Stage 6 Final Report Group 39

1. **Please list out changes in your project directions if the final project is different from your original proposal. Are there other things that changed comparing the final application with the original proposal?**

   Our final product has almost the same functionalities as proposed in the original proposal. We also never changed our goals or directions. There were 2 minor changes in terms of features, and they will be further discussed in later sections:

   1. We proposed to ask users to add tags (e.g. spice lovers) to their userprofile and we will make recommendations based on these tags. In the final implementation, we changed to asking users to add favorite foods (e.g. tacos), "preferences", out of the consideration that food items are more straightforward and will correspond better to the menu of each place to eat.
   2. Another change was an additional functionality: the search button, in addition to the suggest button. This button allows regular search with keywords and therefore gives users more flexibility when looking for places to eat.

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

   Our application will be useful for those needing a quick and legit decision on where to eat right now. This was achieved by successful implementation from both back and front-end. We were able to store, gather, and make use of both places and users' data in the backend. On the front, we allow users to add information as they wish and receive suggestions for food places to go to, as well as search for places with keywords to get ideas. We also used a recommendation algorithm (Neural Collaborative Filtering) to make better suggestions to the user so that it is more likely that they'll take our advice.

   Despite the various features that we achieved, there are also aspects that we did not have time to get to. For example, we did not embed any maps in our website, so it may not be the most convenient if users need to copy the restaurant address into another app to get directions.

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

   For schema, we added price level, ratings, number of ratings, and addresses for restaurants, cafes, and bars to represent the characteristics of these places, so we can more easily match them with the preferences of the customers. We also added additional order tables for each type of place so that the recommendation can be made based on this extra information.

   As for the source of data, although we were able to collect real data on places by scraping from Google maps, we were not able to obtain real information on user profiles. Therefore, we manually generated fake data for tables related to users instead of collecting real ones.

4. **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?**

We slightly modified our ER diagram a little at stage 3. For example, we changed the primary key in the Customers table from an arbitrary id to the username since we will require that a username not be used repeatedly on our website. We also changed the relationship between Customer and Credentials to be a one-to-one relationship, and separated out the Orders and Serves relation for the 3 types of places we have. These changes are all for more straightforward and easier implementation later on.

We also added a few additional features. As mentioned in Q3, we added a few additional fields to each place we included. This is real data including aspects such as price-level and ratings from Google Maps. This data is simply there to allow better or more personalized recommendations based on this additional information. Our final implementations then closely followed the updated diagram. There are admittedly, more improvements to be made, but our updated version seems suitable for our purposes and turned out to be working well.

5. **Discuss what functionalities you added or removed. Why?**

As mentioned and explained in Q1, we swapped out the tags to preferred food items for better correspondence with the menu items. We also added a search function in addition to the suggestions, for a better user experience. We also added a few operations, such as changing passwords, deleting accounts, etc., that users can perform to manage their accounts, and these were not covered explicitly in the initial proposal. These are all necessary functionalities relating to user accounts, so we added them to the expandable menu on the website.

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

Our advanced database programs implemented at stage 5 include a trigger and a stored procedure. The trigger can smartly decide if the user has additional preferences based on the user's order history. A new entry will be inserted into the favorites table if the user orders the same food more than 3 times. This way, we can learn and update a user's profile in real time as they use our website. The SP is used to find candidate places for suggestions with a specific user input. Since we will be repeatedly extracting such information for recommendations for a user, we created a SP to make things easier.

In stage 3, we also developed two advanced queries that helped our database. One was for a complicated search with a specified type of restaurant combined with the users' profile. This was built for more personalized search results. The other query was built to help with our purpose of extracting the summary report on our users. We can both directly check in our work bench to

easily see the records of users and restaurants and acquire a few summary statistics. The query was for pulling users who ordered food that was out of their comfort zone, reflecting whether they took our advice and tried something novel. These queries will demonstrate the influence of our website and motivate us to enhance our application.

7. **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 maintain your project.**

   Zhenan Shao: One technical challenge we encountered was during the data collection process. The specific information about restaurants that we need for our app seems to be hard to extract automatically. For example, on google maps or yelp, the maximum search results for a location is 200, meaning that a lot of restaurants were omitted from the results and there was no way to easily get 1000 non-repeating results. This prevents easy scraping using the html information. The way that we coped with this was to first pull from government public databases where a roster of restaurants was stored, which is usually used for health inspection purposes. Then we used the Places API from Google for scraping data using these names. However, note that the Places API is limited in the amount of information it can give you. For example, the menu information cannot be obtained easily, which is why we had to generate bogus food item data.

   Ben: We have several different entities in our database model. Sometimes it is necessary to label them distinctly different in the column name like in the OrderBar table there is a bar_id a food_id but in the context of the application sometimes we are looking at a set of entities and only selecting a single member from either of those entity sets. To avoid mixing up keys in query result dictionaries it can be beneficial to alias query result column names with the SQL “AS” keyword. Such as “SELECT res_id AS id…” so that when passing the data to be rendered with a template the page can expect just data[“id”] and not have to worry about it being res_id, bar_id, cafe_id, or some new type of place since our design is flexible in this way.

   Yuanfeng Li: The test data of the model actually didn’t turn out to be as good as expected. Later I realized that it is probably due to the fact that the key information used for recommendation, the user behaviors are essentially randomly generated so testing with such data does not even make sense. Hopefully the effectiveness for our chosen model has been tested in lots of other similar scenarios so we still can use it in our application.

   Yanzhen: When writing the algorithm using data from the database, we found we still need to make another page in our website to display the result generated by the algorithm. Therefore, I added another html file to create a result to test the algorithm, and then delete that file. However, it is more simple to first export the data to a csv, test and fix the algorithm, and then incorporate the whole algorithm into the website structure.

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

We can improve on our recommendation algorithm. Right now we represent each users' preferences as a vector that has a length of the number of restaurants, which could become a burden if we include a lot more restaurants. It is possible to use another neural network that stores restaurants' id in the vector instead of just storing 0 and 1 into the vectors. It is also possible to use a network that keeps up with the updating and deletion of the record.

In addition, we could use a noSQL database that stores the dynamic components of our database such as: favorites, orders and perhaps comments that we can display when we make a new recommendation.

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

We all worked together, with timely communication and supporting environment, to present this final version of the web application. The written documents at each stage were all a collaborative effort, and here is a brief list of tasks we are responsible for during the implementation of our web app:

Zhenan Shao: data collection, building SQL databases/queries, frontend formatting.

Ben: main page frontend with place keyword query functionality, fetch place details from DB and render page, user management pages: preference settings, adding user order histories to database, fetching user order histories from database page, logout.

Yuanfeng Li: research on neural collaborative filtering algorithm, build according model, train and tune model.

Yanzhen: Some backend features, data processing function for recommendation algorithm