

SugarTracker

Less Sugar Does More for Your Health

Arth Bhattarai 301268283

Alice Zhang 301221849

IAT352 D101

Professor: Marek Hatala

Teaching Assistant: Fatemeh Salehian Kia

Executive summary	2
Project proposal	2
Data requirements analysis	3
ER diagram	3
Implementation decision for DB tables	3
Data description and process	3
Snapshot of database structure from phpMyAdmin	4
Implementation details	4
Snapshot(s) of the front end demonstrating the functionality	4
Description of the backend support	4
Description of the front end dynamic support/ajax	4

Executive summary

Sweet drinks are cheap and easily available, but can also be agents of obesity, diabetes, and other related disease. They have become more accessible than water itself in some areas. The problem we aim to solve is to help the user gain insight on how much additive sugar they consume in a week.

SugarTracker is an application that lets users see how many grams of sugar are in conventional sugary drinks and how it accumulates in their diet. The purpose is to implement a system to track a user's sweet beverage consumption, while providing a quantifiable view of their additive sugar consumption through sweet beverages. SugarTracker is intended for adults and young adults without a medical background who want to increase awareness on their sugar consumption, learn how much sugar is in a commonly-consumed sweet drink, and reduce their sugar intake from beverages.

Users can view the details on one drink, its description, and the grams of sugar per 100mL. Users can register to become members. Members are able to add drinks to their own list of consumed drinks, which will append to their amount of sugar consumed per week. Members can track their current consumed sugar on a progress bar, for which they can set the maximum amount. They can also see the current consumed sugar in terms of easily-visualized sweet snacks.

This report will analyze and discuss the design, data requirements, and implementation decisions of SugarTracker and explain the front-end and backend functionality.

Project proposal

SugarTracker helps a user keep track of the sugar they consumed in the timespan of a week and allows them to visualize the amount of sugar in terms of units of different sweet snacks such as Hershey's snack-size milk chocolate bars¹, gummy bears², Oreo cookies³, tic tacs⁴, and sugar cubes. Visitors can view different sugary drinks

¹ HersheyBar Image,
https://vignette.wikia.nocookie.net/logopedia/images/d/d6/Hershey-bars-milk-chocolate_lg.png/revisi/n/latest?cb=20170204211133,

Hershey Nutrition Info: Label.

² <http://pluspng.com/img-png/png-gummy-bear-pink-gummybear-png-300.png> - gummybear image.
Nutritional Info: livesitrong.com

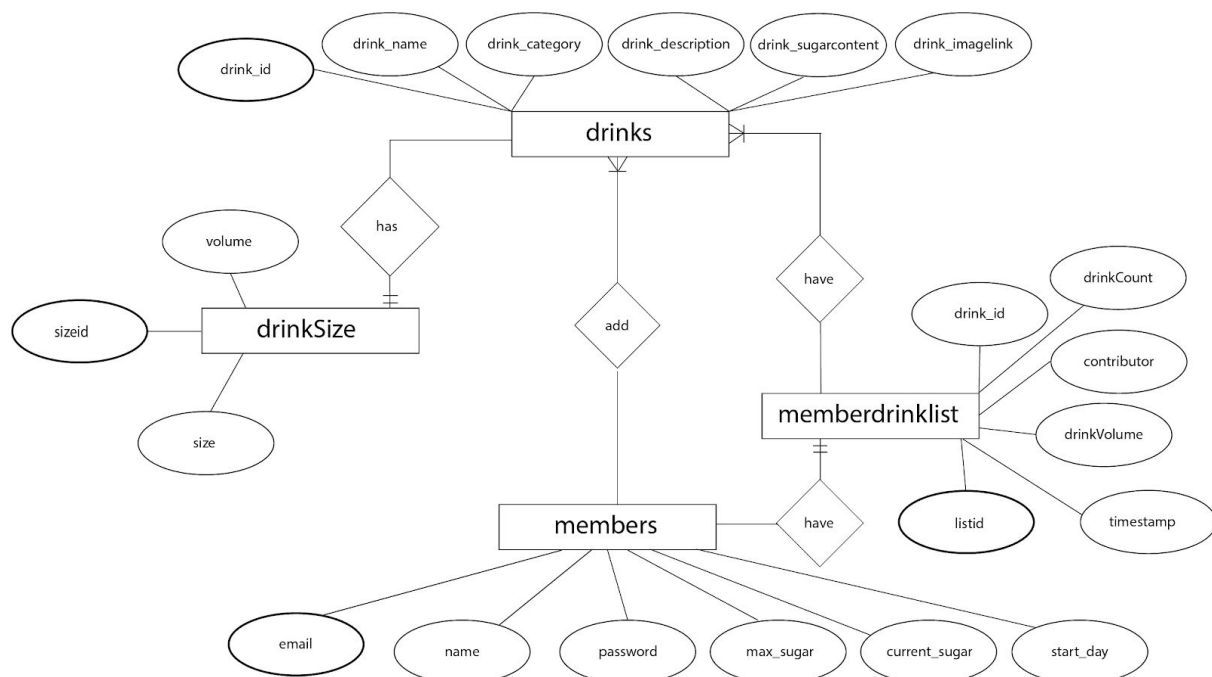
³ https://png.pngtree.com/element_pic/16/11/25/f4d26ffd5e2e1734ad282a9afb531d7e.jpg - oreo png,
Nutrition Info:<http://www.sugarstacks.com/cookies.htm>

and their respective detail pages, which contain information about the grams of sugar that are in 100mL of the drink, and how that amount reflects toward an equivalent measure in terms of a sweet snack.

Visitors can register with their email and name to become a member. In addition, members can record the sugary drinks they drink in one week, set their personal maximum goal of sugar intake from sweet beverages, receive warnings for reaching and exceeding the goal, and measure their consumed sugar amount in terms of the aforementioned sweet snacks.

To increase customization, SugarTracker allows members to add their own drinks to the database. Members can also make their own custom drinks to add to the database. The only prerequisite information for adding a drink is the drink's volume and sugar content in grams. To assist members who do not use grams in measurement, a helpful conversion table is placed beside the form for adding drinks.

ER diagram



Our ER diagram is made with 4 tables: members, memberdrinklist, drinkSize, and drinks. The main table is drinks, which we already populated with 20 entries. For each drink entry added by the members, there can be one drink size, which is its own table containing names and volumes of different sizes. Entries to the members table are added as users sign up to be members. The memberdrinklist stores all the

⁴ <http://www.afdr.org/showfile.ashx?id=c5f417cc-2c2a-4536-8b3b-9c222eef76f7> - Tictac Image, Nutrition Info: Label.

drink entries made by all members, associating entries to a certain member using the contributor foreign key.

Data requirements analysis

SugarTracker uses data on sweet drinks and their properties and sizes and the corresponding amounts in millilitres (mL). We made our own data to store member credentials and a list of drinks belonging to all members, which is connected to the members data by a “contributor” attribute. This section explains how we acquired the data and planned on its structure in relation to the ER diagram.

Firstly, we needed data on the many commercially-available sweet drinks. We were able to find this data in different health websites⁵, but not many were in JSON format. To allow the room for members creating custom drinks and to moderate our workloads, we found 20 readily-available sweet drinks to populate our database with. Along with drink id, name, description, category, and image link, we have an attribute of grams of sugar per 100mL.

We also needed information on drink sizing that would be flexible across different types of drinks, as most commercial standard sizes have volume differences based on company⁶ (eg. Blenz vs. Starbucks) and based on country (eg. American sizes are larger than the Canadian sizes for most brands). Thus, we would need a table with an assortment of sizes we can choose from. We went on websites of different providers of sweet drinks to find the sizing information, converting from imperial to metric versions of measurements wherever necessary. We used Google’s conversion tool to obtain the converted values.

For members, we needed a unique email that they register under, and a name that will be used in their profile page for a more personable greeting once they arrive at the profile page. We needed to store their current sugar intake and their personally-set maximum sugar intake, which would be a default value calculated as 7 times the World Health Organization daily recommended amount of additive sugar

⁵www.health.act.gov.au/sites/default/files/Fact%20sheets/Sugar%20Content%20of%20Popular%20Drinks%20-%20Considerations%20for%20Children%20in%20Sport.pdf

www.ethicalconsumer.org/ethicalreports/softdrinkssectorreport/sugarcontent.aspx

www.sugarydrinkfacts.org/resources/nutrition/Nutritional_Content.pdf

⁶ <https://www.starbucks.ca/menu/nutrition-info>
<http://www.timhortons.com/ca/en/menu/nutrition-and-wellness.php>

consumption⁷. The member may change that value at any time. The date to start counting the week is used to calculate when their own drink tracking list should reset.

Implementation decision for the DB tables

We sorted our information based on drinks, sizes of drinks, members, and members list. The drink's data would contain the description, an image link that needs to end in .png, the sugar content per 100mL of the drink, and the category it belongs to. We made 5 possible categories for drinks which are alcoholic, caffeinated, dairy-based, energy/sports, fruit-based, soda, and other. These attributes are not part of another table in the database, but are options that are consistent throughout all our web pages. The drink_id and user's primary key, the email, connect the drink to the memberdrinklist table after the member has added it. The query of which is

```
"SELECT * FROM drinks JOIN memberdrinklist ON drink_id = memberdrinklist.drinkID WHERE contributor='$thisemail'"
```

with \$thisemail storing the value of the \$_SESSION variable holding the signed-in member's email.

Because more than one drink can be added to the memberdrinklist, the memberdrinklist table has a one to many "has" relationship with the drinks table. Members have a relationship with memberdrinklist that is one-to-one, since one member has one drinks list. One member can add many drinks, whether they are custom drinks or drinks from our base database, to their list. Therefore, members have a one-to-many relationship with drinks. Each drink has one drinksize, so there is a one-to-one relationship between those tables.

The members table contain attributes about members, which they input as they register. We initially had the member's start_day (the date to start counting the 1-week interval) as a String, which would equal the DATENAME(weekday, GETDATE()) function that is called in SQL. But upon realizing that just checking if the day of the week of the start date equals the current date's day of the week (eg. 'Tuesday == Tuesday') is not enough, as it would need to consider cases where the user logs in after 7 days, we decided to make it a DATE type variable in the format yyyy-mm-dd. This allows a second check to be performed on the number of days elapsed between the current day and the start day. As for the max_sugar attribute, we originally made it equal a fixed value upon registration. However, we considered the daily recommended amount of sugar that the World Health Organization recommended⁸ and multiplied that value by 7 to include a more realistic maximum for the user. Max_sugar is an integer so that when we calculate the condition for

⁷ <https://www.cbsnews.com/news/world-health-organization-lowers-sugar-intake-recommendations/>

⁸ <https://www.cbsnews.com/news/world-health-organization-lowers-sugar-intake-recommendations/>

exceeding the weekly amount, we avoid any floating point position errors, as `current_sugar` is a float variable for the sake of precision on describing amounts of sugar consumed.

`Memberdrinkslist` table is connected to `membertable` by the foreign key contributor, which is equal to the member's id (email) when associating a drink to the member that added it. When the member is on the profile page, the sql query to display the drinks is

```
"SELECT * FROM drinks JOIN memberdrinklist ON drink_id =  
memberdrinklist.drinkID WHERE contributor='$thisemail'"
```

So they can see all the drink they added, and nobody else's.

Data description and process

We manually populated the drinks table with 20 entries initially. The information we found were from PDF files containing information about sugar content in different drinks, as well as fitness or personal health websites with similar information. Sometimes, we would convert the volumes from fluid ounces to millilitres, or convert a different ratio of sugar to volume to the one we needed (per 100mL).

We originally found it challenging to store the sizing information for the different drinks without having redundant data. We originally thought that each drink could have a size, but due to the multiple possible sizes and the difference in sizes from company to company, we realized this was going to end up in redundant data. To remedy this, we made a `drinksize` table that stores the primary key `sizeid`, then two more attributes, `size` and `size volume`. When a user adds a drink to the database, they can add in terms of a certain volume, or write their own volume in an input field. Similarly, when a member adds their custom drink, the user can choose from predefined volumes as well as write their own. Possible volumes are displayed in a drop-down list using the following code in `showdrinks.php`:

```
$querySize = "SELECT * FROM drinkSize";  
$resultSize = mysqli_query($db,$querySize);  
if($resultSize){  
    while($row = mysqli_fetch_array($resultSize))  
    {  
        echo"<option  
value=\"\".$row['volume'].\">\".$row['size'].\"</option>\";  
    }  
}
```

For most of our data, including sizing, the tables were populated using mySQL insert statements or by means of the myphpadmin user interface. The data for the sizes were found from Google searches on the different branding sizes and standard sizing for commercial packaging. Finally, we made two test accounts for testing members-only functions, making more as necessary to test the functionality of register.php and changing the user's settings in profile.

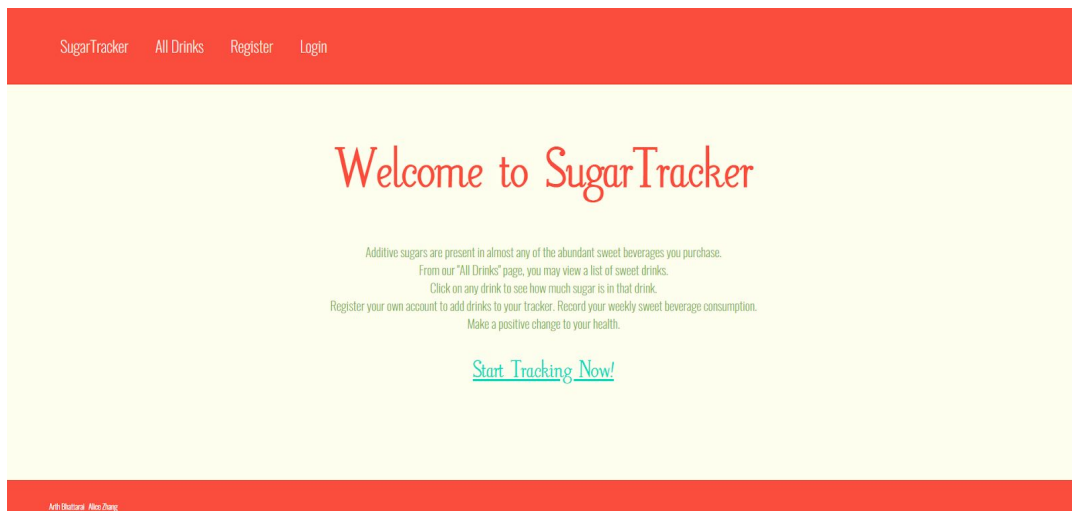
Snapshot of the database structure from phpMyAdmin



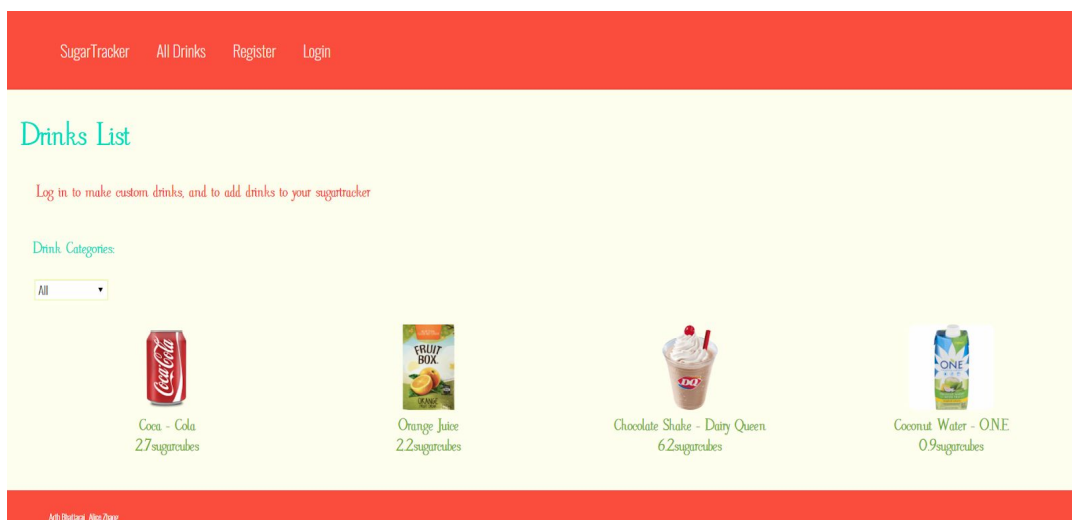
Implementation details

Snapshot(s) of the front end

This part indicates visitor/generic user functionality:



Landing page



Landing page

This part indicates member functionality:



Visitor can register

SugarTracker
All Drinks
Profile
Logout

Welcome, Alice. View your weekly sugar intake below.

Your weekly sugar tracker will reset in 7 days.

0.00 grams
210 grams

The WHO guidelines recommend consuming a maximum of 30g of additive sugar per day.

Edit my details
Reset Tracker

Drinks

Arth Bhattacharjee Alice Zhang

Login or register will redirect to profile page upon successful submission.

Drinks

Time Consumed Drink Name Drink Volume Total Sugar (grams) Comparison

Add a Drink
Create a Custom Drink

Sugar Comparison Chart

Type


Sugar Content

4 grams of sugar.


0.45 grams of sugar.

0.81 grams of sugar.


Login or register will redirect to profile page upon successful submission.

Add to Tracker

Caramel Macchiato - Starbucks
17sugarcubes


Choose Volume:
ChooseOne
or Write The Volume in mL:

Add to Tracker

Mountain Dew
3sugarcubes


Choose Volume:
ChooseOne
or Write The Volume in mL:


Add to Tracker

San Pelligrino blood orange soda
2.5sugarcubes


Choose Volume:
ChooseOne
or Write The Volume in mL:


Add to Tracker

Ginger Ale - Canada Dry
2.3sugarcubes

Choose Volume:
ChooseOne
or Write The Volume in mL:

Add to Tracker


Add to Tracker


Add to Tracker


Add to Tracker


Arth Bhattacharjee Alice Zhang

The drinks.php page will display the whole database of drinks, which the member chooses to add.

Adding a New Drink

The WHO guidelines recommend consuming a maximum of 30g of additive sugar per day.

Drink Name:

Drink Description:

Drink Image Link:

Enter total sugar amount: grams

Enter Volume of Drink: milliliter

Drink Category:

3 teaspoons	1 tablespoon	1/2 ounce	14.3 grams
2 tablespoons	1/8 cup	1 ounce	28.3 grams
4 tablespoons	1/4 cup	2 ounces	56.7 grams
5 1/3 tablespoons	1/3 cup	2.6 ounces	75.6 grams
8 tablespoons	1/2 cup	4 ounces	113.4 grams
12 tablespoons	3/4 cup	6 ounces	170.1 grams
32 tablespoons	2 cups	16 ounces	453.6 grams

Arif Bhutani Alex Zhang


Members can alternatively make their own custom drink.

42.60 grams  210 grams

The WHO guidelines recommend consuming a maximum of 30g of additive sugar per day.

[Edit my details](#) [Reset Tracker](#)

Drinks

Time Consumed	Drink Name	Drink Volume	Total Sugar (grams)	Comparison
2018-04-14	Mountain Dew	355ml	42.6	 Number of Sugar Cubes: 10.7 (42.6 grams)

[Add a Drink](#) [Create a Custom Drink](#)

Arif Bhutani Alex Zhang

After adding a drink, the member can see how it affects their sugar consumption progress bar.

Your weekly sugar tracker will reset in 7 days.

42.60 grams 210 grams

The WHO guidelines recommend consuming a maximum of 30g of additive sugar per day.

[Edit my details](#) [Reset Tracker](#)

Change your name:

Change your personal maximum sugar intake:

Comparison Unit:

Change your password:

Confirm new password:

[Confirm Changes](#)

Arth Bhattacharjee Alex Zhang

The member can change their personal stats by toggling this form in their profile page.

Description of the backend support

The backend support is done with PHP, with database CRUD operations done with MySQL. The landing page uses jQuery for the fade-in animation of the text. The jQuery code for the functionality is in myJSFile.js, in the document.ready() block. On the alldrinks.php page, an ajax function supports the filtering of drinks by their category. When hovering over a drink, css functionality makes a box shadow appear over the image icon of the drink. The ability to add the drinks, if the user has logged in, is processed by php sending a query to MySQL. If that query is successful, it will append a drink to membersdrinklist under that member's email as the \$_SESSION['valid-user'] value. The drink description page has a query to MySQL where the drink id is returned, then appended to the URL. When the member logs in, there will be a query to MySQL to check if the email matches an already existing entry to the email attribute (primary key) in the members table. If so, the email and the current_sugar of the user are both stored as a \$_SESSION variable. Upon registration, a query is run to see if there are already-matching credentials, and if not, then registration is successful and they are redirected to their profile page with the \$_SESSION variable as their email and the current sugar as 0.00 grams. Editing the user settings will send a query to change the respective values in the members table, and the password will only change if both the password and confirm password fields are filled, and with the same password. The reset button allows the member to clear their drink list and make the tracker count the current day as the start_day, or start of the week that it will track. Adding a custom drink will insert a value with the respective attributes to the drinks page, which can then be seen in drinks.php. Any member can then add that drink to their tracker.

Description of the front end dynamic support/ajax

In drinks.php (All Drinks in the nav bar), the drink list is dynamically populated as per the category chosen by the user in the dropdown menu. The dynamic change of content is supported through asynchronous javascript, which fills in the div where id=txtHint with content from showDrinks.php. The script loads the page, asynchronously or in a separate thread thus, not having to refresh page every time the content changes. The dropdown on being changed activates the script, the data from the dropdown is passed along in the URL of showDrinks.php as "showDrinks.php?category="//datahere". The data is obtained using \$_GET in showDrinks which, then uses the appropriate query to populate the list of drinks. Likewise, the same technique is used in drinkdetails.php to dynamically change the type of comparison (sugarcube, tic tac, hershey, etc) of sugar content as per the users choice from the drop down list. In this case, the data obtained from the drop down is the type, and the data passed in the URL are the amount of sugar in the drink, and the the type chosen from the drop down. The script fills in the div with content from sugarcubes.php in this example of asynchronous loading of php content. As per the type chosen, the data sent is changed, and the image produced is changed.

Likewise, the 'Reset Tracker' button on profile.php page, loads the content from page changeuserdeets.php asynchronously below it when pressed. The function used to script this interaction is from the jquery library. Although no data was sent during this interaction using the jquery method makes it easier to send multiple packets of data through one ajax call, instead of having to code them separately. The button dynamically creates a dialogue box, which asks if the user wants to continue wiping their tracker data, if the member presses yes, the data is wipe asynchronously and then the page is reloaded.