CS 411 Project Reflection Report

Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

I think there are several changes in our final project.

1. In our original proposal, customers search for restaurants by city. But because the size of our dataset is limited the number of restaurants is too small if it's divided by city. In addition, since our restaurants are selected at random, sometimes some small cities will be selected, but some big cities may be ignored.
2. In our original proposal, we add a table named ingredient, from which we hope customers can know about what ingredients are the dishes made of. But later in practice, we find there are some problems. Firstly, the raw materials(ingredients) used in each restaurant are very different so it's hard and useless to form an ingredient table. Secondly, we find that restaurants may not be willing to disclose detailed information about the ingredients, such as manufacturers.

Discuss what you think your application achieved or failed to achieve regarding its usefulness.

I think our application has achieved that customers can know the rate, price, and restaurant of a dish, especially a rate, which is our creative point different from Yelp. We add a score on each dish so that the customers can clearly know what the recommended dish with a high score, and what's the dishes that may be not recommended by other customers with low scores. This enhances the customer's dining experience, thereby enhancing satisfaction with the application. On the other hand, restaurants can make adjustments to the dishes with low scores.

Discuss if you changed the schema or source of the data for your application

We changed the schema. The ingredients table and madeby table were dropped. This is because as we design the system, we thought displaying the ingredients for each dish is a necessary feature to help people avoid allergies and fit their diet preferences. However, in the actual implementation, we found the food data we have is limited because yelp only provides data relevant to the restaurant. With such limited data, the feature would be useless. So for the current version of our application, we decided to drop this feature.

As for the data for our application, we still use the dataset from Yelp as our main dataset. However, there is too much redundant information on Yelp, such as the friend list, the votes counts, and the evaluation counts, which is not used in our application. Hence, we only keep the needed attributes from the Yelp dataset.

In addition, we also need some other attributes which Yelp can't provide, such as those on the dishes table. As a result, we create data in the dishes table by ourselves.

Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

For the original design, we implemented a design with restaurants, the dishes they offer, within which the ingredients they have, as well as users and their comments based on each dish. Since yelp provides limited data, there's no need to keep the ingredients table. We consider it a redundant feature and we decided to drop the table. On the ER diagram, we decided to erase the part of the ingredients entity and the "Made by" relationship in the final design. Thus, the final design is more suitable for application in real life. Also, it is easier for us to make datasets in the updated structure.

Discuss what functionalities you added or removed. Why?

Based on the traditional design of the Yelp platform, we discovered that Yelp only pays attention to the ratings and recommendations of restaurants, but no certain dishes are considered in their system of comments and ratings. In this case, we aim at making a system of recommendations based on dishes, so that people who favor a certain dish can find the best in an area. Based on this idea, we added a relationship where comments are made on every dish, instead of every restaurant. Also, originally we decided that we would create a functionality that would display every ingredient of each dish so that people with certain allergies could filter out the dishes. After examining the datasets we are able to retrieve, we decided that this functionality would be dropped for a limited dataset and its uselessness.

Explain how you think your advanced database programs complement your application.

In the database design, we create triggers on the table of comments. The trigger makes sure that the rating is within the range of 0 to 5 before being inserted into the table. Also, a trigger of star check is implemented in restaurants, making sure that the star is in the range of 0 to 5. What's more, a stored procedure is created to help the system adjust the ratings of every restaurant, based on the rating of every dish. If a dish rating of the restaurant is lower than the average rating of this dish in each restaurant, it would cause the rating of the restaurant to drop and vice versa. This technique helps the system to adjust itself dynamically, not only by the data inserted, but also by the relations between relevant data.

Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

Since we did all of the work locally instead of putting it on the GCP, the first technical challenge that we faced was the installation of SQL. There are different versions so we have to make sure that we download the right version corresponding to our computer system.
The second technical challenge that we faced is figuring out how to use the SQL query that we write in the SQL workbench. It's not exactly the same as what we do in class practice, so we have to search online to figure out it.
The third challenge is each member of the team has a different technical background. And in web application development, everyone has their own preferences: java(spring boot), python(Django), and Javascript(nodeJS). Besides this, none of us have experience with front-end development. So finding a framework to work on and distribute the work accordingly is a hard time.
The fourth technical challenge is that stored procedures cannot be tested directly in SQL. Hence, we created a separate test database for the tables that stored procedures may impact. Considering that the related data is small, we can easily debug the application.

Are there other things that changed comparing the final application with the original proposal?

In the original proposal, we plan to use the user information from the yelp dataset to create user entities in our application. But during the implementation process, we found that there is too much redundant information in the yelp dataset such as the friend list, the votes counts, and the evaluation counts. In our proposal, we only want to make the user table store the username and password so that those additional attributes are unnecessary for us. Therefore, we choose to auto-generate the user information that we want.

Describe future work that you think, other than the interface, that the application can improve on

First of all, the current version of our application is very basic. When the user does a search action by the states, the output is a table with restaurant information and a link to the dishes of that restaurant. By clicking on the link, the user can access the ratings and the comments of all the dishes that the other users added to the restaurant. A little

improvement on it is that we can add the feature by using some existing APIs to let the users be able to upload images to the dishes. Also, we can add a feature to allow users to save their favorite dishes. Secondly, there are some new searches that we can implement. For example, let the users search the restaurant by dishes in all of the restaurants in the state. Users can see the ratings of restaurants with the dishes they want clearly. Thirdly, our rating adjustment method is too simple right now. We used a scale from 0 to 5, if a user gives the dish a rating that is below the rating of the restaurant, the ratings of the restaurant will get down directly. Also, if a user gives the dish a rating that is above the rating of the restaurant, the ratings of the restaurant will increase. We can add a sentiment analysis model here to interpret the comment of the user to get scores that the application used to adjust the ratings of restaurants. Also, adding this model can also help us to determine whether the user gives ratings that do not match their actual experience.

Describe the final division of labor and how well you managed teamwork.

The final division of labor in our group is pretty clear. We split the application into four parts(entity, controller, service, front-end) and everyone takes one part. Since we are all new to application development, we pretty much followed the video tutorial that we found online to construct the back end and front end. We help each other through the chat group. For every stage, we had two meetings, we split the work in the first meeting and combine the work together in the second one. If we find something inappropriate we directly talk to each other so that we can find the solution to the problem quickly. The communication in our team is very smooth, and all the members are kind and patient.

Demo Video:
https://drive.google.com/file/d/1mcGfEsMdRQW8mWJV3OJ3W7sz-X7yE9N4/view?usp=sharing