orders
- See the scores and read the reviews of restaurants and couriers

Couriers:

can do the following:

- Choose the regions that they want to deliver food to.
- Set the delivery fee for each region.
- Set their working hours.
- Share his live location.
- Decline to deliver an order.
- Set the status of an order

Restaurant Owners:

can do the following:

- Add food items to the menu or remove the items from it.
- Change the prices of the food items.
- Comment on the customer reviews.
- Start discounts with a certain percentage and starting-ending dates.
- Set opening and closing hours for the restaurant.
- Set the status of an order.
- Cancel an order.
- See the statistics (money gained per day, order amount per food item, etc.) about their orders using either graphs or charts.

Admins:

can do the following:

- Approves reviews.
- Approves the addition of new restaurants.
- Approves the addition of new couriers.

Non-functional Requirements

Performance

Since there are many similar services available, our service must be running smoothly to attract the users. The response times for any type of user (customer, courier or restaurant owner) should not exceed 1 second.

Supportability

As a website, our service should support different screen resolutions so that users will not have any problems accessing the service.

Reliability

The information of users must not be lost and data should not be corrupted in cases like server crashes.

Security

Private information of users must be secured and shouldn't be seen by the others. Some other information such as the addresses of the customers can only be seen by the couriers only during the delivery time.

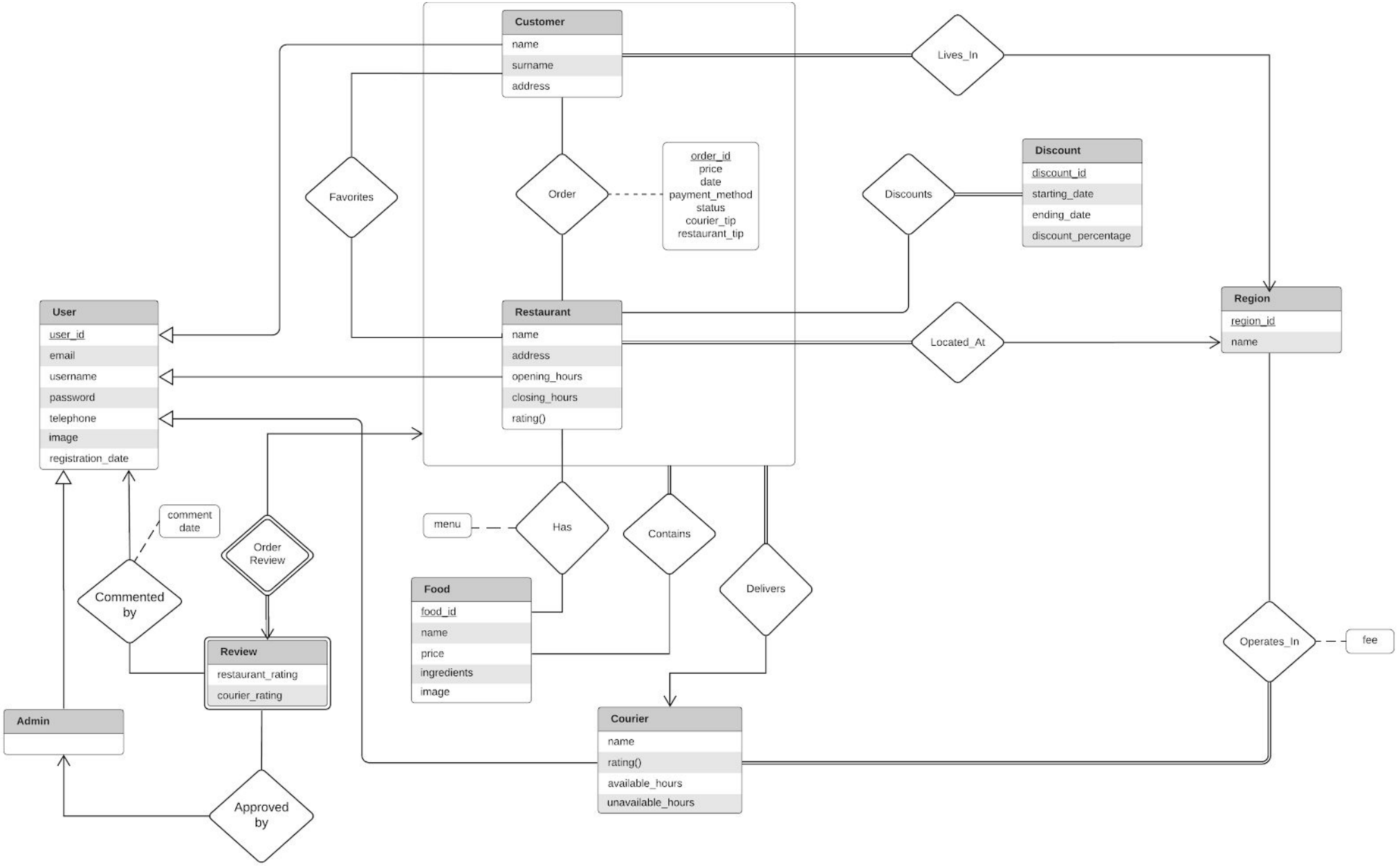
Constraints

- React.js will be used for the website design.
- PostgreSQL will be used for the database management system.
- Our reports will be published at https://scavbob.github.io/

Limitations

- Each user must have an email address and password to login to the website.
- User password must have at least 8 characters and contain at least one digit and one uppercase letter.
- Users can only rate and review the whole delivery process after the order is placed and delivered.
- Users cannot change their reviews once they have been submitted.
- Users cannot place orders from multiple restaurants in a single order.
- User reviews must be approved by the admins.
- Restaurant owners and couriers cannot see the personal information of users that post a review.

E/R Diagram


Customer
name
surname
address
Lives_In
Favorites
Order
order_id
price
date
payment_method
status
courier_tip
restaurant_tip
Discounts
Discount
discount_id
starting_date
ending_date
discount_percentage
User
user_id
email
username
password
telephone
image
registration_date
Restaurant
name
address
opening_hours
closing_hours
rating()
Located_At
Region
region_id
name
comment
date
Order
Review
Commented
by
menu
Has
Contains
Delivers
Food
food_id
name
price
ingredients
image
Review
restaurant_rating
courier_rating
Operates_In
fee
Admin
Courier
name
rating()
available_hours
unavailable_hours
Approved
by