411 Project Report

Team 101: nehasv2, pawudu2, alizain2, manasv2

The final iteration of our project is very similar to what we had initially planned in the proposal. We were able to fully integrate most of the functionality that we targeted:

- a button for randomly picking a recipe with long-term preferences and constraints
- a textbox to enter the ingredients available so that the website can return recipes that can be made using only those ingredients
- an option to browse recipes and add them to your 'Liked Recipes'
- a tab that allows you to see which ingredients you would need to buy in order to make the recipes in your 'Likes'

We decided against complicating the app by including preferences information and meal planning capabilities, so we did not include the following functionality:

- a survey for allergy information and long term preferences
- an option to enter the maximum calories in a recipe

To remove the preferences functionality, we decided to forgo the Preferences, Restricts and Prefers tables that were mentioned in the Database Design document. To remove the meal plan functionality, we simply did not add a feature to the WebApp that allows the user to filter recipes based on calorie level. This was replaced with the fridge because the preferences were taking away from the focus of the web application and becoming more of a health and food app which was not the intention. The intention of the fridge table was to allow the user to store their ingredients and this would help generate new recipes based on what was on the users fridge, allowing more use cases for the overall web application

PantryPal allows users to add the ingredients they currently have to their profile and allows them to interact with existing recipes. The aim was to take the burden of planning meals and finding recipes off of the users and make eating healthier more streamlined. PantryPal achieves this in three different ways:

1. It allows users to search for recipes that only include ingredients that they already own
2. It keeps track of users' Liked recipes and creates a grocery list for them that indicates which ingredients they still need (not currently in their fridge) to make all their liked recipes.
3. It generates a random recipe if a user doesn't know what they want to eat.

In the proposal, we hoped that PantryPal would make meal planning easier, less time consuming and less of a planning burden. While the current functionality of PantryPal meets these goals in the most basic aspects, it could do more. For example, we could search random recipes based on whether they are full meals, snacks or desserts. We could also rank the recipes in terms of

popularity or rating. Though these are not required to make the app useful, it would make the user experience better.

Here is a full list of functionality:

a. Get a random recipe
b. Create a grocery list
c. Add and remove ingredients from the fridge
d. Add and remove a recipe from a user's Likes
e. Click a recipe to see it's instructions
f. Create an account (create a new user)
g. Change your username
h. See reviews for a recipe
i. See the average rating for a recipe
j. Add a review to a recipe

As mentioned previously, we changed the schema of the database slightly. We removed all tables containing a Preference_Id. The source or our data remained constant. Outside of these minor changes to the database, we did not change any of the UML components from the Database Design Document. One potential schema change could have been to create a new relation for the Grocery list and insert into it after every insertion into the Likes table and delete from it after every deletion from the Likes database. Currently, we have a query to check which ingredients should be added to the grocery list. If it was its own table, we could add more advanced queries related to the Fridge and Grocery list more easily. For example, if a user had bought everything on the grocery list, we could add all of those ingredients to the fridge and clear their grocery list.

The trigger helps the application by sanitizing out data and ensuring the user input is clean and aligns with real-world constraints. The stored procedures make the application more efficient. We made the functionalities that were the most common, or most complex stored procedures so that we would only have to change the function at one place rather than everytime we use the query.

Some technical problems encountered during the project was finding functionalities that would be useful to the user and help make the web app more likely to be used. This was seen when adding triggers and transactions because many cases were already accounted for in the backend and the database did not need it in some cases which constraints covered for most of the trigger functionalities. This was solved by changing some of the original queries and adding some advanced functionalities to the wep app to allow for more complicated scenarios when accessing the database. Another challenge was handling the speed of the overall web app and the database which was solved by filling in some blank keys and making sure all tables were filled to all cases to make sure null values and sets were not being sent in. A better outlined schema would help

when designing the tables because the preferences to fridge adjustment was made later and updating the database for this addition needed some planning to be used properly so in the future if the schema was drawn a little more descriptively it would be easiest to implement and create queries for functionalities that users could find useful and benefit the application.

Another technical problem was connecting the database to the backend. The connection could sometimes be unstable, so we would get timeout errors or port closed errors. This took a while to debug but eventually we decided to just connect to the database independently every time we had to make a CRUD operation. We also had to add the IP addresses to GCP that we were working from in order to access the database, which took a while to figure out.

These are all the changes we made to the Web App from the proposal. Our proposal was slightly ambitious but covered all of the functionality that we had envisioned. So, this final iteration contains as much of that functionality as we were able to implement given the time constraints and other commitments.

In the future the application can be improved by increasing the speed of some queries by applying some indexes especially on commonly used recipes and ingredients to speed up the overall application. Some advanced features that could make the app even more useful if users could share recipes to other users and that could create a community of people's own Pantries and allow them to make it public. Creating custom recipes and adding ingredients would be useful to make the app more personalized to the user instead of just using pre defined recipes. Furthermore, we may be able to implement the initial proposed functionality: allergy information and meal planning based on calories. We could also update the app so that users can change their reviews later, or interact with all the reviews that they wrote on their settings page. We also would have benefitted from a user_id and password that was user generated rather than assigning each new user an id and expecting them to remember it. This would be a good foundation for improving the app and more features would be added to make them all flow together with the previous app. Later when the app had included these features the preferences category could be included again and this could track allergies and eliminate recipes certain users couldn't include because of these conditions making it more custom to each person and having a max calorie tracker to help personalize this app even more.

Some members managed the aspect of the database while others connected the rest of the parts to integrate with the front end. The division of labour was not completely fair, with two people putting more effort in than the other two, but everyone was given opportunities to contribute to the Web App and have their ideas heard. Queries and constraints were managed by some members and others integrated this with procedures and triggers. After the integration the front end was developed by first connecting it to the database and adding in tests to make sure all data was accessed properly and data cleaning was also done at this step as many inputs were found

NULL and corrupt data which was also found with error checking. Each member worked on their own query for the functionality of the app and tested these queries to ensure they worked with the selected tables in the database. This might have not been the best division because some interactions were more complicated than others but overall all members contributed to some part of the web application.