Project Reflection Report of the Diet Planner

Team Member:
Zhaokuan Chen zc56
Haoran Jiang hj28
Jiachen Kou jkou4
Siwei Zhang siwei4

Changes in Directions: No change was made.

Application Usefulness:
The current version of the application can carry out most of the features and accomplish the majority of the requirements that were designed earlier in the stage. Users can use the application to plan their everyday diet plan, keep tracking their daily nutrition and calorie intake. From the perspective of the weight management recorder application perspective, the usefulness of our application can be considered moderate.
Changes of Schema or Data Source:
Changed userId, recipeId, planId to auto_increment.
Added ON UPDATE CASCADE to both foreign keys in table UseFood.
Changed the attributes of User to User(userId, userName, email, age, gender, password, last_login, superuser, data_joined, first_name, is active, last_name, recommend_cal).

Changes to ER Diagram or to Table Implementation:
The current version of the tables which has attribute userId now store it as an integer where initially we use a number string(for example “00001”) to store it. However we found that it simply led to unnecessary extra work on type conversion since users can not see their user id, thus we eventually change that attribute.

Application Functionality:
The application provides CRUD functions for users to store and manage their diet recipes, weekly plan, and recipe ingredient information. The food/ingredient build function allows users to manually enter nutrition information per unit in grams; the recipes build function allows users to select ingredients from the existing foods; the week plan function lets users include multiple recipes that are personalized to their own needs. Users could also view a nutrition summary of their weekly plan, furthermore, they could manage their application account profile which contains individual personal information. Additionally, according to the gender and age provided by the user in the user profile, the application recommends personalized calorie consumption which is the average calorie consumption of other users at the same gender and same age as the current user. One of the functions that were deleted from the application was displaying detailed ingredient information in the weekly plan recipes' view.

Advanced Database Programs:
We implemented four procedures and two triggers for the database. These procedures make CRUD simpler and these triggers automated the database whenever the User table is inserted or updated.

The first procedure is CreateNewRecipe(IN userId INTEGER, IN recipeName VARCHAR(255), IN foodWeights JSON). It creates a new recipe with recipeName for the user with userId. For the newly created, the procedure assigns the food used by the recipe and the weight of each ingredient per gram.

The second procedure is CreateNewPlan(IN userId INTEGER, IN userName VARCHAR(255), IN recipeList JSON). It creates a new plan for the user with userId. According to the recipeList parameter, the procedure assigns the recipes in the recipeList to the newly created plan.

The third procedure is UpdateRecipe(IN recipeIdInput INT, IN recipeNameInput VARCHAR(255), IN foodWeightsInput JSON). It deletes the recipes that the user wants to update and creates a new recipe with a newly assigned recipe name and food weights.

The fourth procedure is UpdatePlan (IN planIdInput INTEGER, IN recipeList JSON). It deletes the plan that the user wants to update and creates a new plan with a new recipe list.

The first trigger before every insert of the table User. When a user creates a user profile. The trigger collects the gender and age entered by the user and provides recommended calorie consumption for the user by setting the attribute recommend_cal to calculated value. The recommended calorie consumption is the average calorie consumption of other users with the same age and gender as the current user.

The second trigger before every update of the table User. When a user updates a user profile. The trigger collects the gender and age entered by the user and provides recommended calorie consumption for the user by setting the attribute recommend_cal to calculated value. The recommended calorie consumption is the average calorie consumption of other users with the same age and gender as the current user.

Technical Challenges:

- Zhaokuan Chen:
    1. It should be noted that manipulating the database using raw SQL query can be difficult and time consuming for a backend built by Django. The reason is that Django supports ORM better than raw SQL query and the documentation related to raw SQL query is lacking in the Django community. Therefore, I would advise avoiding using Django as your backend because you can easily run

into errors and bugs that you can hardly find any solution online. For example, I ran into trouble when I tried to integrate the authentication functionality that Django provides with the existing User table in the database. Because the default table that the authentication module uses is hugely different from the User table in the database, I changed the User table a couple of times and ended up adding a few more attributes in the User table.

2. Django requires writing a model to represent each table in the Database and the type of each attribute in the Django model is different from the SQL attribute. Hence, after integrating the backend with the existing database, it can be difficult to manipulate the database from the backend side. Even if you want to change the database from the database's side, after changing the database from the database's end, developers need to manually adjust the model. I strongly recommend building a perfect database table with appropriate constraints which would hugely ease the workload of the backend development. For example, you can make the primary key auto_increment to save troubles, think of both ON UPDATE and ON DELETE constraints beforehand to ease the workload afterwards.
3. Additionally, calling a procedure that requires a JSON input is especially tricky. The parameter of the procedure must be a JSON in a single quote (ie. '[{"foodId": 1, "weight": 1}, {"foodId": 2, "weight": 2}]'). Because JSON in python is a list of dictionaries and the keys in the dictionaries are single quoted, the procedure will end up receiving a list of dictionaries with a single quoted key (ie. [{'foodId': 1, 'weight': 1}, {'foodId': 2, 'weight': 2}]) instead of a proper JSON file. Moreover, casting the list of dictionaries to string would make the list of dictionaries in a double quote (ie. "[{'foodId': 1, 'weight': 1}, {'foodId': 2, 'weight': 2}]") instead of a single quote. I ended up using json.dumps to change the list of dictionaries into a JSON string, passed the JSON string to the procedure, and solved the problem.

- Jiachen Kou:
    1. The biggest technical challenge is to achieve page redirection in React since the logic behind it is totally different from classic html. Instead of directly loading each url the page corresponding to when it attempts to load a page, React loads everything at once time and lets the app control page rendering. Thus in React, we need to add Router and Switch to achieve this. Besides, to make sure each

button can jump to the page we want it to load, we also need to use history.push(domain name we define on the root when we render this app). By doing so, we achieve the page loading property in React.

- Siwei Zhang:
    1. Web component modularization was one of the challenges faced in frontend development. Table display is a component that has been used on many web pages, however, data with different numbers of columns and rows were being put on the table. Thus, the compatibility and extensibility of this component have to be considered during the initial design phase, otherwise, the component will need to be re-written frequently to suit different application scenarios.
    2. APIs were provided in this project for the communication between the front end and the back end. To send information from the front end to the back end, the front end will need to pack all the user-input data into a JSON format object in order for the back end to parse. Difficulties usually occur during this stage, JSX is an implicit type conversion language that sometimes does not comply with some JSON convention which leads to the backend not successfully reading the data received.
    3. Communication between React child components and parent components was also a challenge when the nested structure grow complicated, it is especially hard to control the data flow since data in React only flow downward, in other words, child components were able to access all the data and parameters from parent components, but no direct way for a parent component to access or modify data in children component.
- Haoran Jiang:
    - 1.The biggest technical challenge is learning front-end stuff from nothing. Especially the hooks and render. Since I did not use some third-party packages, it’s pretty hard to debug and adjust the frontend page format. What’s more, while we were programming, we had to assume ourselves as users, evaluating the end-user requirements and optimizing the design accordingly. What's more, processing the data returned from the background and displaying it on the front page is also very challenging. What's more, when we add button events such as "submit", I need to figure out how the click on event works to make sure the data user wants to submit can be correctly sent to the backend.

    - 2. Delete function on WeeklyPlan was also very challenging. Since we wanted to achieve the function that users can delete the selected rows of the food. It took me a very long time to figure out how to send the selected rows to the backend with a given api. And reflect the result after deletion to the frontend page.
- ***Other Changes in the Final Build***: We followed our original proposal and kept most of the taks and requirement we had in our proposal. In the final build, we refined some functions' definition alone with their input and output. We also modified some attributes in the database table to better suit our utilization.

Future Work:
This project restricts the use of ORM in the backend. However, I believe that ORM is much more elegant and safer than raw SQL queries when dealing with simple queries. Therefore, I would retain raw SQL queries that deal with complex queries such as procedures and change raw SQL queries that only deal with simple queries back to ORM. Additionally, I am thinking of introducing advanced RESTful API into the backend, such as viewsets, routers, and generic views, to make the project close to the industrial standard. Moreover, I would like to introduce more functionalities to this application and connect the new APIs with the frontend to create a smoother interactive experience.

We also think that this application can have extra functionality like based on the recommended calories we calculated, we can make the application automatically form a recommended diet plan for the user based on the existing recipe user has recorded. Moreover, we think it is also feasible for this application to calculate an appropriate diet plan based on the fat, protein and carb that the user entered.

Division of Labor:
- Zhaokuan Chen:
    1. Implemented the backend using Django
    2. Created API to connect the backend and the front-end
    3. Optimized and Maintained the database
    4. Created the procedures and the triggers in the database

  Reflection: I completed the backend and actively communicated with my teammates who worked on the front-end to resolve any issue between the connection of the frontend and the backend.
- Haoran Jiang:
    - 1. Front-end development for the project. Designing Delete function for Food page.

    - 2. Design Plan and WeeklyPlan page, implementing Create and Delete function for these pages

  Reflection: I finished front-end pages for Plan and WeeklyPlan pages. I communicate with the backend team to update the API to make sure the frontend tests can go smoothly.

- Jiachen Kou:
  1. Implemented the user related part and navbar at frontend using React.
  2. Implement page loading functionality at frontend.
  3. Created and assist managing the Cloud database on GCP

  Reflection: I was asked to switch to my role to frontend after stage4 and contribute to part of frontend implementation. I communicated with my teammates to make sure the communication between frontend and backend is successful.
- Siwei Zhang
  1. UI and frontend architecture design for the entire project.
  2. Majority of front-end development before stage 4, including Create, Read, and Update functions of the frontend food data.
  3. Implementation of CRUD functions for frontend recipe data, and Update function for frontend plan data.

  Reflection:I designed the overall frontend structure for the team and modularized some frontend components for better program architecture. I also assisted my teammates on their frontend development tasks and communicated with the backend team for better front-back end bridging.