Stage 6: Project Report

Team: Q-Team124-triggered

Ruchika Biswas (rbiswas4@illinois.edu)
Thihan Moe Kyaw (tkyaw2@illinois.edu)
Avani Puranik (avanip2@illinois.edu)
Serena Tzeng (stzeng2@illinois.edu)

1. **Please list out changes in directions of your project if the final project is different from your original proposal(based on your stage 1 proposal submission).**
   For the most part, our group was able to complete the functionality we had laid out in our project proposal from Stage 1. One main piece of functionality that we deemed infeasible after looking through our data was searching recipes by recipe attributes such as cuisine and preparation time. Another piece of functionality that we did not implement was editing a recipe that has already been added to the database by the user. These were the two main features mentioned in our proposal that were left out of our final product. Additionally, our recommendation system was intended to be also based on what ingredients a user had available, but we decided to remove this aspect and base it solely on recipes they had rated highly without taking any input from them.

2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**
   Our application was executed extremely well based on what the intended goal of the project was at the time of the proposal. We had hoped to create a customized experience where users could search for recipes based on their own criteria, create their own recipes to share with other users, and provide their feedback on recipes they tried. From all accounts, our application was very useful in providing users with a holistic experience with everything related to sharing, searching and rating recipes to be able to expand their horizons and try new recipes. Additionally, the feature we implemented to generate recipes for users based on their determined preferences was especially useful as it could provide direction to users who may not have a specific recipe in mind.

3. **Discuss if you changed the schema or source of the data for your application.**
   The dataset we had initially looked at during the beginning stages of the project when selecting our topic was the one we used for our database in the final product. We did have to do quite a bit of data processing and cleaning using Python, however, since the data had lots of entries and was not formatted very well for entry into MySQL. Out of the 72,000 entries in the dataset, we took the top 2,000 recipes for use in our database

and removed all of those that had problematic entries in the various fields available. In terms of the schema, we generally kept the relationships between the various tables the same, but ended up removing specific tables after getting further into our implementation, which we discuss in Question 4 as well. Additionally, we removed foreign key constraints from the recipe tags and recipe ingredients due to the removal of tables from our overall database. We also changed some little things, such as adding a column to our users table for password/authentication and removing creation_date from the recipes table. Outside of just changing the size of VARCHAR types while adding data into our tables, these were the only structural changes we made to the schema during development.

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?**
   In terms of the ER diagram and table implementations created as part of Stage 2 of the project, we had planned to implement a recipes table to hold all the recipe information, two ingredients tables and two tags tables, where for both of these sets of tables, one would hold just the name of the ingredient/tag and the other would connect the name of the ingredient/tag to a specific recipe. Our thought process for having this was so that implementation would be easier if users wanted to update the names or other information about an ingredient/tag. However, we removed one table for both of these and simply had them connect to the recipe during our development, since we decided to not allow users to update this information after inputting it initially. Additionally, the schema correlating to recipe collections feature was not implemented since we did not have time to implement this additional functionality due to lack of time. In the project proposal, however, we had listed this an optional feature, so we focused on our main implementation instead. A table that we ended up adding was one to store the average ratings of each recipe, because this would make it easier to implement the recommendations feature we created. We made some little structural changes as we better understood our application, such as adding a column to our users table for password/authentication, removing creation_date from the recipes table, and changing the size of some VARCHAR types. Based on what we wanted to accomplish for our final product, the final design was more suitable as it was easier to work with and served the purpose it needed to without overcomplicating the database.

5. **Discuss what functionalities you added or removed. Why?**
   Outside of the functionality mentioned in Question 1, where we did not implement search by recipe attributes such as cuisine and preparation time, as well as the ability to

edit a recipe that has been created by a given user, we implemented all functionality we proposed in Stage 1. The reason for removing the search feature was that we felt that the data was too incomplete for this to be a meaningful search feature to implement at the time. The reason for removing the ability to edit recipes was because we were not able to get this piece fully working prior to our demo due to issues we ran into with our database and we would have to use a transaction which would have taken a while and we chose to use a stored procedure as one of the Stage 5 requirements already, so we just decided to leave that.

6. **Explain how you think your advanced database programs complement your application.**
   Our database programs, the trigger and stored procedure, worked hand in hand to provide the user with some personalized recommendations. Our trigger was used to update our average ratings table, which gets updated when a new review gets added to the database by a user. Then, this worked with our stored procedure, which used the average ratings of a recipe along with the information we knew about the user and the tags they commonly tried and rated highly, to recommend new recipes for the user to try that they have not yet tried. We implemented this in such a way that the most commonly reviewed recipes would be higher up on the recommendations list, while those with less reviews were considered less reliable, and were therefore lower on the list. Additionally, if the user tried the recipe and rated it, this recipe would then be removed from the recommendations list by the stored procedure since this was no longer something they had not tried before. These two database programs complemented the application because they further personalized and customized the user experience to provide them with recommendations of recipes they might like based on what they have already liked.

7. **Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future project team could use this as helpful advice if they were to start a similar project or were to maintain your project.**
   Ruchika Biswas: My main technical challenge was understanding how to use MYSQL workbench on top of how to connect GCP with the application. The yaml that we had created wasn't being read from correctly and it would not connect correctly because of that, and it was frustrating. After doing quite a bit of research, we switched our approach and simply added the requirements to the file containing the routes for the application, and ensured that these were not committed to our repository for others to be able to access our database.

Avani Puranik: Learning how to use MYSQL Workbench was a little difficult because it follows a specific format that you have to maintain while making queries that we did not know about earlier. Also, especially when creating our tables and entering data into them, we faced many problems with invisible characters and incompatibility between MySQL and our dataset, which was really frustrating to debug. One solution to this could be directly importing data to GCP and bypassing MySQL workbench altogether, since it is a little challenging to get used to. Additionally, it might be helpful to avoid using Kaggle datasets, as these are known to have compatibility issues with MySQL.

Thihan Kyaw: The main challenge I faced throughout this project was the web development experience. Many of the tutorials provided in class were given in a linux terminal but not shown with powershell. Migrating from linux to windows caused some problems that took a lot of research to resolve. The solution I found through my research was to use bootstrap, since we did not have time to implement our application using react.js.

Serena Tzeng: One of the technical issues that I personally faced was while creating the user login system. Initially, we were keeping track of users with a global variable and stored passwords directly in the database, which we realized was not secure. I did not know anything about login authentication and how to implement it, so I faced a lot of challenges with that. After reading through some documentation and tutorials, I realized I could create a very simple login system using Flask-Bcrypt and sessions. When a user creates an account and sets a password, Flask-Bcrypt helps securely hash the password to protect against potential password breaches. The hashed password is then stored in our secure database to prevent unauthorized access. Using sessions helps keep track of who (if anyone) is currently logged in and can be accessed by any route.

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

   Outside of the changes from the proposal mentioned above regarding the removal of certain functionality due to infeasibility or other issues, there were no other changes we made to our proposed core functionality from Stage 1. To reiterate, these pieces of functionality were the ability to edit a recipe and search for a recipe by specific recipe attributes, such as cuisine. We also changed course on one piece of functionality with respect to the recommendation feature, which was initially meant to be based on ingredient input from the user, but we decided it would be better to base our recommendation system off of information we stored about users in our database. We were able to implement all other deliverables from the proposal while ensuring that the technical requirements(stored procedures, triggers, advanced queries, etc.) were met as well.

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

   Other than improving the user interface, there are many things our application can improve on, such as adding more functionalities and making the code more readable, reusable, and modular. We can implement some of the additional functionalities that we mentioned in the project proposal. For example, we could add functionality where users could save recipes to "My Recipe Books" to look at later, which could include different collections of recipes (Favorites, Want to Try, Thanksgiving). We could also allow users to view a history of recipes that they have used/viewed, share recipes with friends, follow a recipe creator or collection, and generate a shopping list based on a recipe and ingredients you already have. In addition, we can add to our "I'm Feeling Lucky" recommendations feature so that you can give ingredients you already have to get a recipe you can make immediately. We can also improve our search functionality by making a combined search bar, where you can filter recipes by any combination of attributes. To make our application more efficient and improve the code quality, we can make creating/editing/deleting recipes a transaction - since it deals with multiple tables, having it be a transaction can ensure that it is atomic. Our current trigger for calculating the average ratings is not efficient because whenever a rating is added for a recipe, it recalculates the average rating by getting all of the ratings of that recipe and finding the average. Instead, we could make this more efficient by keeping track of the number of ratings per recipe and recalculating the average by only adding the new rating and dividing by the number of ratings for the recipe.

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

    Especially for the midterm and final demos of our application, the team worked very closely together on each aspect of the application, so there is quite a bit of overlap when it comes to the division of labor. With that being said, the division of labor per team member was as follows:

    <u>Ruchika Biswas</u>

    - Frontend setup for midterm demo to display user information and advanced query results using HTML
    - Backend setup for buttons to create a main landing page with routes to each of the CRUD and advanced query operations as well
    - Initial project idea creation by researching databases, coming up with core functionality, and writing up the proposal
    - Using Python to complete data cleaning and assisting with importing data into the database via MySQL Workbench

- Creation of basic CRUD queries
- Idea generation for stored procedure, as well as implementation and integration of it into the application
- Idea generation for trigger

Avani Puranik

- Frontend setup for midterm demo to display user information using HTML
- Connection from application to GCP using flask_mysqldb library in Python
- Setting up tables in MySQL Workbench and importing data into database to be updated in GCP
- Testing various indexing strategies on the advanced queries written as part of Stage 3
- Creation of basic CRUD queries
- Main implementation of stored procedure and integration into the application
- Implementation of the trigger and adding necessary tables to account for it

Serena Tzeng

- Frontend setup for midterm demo and finalization of frontend for final demo using HTML
- Creation of database schema and relations between the various tables and data selected for the database in Stage 2
- Determining core functionality and feasibility of overall project idea for proposal
- User login system to authenticate user using hashing, all user information additions, changes and deletions for final demo
- Creation of basic CRUD queries
- Backend routes for all user related functionality and display pages for recipes
- All queries related to recipe creation and deletion

Thihan Kyaw

- Frontend finalization assistance for final demo using HTML
- Project idea creation assistance by researching databases and determining core functionality
- Creation of basic CRUD queries
- Implementing and debugging issues related to the trigger
- Setup for GCP and creating the instance to be shareable with the team