

# Bubble MilkTea Final Report

## 1. Briefly describe what the project accomplished.

It is an awesome software for Bubble MilkTea-holics who are concerned about gaining weight. Our Bubble MilkTea Recommendation software can store your nickname and favorite milk tea order guides, it can also recommend popular milk tea order guides according to your preferred Calorie level. In addition, the built in calorie calculator can help you figure out what tasty toppings you can add to your favorite drink while not ruining the diet.

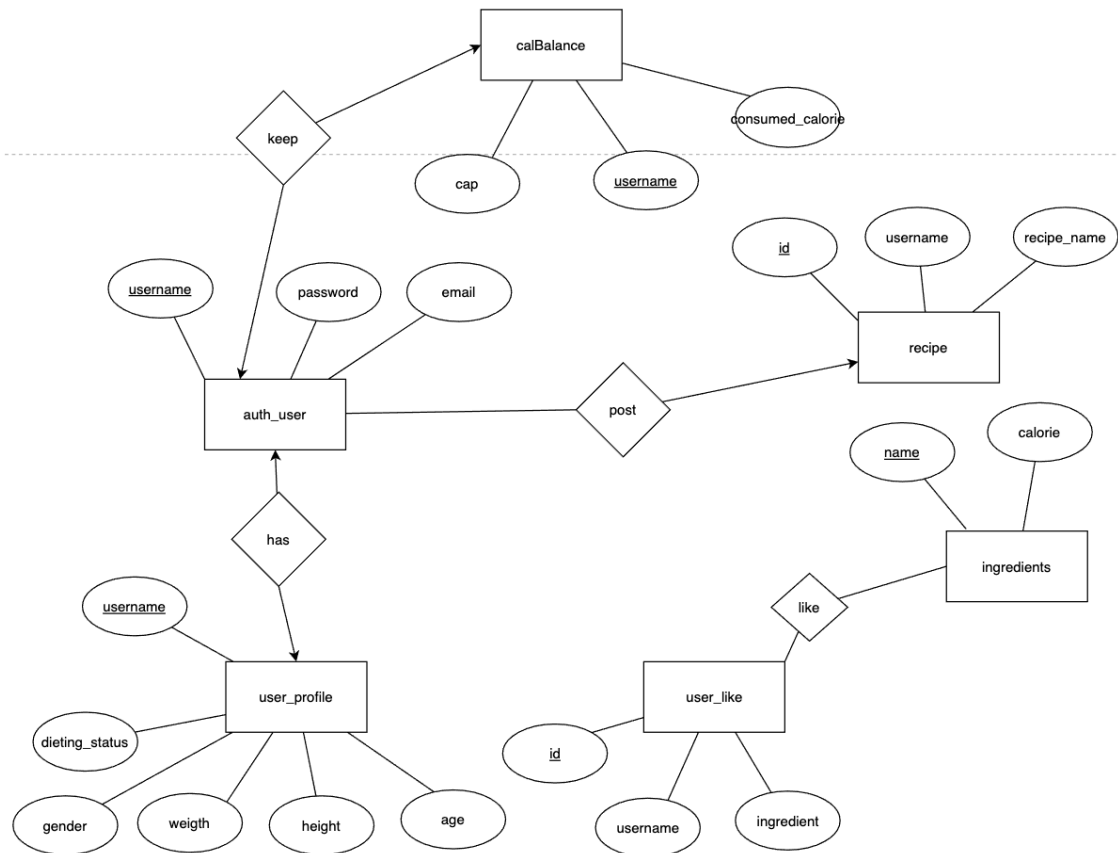
## 2. Discuss the usefulness of your project, i.e. what real problem you solved.

We made it easier for people to enjoy their favorite drink while knowing exactly how much calorie is going into their systems. This doesn't just help with people that are dieting, it can keep everyone mindful of the importance of the right amount of calorie intake. As for the people that just want to savor their drink, our app can recommend the best toppings from the numerous options.

## 3. Discuss the data in your database

All the data in our database were user generated and manually inserted into the database. Our MySQL database contains user\_auth, user\_profile, recipe, ingredient, calBalance and user\_like. The data include profile information of users(user name, password, etc), user added recipes, ingredients details, calories balance and ingredients that users like. recipe details. Our MongoDB database contains various milk tea recipes which contain the name of each ingredient and its quantity.

## 4. Include your ER Diagram and Schema



```

create table user_profile
(
  username varchar(100) primary key ,
  age int(100),
  gender  varchar(20),
  height  decimal,
  weight  decimal,
  dieting_status varchar(100)
);

```

```

create table ingredient
(
  name varchar(100) primary key,
  calorie integer
);

```

```

create table auth_user
(

```

```
username varchar(100) primary key ,  
email varchar(100),  
password int(100)  
);
```

```
create table user_like  
(  
Id int (100) primary key,  
username varchar(100) ,  
ingredient varchar(100)  
);
```

```
create table recipe  
(  
Id int(100) primary key,  
username varchar(100) ,  
recipe_name varchar(100)  
);
```

```
create table calBalance  
(  
username varchar(100) ,  
cap int(100),  
consumed_calorie int(100)  
);
```

5. Briefly discuss from where you collected data and how you did it (if crawling is automated, explain how and what tools were used)

We sent out questionnaires to our friends on social media apps to get user profile information and ingredients that users like. The data gathering process was mostly volunteers filling up google forms. We then analyze the data and insert it into the database using Python script. For the ingredient table, we manually collected calorie information from [www.nutritionix.com](http://www.nutritionix.com), a website allowing users to query nutrition facts of given food.

6. discuss how you used a NoSQL database in your project

We used NoSQL database to store recipes both created by users and recipes created by us. Recipes have no correlation with each other therefore it is more efficient to store them in a non-relational database.

7. discuss your design decisions related to storing your app data in relational vs. non-relational databases.

We store recipes in non-relational databases. Because each recipe could have different numbers of ingredients along with their quantities. We are not sure how many columns we are going to have so we can't use relational database. While NoSQL databases have dynamic schemas for unstructured data. So we decide to use MongoDB to store our recipes.

8. Clearly list the functionality of your application (feature specs)

- Each user having a profile that stores their biometrics related to dieting
- Tracker for controlling calorie intake
- Calorie calculator for a drink
- User created recipe / lookup
- Recipe lookup on things other than drink (chicken, banana, you name it)

9. Explain one basic function

One of the basic functions in our app that also had uppermost importance was the calorie calculator. The calorie calculator first gets a list of ingredients from the backend and presents it in front of the user. The user then has the choice of adding them to their favorite drink. A circular progress bar on top shows exactly how much calorie the drink will be containing and how much calorie can the user "spend" for the drink.

10. Show the actual SQL and NoSQL code snippet

```
@api_view(['POST'])
@permission_classes((permissions.AllowAny,))
def get_user_profile_by_name(request):
    username = request.data['username']

    with connection.cursor() as cursor:
        cursor.execute("select * from user_profile where username = %s", [username])
        row = cursor.fetchone()
        columns = [col[0] for col in cursor.description]
        if row:
            return Response([
                dict(zip(columns, row))
            ])
        else:
            return Response({"error": "user id not found"})
```

Figure: SQL code snippet

```

37 @api_view(['POST'])
38 @permission_classes((permissions.AllowAny,))
39 def get_ingredients_by_name(request):
40     name = request.data['name']
41     ret = Recipe.objects.filter(name=name)
42     data = []
43     for r in ret:
44         data.append(r.ingredients)
45     return Response(data)
46

```

Figure: NoSQL code snippet

11. List and briefly explain the dataflow, i.e. the steps that occur between a user entering the data on the screen and the output that occurs (you can insert a set of screenshots)

Dataflow 1: Edit user profile

The screenshot shows a web form for editing a user profile. It has a light blue header and a white body. The form contains the following elements:

- Age:** A text input field with the placeholder text "Enter age".
- Gender:** Two radio buttons labeled "Male" and "Female".
- Height:** A text input field containing the value "160".
- Weight:** A text input field containing the value "60".
- Dieting Status:** A text input field containing the value "YES".
- Submit:** A blue button with the text "Submit".

user enter certain informations in the UI

the frontend handles input information and stores as states

the frontend sends http request to the backend URL

the backend handles the update

	username	gender	age	weight	height	dieting_stat...	
▶	andrew	Male	25	90	176	YES	
	anthony	Male	37	75	168	YES	
	ashley	Female	20	69	182	NO	
	barbara	Female	23	190	190	NO	
	betty	Female	38	110	160	NO	
	charles	Male	26	54	171	NO	
	christopher	Male	45	77	178	NO	
	daniel	Male	47	77	177	NO	
	david	Male	24	60	178	NO	

the mysql database updates

the frontend asks for the data and set states according to responses

the backend handles the request and sends data back

the frontend renders the state

age: 15

gender: male

height: 160

weight: 60 lbs

dieting status: YES

the user gets information

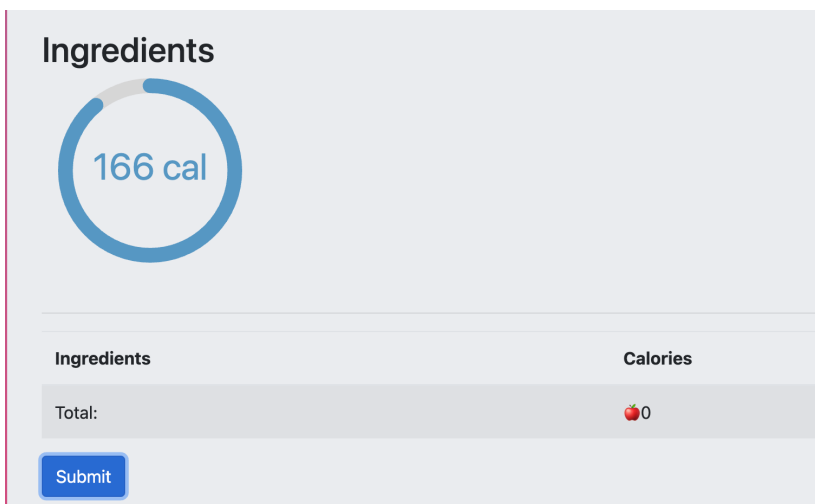
Dataflow 2: Calorie calculator

Ingredients	Calories		
thai tea	5		
tea	25		
taro ball	270		
syrup	265		
sweet red bean	170		
sugar	387		
pudding	130		
oreo	160		

user click on ingredients











the frontend adds the selected ingredients into a list and calculate the sum of the calories



the user clicks on submit and the frontend will send http request to the URL with the data of consumed calorie, and then the circle graph will reflect the change

username	consumed_calorie	cap	
Ada	1100	1500	
Alice	1800	1500	
andrew	5	1500	
anthony	400	1500	
ashley	400	1500	
barbara	300	1500	
betty	50	1500	
Bob	400	1500	
carol	30	1500	
charles	40	1500	
christop...	200	1500	
daniel	100	1500	
david	50	1500	
donald	200	1500	
donna	20	1500	
dorothy	500	1500	
duo	1000	1500	

Dataflow 3: recommendation

bubble	362		
brown sugar	363		
battered pearl	360		
almond milk	23		

You will get recommended recipes lower than entered calories

Set

Ingredients

the user enters expected calorie level

the frontend send the http request with the expected calorie to the backend, and the backend will do a dfs search and return recipes ( combination of ingredients which has a sum of calorie lower than expected one )

You will get recommended recipes lower than entered calories

Set

Ingredients

pudding

oreo

pudding

sweet red bean

honey



12. Explain your advanced function 1 (AF1) and why it's considered as advanced.

In advanced function 1, our group applied the stored procedure. In fact, stored procedure is a set of SQL statements and can be called by the function in Django. The main idea of stored procedure relies on two aspects, and we achieved it in the statement of 'if... else...', because we want to show an alert to our users, so if the additional calories of the ingredient the users choose are more than the available calorie based on our cap (1500 Calories), we will return an alert message in our demo ('Stopping drinking more bubble milk tea !'). Else, we will update the table called 'Cal Balance', and add the additional calories to the existing or consumed calories. In the recommendation part, we use all the ingredients from the ingredient table and DFS algorithm to find all the possible ingredient combinations. The sum energy of items in each combination equals the target value within a certain range that system sets. If the user chooses the favorite mode, then we will use information from the user\_like table and each of the recommendations will contain the ingredient they like.

Although our applications are similar to some apps' functions, our application is creative enough. Milk tea is popular with people nowadays, in our campus there are about 6 milk tea shops. So when users log in to our website, with this function it not only helps to record people's favorite ingredients, but also creates an alert for people who like milk tea but also what to keep a good diet status when enjoying drinking the milk tea.

13. Describe one technical challenge that the team encountered.

There was a technical challenge when we wanted to embed the mysql database and mongoDB database together in our django project. We first seek for a solution called django database router, but we found it complicated. This solution would require another file for the router and almost 100 lines of codes to realize the multi database router. And the router requires models but we didn't use models and used raw mysql tables and database connection cursors instead. Using database router would force us to rewrite a lot of codes. Therefore we seek for an alternate. We finally found out that database connection cursor had a very simple approach to achieve multi database implementation by using database connections instead. It was impressive to search deep through the Internet to find a perfect solution.

ps: I am happy to share the solution in case anyone's curious

<https://code.djangoproject.com/ticket/12941>

14. State if everything went according to the initial development plan and proposed specifications, if not - why?!

Since we planned out each step and divided work with details at the start. We didn't run into too many obstacles. Whenever something gets slightly too problematic for a single person to deal with, we gather the entire group to come up with a solution.

15. Describe the final division of labor and how did you manage team work.

The way we thought was best for creating a good project was have one person supervise, overlook progress and assign tasks. And three others working on frontend, backend and database. We divided tasks based on our strengths to maximize efficiency and ensure quality. However, whenever we need to brainstorm for ideas, it's always a team effort. The report was a group effort, each group member writes about their corresponding contribution.

Hanyue was our team leader and fit for the supervisor role. She kept track of deadlines and objectives and assigned everyone tasks respectively. She then puts everything together and discusses the details with the rest of the team.

Johnny (Ruining) was the front end person, he made front end components based on the needs of the project. He designed key components for the front end such as the recipe adder, the recipe search book, login / sign up page, user profile page and etc.

Yingyue took care of the stored procedure in the advanced function. The stored procedures is a set of SQL statements that can be called by the function in Django, which is our backend. The stored procedures calculate calories and update the calorie balance table, which is the core of our project.

Yixuan was in charge of the backend with django and wrote the backend code of advanced function 2. She also helped link up the communications between the database and the backend.