Food Waste Tracker

Michael Liu, James Wong, Divya Sengar, Alan Kai, Andrew Chuba
CS4624: Multimedia, Hypertext, and Information Access
Virginia Tech, Blacksburg, VA 24061
Instructor: Dr. Edward Fox
Client: Susan Chen
4/28/2017

Table of Contents

Table of Figures

Table of Tables

I. Executive Summary

Approximately 40% of all edible food is wasted each year, costing family approximately $1,500 a year. Consequently, we undertook a task for our client, Susan Chen, in an effort to combat this issue. Our client, currently a first year, graduate student at Virginia Tech pursuing a Master's degree in Human Nutrition, Food, and Exercise, requested that we create an online-based, public service announcement tool to raise awareness.

After several rounds of concept and design refinement, the solution was realized in the form of a website. The purpose of this website is to allow visitors to visualize the current and long term, extrapolated impacts to them and society from food wasted in just a single meal. One video was created for the website to educate users on statistics related to wasted food. Another was created to urge users to not waste food at home.

The front end, i.e., webpage experienced by visiting users is ultimately an HTML document. It is also powered by JQuery to add a number of useful functionalities. One such function is an auto-complete feature so that users can dynamically see available options as they search for food types. On submitting their inputs, visitors will be shown a statistics page powered by D3.js, a JavaScript library for data-driven documents.

Node.js is used on the server-side to provide the user input and statistics webpages. When a visitor submits the food types and amounts wasted to the server, the server queries the MySQL database for the appropriate data. The MySQL database is built on top of two datasets, one from a study by the USDA Agricultural Research Service and another from a separate study by the USDA Economic Research Service. This provides 7-year spanning, nationwide unit price averages for numerous food groups to calculate the desired statistics.

Certainly, there were a few challenges that appeared during the project's development. One was attempting to fuse two independently gathered datasets. Another was dealing with improper user inputs. However, as the issues were debated, they were eventually solved one by one.

Ultimately, the current product fulfils the required task and goal. Users can calculate their wasted food and from a single source, see the result of that and the impact on them and those around them. Nevertheless, the current richness of the data in the database and modernity of the webpage design means that there is still untapped potential for improvement for this product.

II. Introduction

A. Goals and Objectives

Our objective is to create a visually appealing webpage that engages users to enter food waste information and examine a visual representation of the cost of incurring the waste. To achieve this objective an interactive website will be built with facts, figures, and videos as a public service announcement to users. Users will be able to enter a description of their last meal, the amount wasted by estimated weight, and then see a visual extrapolation of that information. The information will present the user with how much money, in dollars, can be saved over a period of time. We hope that users will be more self-aware of food waste when presented with how food waste can financially affect them and how much potentially could be saved. The website is to serve as an educational tool to the public. The website will be mobile compatible.

B. Client Background

Susan Chen is a first-year graduate student at Virginia Tech pursuing a Master's degree in Human Nutrition, Food, and Exercise (HNFE). Susan's research focus is "Food waste among youth and adolescent". Susan hopes that a website that can visually show the consequences of food waste will engage users and combat against wasting food.

C. Statement of Scope

As a service meant to depict the growing amount of food waste, it's important to realize who the users will be. We will specifically target those who tend to be less aware of their food waste. While this could be seen as being very vague, having as large as possible base of users would be the best in this situation.

D. Major Constraints

In order to serve as many people as possible, it's important to use technologies that everyone can access. Thus, we have decided to limit ourselves to using a website, since it is expected that the majority of the target audience will have access to the internet.

We also realize the importance of having a seamless experience that keeps the user engaged in the entire process. We want to ensure that no one loses interest midway through the process, so our design will be based around simplicity and ease of use.

In the vein of keeping the user engaged, we have decided to design a website with as few pain points as possible in its interaction. Thus, we have decided to use pictures to display potential food groups that the user can easily recognize and choose from.

E. Organization of the Report

The Users' Manual provides a walkthrough to the web site created in this project. It details the expected interaction sequence by the average user. The Developers' Manual provides instructions on how to install the project onto a Linux host machine. Following that are Acknowledgements to those who have given their help with completing the project and References to works used to support the creation of the project. The Appendix section contains developmental reports created over the course of creating the project. Because both the body of this report and the developmental reports reference from the same pool of figures and tables, Figures and Tables both come after the Appendix.

III. Users' Manual

This website can be used to show food waste over time based on the food a user has wasted in a session. It works as follows:

The front page includes 2 fields, one for "Food Group" and one for "Mass in grams" and two buttons "Add food" and "Calculate Results". The user can use these to enter in information about the user's food waste.

The user should begin by typing into "Food Group." There are fields that are going to be suggested and images that pop up in the grid below. The user needs to select either an autocomplete entry or an image to select the food group. Once the user has done that he/she can enter in the mass in grams into "Mass in Grams". Then the user can select "Add food" and it will appear to the right of those fields in a list. The user can keep adding in information about their food waste in such a way until they are satisfied. When they are ready to see how much food has been wasted they can select "Calculate Results".

On the next page the user will find the "Food Waste Over Time" graph that shows, based on the amount of food the user wasted recently, how It can numerically add up over time. Also featured are other relative statistics based on that same waste applied yearly such as the dollar value of what the user would have wasted, how long the user could have fed someone else with that presumptive number, and even the food's waste converted into hamburgers, bananas, and chickens. At the end of the page the user will find a link to our videos which educates the user on food waste. The user can watch these two videos and become more educated on food waste to hopefully waste less food in the future.

IV. Developers' Manual

A. Installation

The following instructions will help install the Web server and database on baremetal to have the Food Waste Tracker running as a standalone website.

1. Installing MySQL

```
sudo apt-get update
sudo apt-get install mysql-server
```

Configure the MySQL server during the installation.

2. Importing the Database

Food Waste Tracker Database README

Typical usage:

1. Copy this "FoodWasteTracker_MySQL.tar.gz" archive to the machine hosting and running the MySQL server.
2. Extract the installation files:

```
tar -xvzf FoodWasteTracker_MySQL.tar.gz
```

3. Import and populate the database using one of the following commands:

○ Through a MySQL user requiring a password:

```
./db_import.sh -u YOUR_USERNAME -p YOUR_PASSWORD
```

○ Through a MySQL user without a password:

```
./db_import.sh -u YOUR_USERNAME
```

Verbose mode may also be activated with the -v flag

```
./db_import.sh -u YOUR_USERNAME -v
```

A detailed view of the import script's usage may be displayed with the -h flag

```
./db_import.sh -h
```

3. Installing the Web Server

1. Ensure that both Node.js and npm is installed on the machine.

```
sudo apt-get update
sudo apt-get install npm
```

2. Find the MySQL server's listening port from the config file at [mysqld].

```
/etc/mysql/my.cnf
```

3. Change the fields user, root, port of the MySQL configuration in the FoodWasteTracker's index.js:

```
var connection = mysql.createConnection({
host     : 'localhost',
user     : 'YOUR_USER_NAME',
password : 'YOUR_PASSWORD',
database : 'food_waste_tracker',
port : 'PORT_NUMBER_MYSQL_IS_LISTENING_ON'
});
```

4. Now install all the dependencies

```
npm install
```

5. Then run the web server in the background

```
sudo npm start&
```

6. If the program takes control of the shell, CTRL-C in order to stop it.
7. Now jobs should show the web server running in the background.
8. It should now be running on the machine at port 80.

4. In Case of Failure

1. Ensure that MySQL is running

```
sudo service mysql status
sudo service mysql stop
sudo service mysql start
```

2. Ensure that the web server's port isn't already in use

B. Inventory of Files

A repository for all of the source files may be found at: https://github.com/divyasen/FoodWasteTracker.

1. Database

Refer to figure 9 for the directory tree.

- db_fwt.sql
  - This file contains the SQL queries to build and load the schema/database/tables.
- db_import.sh
  - This script executes the SQL queries with the MySQL command line tool, and checks that the proper number of entries for each table were successfully imported.
- README
  - This file contains instructions on importing the MySQL database.
- dataset_tsv
  - This directory contains .tsv (tab-separated-values) files containing the data sets to import into each table
- test_files
  - This directory contains .expected files used by the db_import.sh file to automatically check whether or not a table after data import contains the expected number of non-null rows.
  - The data in a .expected file was obtained through the following command in the extracted FoodWasteTracker_MYSQL/ directory after manually

creating a table, importing the data, and verifying the success.

- mysql -u USERNAME --password=PASSWORD --database=food_waste_tracker -e "select count(*) from TABLE_NAME;" > test_files/TABLE_NAME.expected

2. Web Server

Refer to figure 18 for the directory tree.

- FoodWasteTracker_MySQL.tar.gz
  - Compressed tar archive of the FoodWasteTracker_SQL directory containing the database import files and script as specified in 4.2.1 Database.
- index.js
  - This script handles serving pages to visitors, taking inputs from visitors, querying the MySQL database, and performing simple statistical/extrapolation calculations.
- package.json
  - This defines the dependencies required by this web server.
- README.md
  - It contains instructions for installing and setting up the web server and database
  - It is also created in GitHub's flavor of markdown.
- views
  - This directory contains all of the webpages sent to the public.
    - about.html
      - This is the about page.
    - index.html
      - This is the home page.
    - sources.html
      - This contains links to all externally linked images used by this website.

        - This also contains a link to the free graphics resource website that is the source of the images in public/.
    - stat.ejs
        - This is the statistics page that would be populated with visualizations from calculations done based on the user's inputs on the home page.
- public
    - This directory contains all the components to generate what will be sent to the public.
        - *.png
            - These are image files used in the statistics page.
        - ejs.js
            - Embedded JavaScript templates and functions to generate HTML code.
        - favicon.ico
            - This is the website's main logo show in a browser's tab.
        - main.css
            - This styles file defines the format and style theme for the pages in the website
        - stat.js
            - This script defines how to programmatically populates the statistics with visualization based on calculated statistics.
        - wastelogo.svg
            - This is the website's main banner

3. Video

Refer to figure 6 for the storyboard.

- Food Waste Statistics - Video
    - https://www.youtube.com/watch?v=2nSkqDrPZeQ
- Food Waste - Video
    - https://www.youtube.com/watch?v=6pqfRlP0eEk

4. Files for Documentation
    - FoodWaster_report.docx
        - This is the project report in an editable, Microsoft Word format.
    - FoodWaster_report.pdf
        - This is the project report in a read-only, PDF format.
    - FoodWaste_presentation.pptx
        - It contains slides used in the presentation of this project in an editable, Microsoft PowerPoint format.
    - FoodWaste_presentation.pdf
        - It contains slides used in the presentation of this project in a read-only, PDF format.
    - Foodwaste_stats.mp4
        - This is the first of the two videos featured on the website.
        - Its purpose is to give users statistics on wasted food.
    - Foodwaste_psa.mp4
        - This is the second of the two videos featured on the website.
        - Its purpose is to urges users to not waste food.
    - FoodWasteTracker-master.zip
        - This is the .zip archive of all source files needed to host this website on a server.

V. Development Progress and Remarks

As was expected, there were issues that cropped up during development. For some of the issues, a solution was found and is currently implemented. For others, a solution would require future work. Some future work possibilities come from just possible ways to improve the product. In many of these cases, at least some lesson in web development was learned. Ultimately, all of the outlined objectives were achieved (Table 1).

In order to have a smooth user experience on the front-end, few user inputs required is preferred. The database is built on top of two independently gathered datasets, so they don't naturally form perfect relations between each other. A rough attempt was attempted, indirectly relating food prices to food nutrition through food groups. However, this would have required additional front-end user input to properly retrieve all the data.

And so, combined with these two problems, many tables in the database were largely left unused. However, these tables were kept in the final implementation, because they may be used for building future improvements. One such improvement would be to directly relate food prices to food and their nutritional data.

There was also one major issue that cropped up that concerned interaction between the user and the server through the webpage scripts. Because the visiting user is given a text box to enter values, even characters could have been entered into a field for numbers. This resulted in the server querying and returning error-laced results. This issue was solved by sanitizing and validating user input from these textboxes and other locations in the website where such issues occurred. Doing so improve the user interaction experience and reinforced the web site's security as well. However, the front-end could still be improved by allowing the user to input different weights other than just grams, and output statistics in other weights as well.

In the end, each of the milestones were reached by the project timeline to produce a product. The final product fulfilled the goals and objectives as set forth at the beginning of this undertaking. It is true that the design, demonstration and testing of the product was done assuming that it would be a standalone website. However, doing so meant that the final package could be used as was expected, as a standalone website, or be integrated into other, existing websites of the client's choosing through the products individual

parts and modules. Additional details may be found in the Appendices, followed by supporting figures and tables.

VI. Acknowledgements

- Client: Susan Chen schen518@vt.edu
- Instructor: Dr. Edward A. Fox fox@vt.edu

References

1. Beltsville Human Nutrition Research Center. USDA National Nutrient Database for Standard Reference. Sept. 2015. Web. https://www.ars.usda.gov/northeast-area/beltsville-md/beltsville-human-nutrition-research-center/nutrient-data-laboratory/docs/usda-national-nutrient-database-for-standard-reference/. Accessed 18 Feb. 2017.

2. Bootstrap. “Bootstrap - The world’s most popular mobile-first and responsive front-end framework.” Bootstrap. N.d. Web. getbootstrap.com/css/ 07 May 2017.

3. Chen, Susan. "Food Waste Literature Review." Thesis. Virginia Tech, 2017. Print.

4. “CSS | MDN.” Mozilla Developer Network. N.d. Web. https://developer.mozilla.org/en-US/docs/Web/CSS 07 May 2017.

5. “D3.js - Data-Driven Documents.” D3. N.d. Web. https://d3js.org/ 07 May 2017.

6. Engel, Kai. *Moonlight Reprise*. Southern City's Lab, 2013. *Free Music Archive*. Web. 17 Mar. 2017. http://freemusicarchive.org/music/Kai_Engel/Irsens_Tale/Kai_Engel_-_Irsens_Tale_-_04_Moonlight_Reprise.

7. “Flickr Services.” Flickr. N.d. Web. https://www.flickr.com/services/api/ 07 May 2017

8. Foundation, Node.js. "About Docs." Node.js. Joyent, Inc, n.d. Web. 06 May 2017.

9. “How to make Chart Animations for Statistics using Sony Vegas.” Youtube, uploaded by NorthernLights, 12 September 2014, https://www.youtube.com/watch?v=1LUti91Cbec&t=328s

10. “HTML | MDN.” Mozilla Developer Network. N.d. Web. https://developer.mozilla.org/en-US/docs/Web/HTML 07 May 2017.

11. “Javascript | MDN.” Mozilla Developer Network. N.d. Web. https://developer.mozilla.org/en-US/docs/Web/JavaScript 07 May 2017.

12. “The Manhattan Project HD1080P Time-lapse.” Youtube, uploaded by Cameron Michael, 7 June 2012, https://youtu.be/sCGuSqBjhfE.

13. “This is Tokyo | Tokyo City Time Lapse Ver.2.” Youtube, uploaded by Yuu, 24 April 2016, https://youtu.be/DCTW2DMpyPs

14. “Free Music Archive”. *Free Music Archive*. Free Music Archive, 2009. Web. 17 Mar. 2017. http://freemusicarchive.org/.

15. Todd, Jessica E., Lisa Mancino, Ephraim Leibtag, and Christina Tripodo. Quarterly Food-at-Home Price Database. Apr. 2010. Web. https://www.ers.usda.gov/data-products/quarterly-food-at-home-price-database/quarterly-food-at-home-price-database/#Quarterly Food-at-Home Price Database-1 (QFAHPD-1). Accessed 18 Feb. 2017

Appendix

1. Requirements

1.1. Client Requirements

We spent two weeks fleshing out the specific design and requirements of the project with the client. Initially we planned for a website fully equipped with a user account login system that allowed for people to enter the amount of food wasted after every meal. After some refinement and research, we decided that this model would be much less effective and introduce unnecessary complexity to the project. We found no similar applications that allowed user food waste tracking in the market, and concluded that the majority of people wouldn't be interested in having a food waste tracker tool.

As such, we began brainstorming on a possible method to have an impact on users. We opted for a public-service-announcement oriented website that would show the user how the cost for the food they wasted in their last meal could add up over time.

After settling on this model, we drafted a list of requirements with our client. These requirements are listed as follows:

- Interactive website that can be accessed from modern Web browsers on both computer and mobile devices
- Informative video educating users on facts about food waste and prevention
- Easy to use tool that allows users to input the different food amounts they have wasted
- Visual that shows the user the impact of their food waste based on their input in the previous step
- Information page that shows users how much total food waste has been tracked by the website, and the impact of that food

1.2. Project Requirements

In order for users to input the foods they wasted, we need to offer an easy input option that minimizes the steps it takes for users to look up their food items and come up with an estimate of the food mass they threw away. We have found a database of store bought food items with varying volume to weight conversions. By extracting around 200 common food items, we'll allow users to easily click through the food groups and select foods that have weight estimations already calculated. In practice, we'll ask for users to input the food they threw away in their last meal and offer them an array of common food items. After selecting a food item, we ask them to estimate the volume thrown out, which allows us to estimate the weight of food wasted through our database.

After collecting the total amount of food a user has wasted in their most recent meal, we use simple extrapolation and arithmetic to show the user how much they potentially waste in a given period of time (e.g. a month or year). All of this will of course be based on assumptions that the user's input represents the average amount of food they've wasted.

After a user has entered the amount of food they've wasted, we add their entry to an internal database. On a separate page, we will display statistics of how much food waste we've tracked from all site visitors, and what the food could have gone towards or the monetary value of all the food thrown out. This page will be viewable by users regardless of whether or not they entered data or not.

1.3. Services

1.3.1. End users

End users will be provided the means of discovering just how much food waste affects themselves and why one should be aware of food waste. Presented with this sort of information visually, the users will be educated by this website so that they may better manage their own habits. Web interaction with the user will be pleasant and smooth to ensure engagement. Visual information presented will be aesthetically appealing. Finally, resources will be provided for further information about food waste if the user wants more information. Ultimately, users are expected to have the following information:

- Meal broken down into:
    - Type of Foods (meat, fish, grain, etc.)

- Weight (grams)

1.3.2. Researchers

As a potential extension to the project, we also discussed adding an internal dashboard for researchers to use the data collected by the website for their studies. Such a tool would require IRB approval so we decided to make our primary focus the core interactive and informative section of the website as outlined above. Depending on our timeline and if we finish early, obtaining IRB approval and building out an internal dashboard could be a secondary objective. Maintainers or people interested in extending this project in the future could also seek to build the internal dashboard.

1.3.3. Maintainers/Extenders

Those in the future who take up our project can have a baseline set of expectations. These will be:

- Modularity - We will keep services and methods properly decoupled, so that reuse will not only be encouraged but intuitive for future maintainers/extenders.
- Maintainability - We will ensure that the codebase is properly documented, so that those who wish to add on can do so with relative ease. We will also set a style standard as well, ensuring it is compliant with the industry. This will also set the standard for future maintainers/extenders to uphold.
- Familiarity - We will use popular, proven technologies in order to power our service. Not only is there a wealth of resources for these technologies, so that any problems can be easily addressed by extenders. But also there is a good chance that these people are already familiar with these technologies.

1.4. Wireframes

The website will have 3 pages: one for the entering in of the food, one for the display of how much money was wasted, and finally a global statistics page. The homepage for the website will be the first page that the user will encounter on access (Fig. 1). The input required from the

user would be the type of food that the user had wasted, the amount of food wasted in grams, and confirmation of the input through a submit button. The second page displays how much money was wasted as a result (Fig. 2). The third page is a global statistics page (Fig. 3). This third page shows how much money the user has wasted or would have wasted over a period of time.

1.5. Backend Design

1.5.1. Database design

1.5.1.1. Data Organization

As seen in the entity-relationship diagram, the planned database can appear fairly extensive, especially if the full range of nutrient attributes were to be fully enumerated (Fig. 4). However, the schema for all attributes in this database can be considered in terms of three discernable sections. The three sections are constrained together only by a single point on the "provides" weak relation. The full range of attributes were decomposed into relation schemas to prevent data loss and duplication in a production environment.

The motivation behind designing around these sections is that there appears to be no single, openly available data-base/-set that synthesizes both the needed food prices and food nutrition data. However, two separate sources of data were found. Data on food prices will be derived from a USDA's Economic Research Service study (Todd, Jessica E., et. al., *Quarterly Food-at-Home Price Database*). Data on food nutrition will be derived from a USDA Agricultural Research Service study (Beltsville Human Nutrition Research Center, *USDA National Nutrient Database for Standard Reference*). Finally, given that users' anonymous data would be collected, for, currently, non-research purposes, a logging feature would also be included in the database. And so, the three sections in the database's schema would be for information on specific foods' nutrition, unit prices for various food groups, and logging for user input.

As suggested by the fact that two separate services of the USDA produced this data, there are certain types of data that conceptually appear identical, but are indexed and must be worked with separately. Had both services conducted their

studies using the same variables, or had the dataset with food nutrition classify its foods under subgroups instead, the two sets could have been related more directly. Nevertheless, it was possible to interconnect the datasets into a single database, represented in a decomposed form of various relation schemas, as shown in the following chart of functional dependencies (Table 2).

First, the Entity-Relationship diagram was drawn according to the needs of this project. Then, the functional dependencies evolving from the relations were derived by, for each entity or relationship, taking all associated primary key values as the input and all other values as output. Afterward, a second route to build a set of relations was taken from the ground up, through the decomposition of a single large schema housing all of the attributes constrained by the desired dependencies. Finally, the functional dependencies from the two approaches were compared, where the first route's design was compared against the theoretically more robust design derived from the second route.

As for the second route, the minimal functional dependencies cover, i.e. the canonical cover, is the least number of relationships/constraints specifying how one set of values implies the existence of exactly one other set of values for an entire schema. All of the attributes for the entire database were considered as a single schema at the start. Then, an attempt was made to put the database into BCNF (Boyce-Codd Normal Form) by decomposing the single, large relational schema into smaller schemas as per rules defined by BCNF. However, an attempt to decompose the BCNF violating relation as currently defined for the "user_input-chooses" weak entity resulted in a loss of the following functional dependency:

$$SubFdGrp\_ID,\ Mkg\_ID,\ Year,\ Quarter \rightarrow UnitPrice\_Avg,\ Standard\ Error$$

Therefore, restarting from the top, an attempt was made to decompose the single, large relational schema defining the database into multiple relations so that they conformed to 3NF (Third Normal Form), instead.

As seen in the the functional dependency charts, there are several locations in the ER diagram where there are more, smaller

dependencies than there were in the same location in the derived canonical cover. Effectively, those smaller dependencies are, together, equivalent to the single, adjacent functional dependency when Armstrong's axioms are applied.

Ultimately, the design of the database will opt for the more fractured version that was derived from the ER diagram. The reasoning behind this is twofold. Firstly, the separation of various elements provides a more intuitive relationship between various tables that will be made from these entities and relations. This would better help those who will take up the project in the future to grasp the semantics behind the relationships. Secondly, this design provides for a more modular approach, should there ever be a need to make changes to the entities/relations/tables.

1.5.2. Usage

As previously mentioned, the database is designed with three major sections in mind, servicing what should be the three most common queries: food (group) unit prices, food nutritional information, and logged user inputs. There is a minimum amount of input information required to retrieve a specific, meaningful piece of data from a query. The required input would be all attributes on the left-hand side of a functional dependency, which would then return at least one attribute on the right-hand side of the same functional dependency. For example, to obtain the unit price of a piece of food, that food's food group ID number, ID number of what marketgroup, at what year, and at what quarter of the year the data originates are all required; {SubFdGrp_ID, Mkg_ID, Year, Quarter} are required to determine a unique {UnitPrice_Avg}.

1.6. Web Server Design

The web server simply serves as an intermediary between the database and the pages the client is served. At this point, it will most likely be an Apache web server that also serves as a Node.js server. Clients will request information from the web server, for example certain foods, and how much money one loses from wasting that amount. Since this information is sitting in the database, it must be served from there. However, to utilize good design in order to enforce modularity, maintainability, and readability, we will be utilizing the

MVC model. By decoupling the database from the web server, we can easily switch between different technologies without much hassle.

The Web server's other functionality lies in doing computational work i.e.:

- Calculating the amount of food waste from what the user said and the prices in the database.
- Extrapolating said data in order to show how much food waste could potentially occur in a year.
- Calculating statistics from what all the information clients have given us, and putting it into parsable format for the client to display.

1.7. Front-end Design

1.7.1. User interface

The goal of this user interface is to be as simple and as easy to use as possible. We do this by having a minimal number of pages and a minimal site layout. Additionally, the website is here to accomplish one goal: to help track the amount of money wasted through food waste. The achieve this goal the user will be able to navigate the three different pages (Fig. 5).

We will accomplish building this website most likely by using Bootstrap, HTML, JavaScript, and CSS (Bootstrap; HTML; Javascript; CSS). In addition, we will use the Node.js server to communicate data between pages. The HTML will be used to structure the various pages, the CSS will be used to beautify the HTML, the Bootstrap will allow for further beautification, the Javascript will be used for basic website functionality, and finally Flickr will be used for image generation corresponding with a selected food.

1.8. Video design

The video will bring awareness to food waste by targeting youths and adolescents. With this in mind, the video must be engaging and modern. We will be taking a first-person view approach to our main homepage video that introduces the issue of food waste. Target time of

the video length will be about 2 minutes. Tools include a GoPro and iMovie, both provided by Innovation Space.

As the storyboard shows, the audience will follow scene snippets of a wasteful person while they are cooking, eating, cleaning, etc. (Fig. 6). This will be followed by a scene of dumping the wasted food into a trash can. The video will alternate between the two scenes for about half the duration of the video and then will switch the wasted food that is being thrown away to money. The wasteful person will start to throw away money, representing the wasted food. This representation will show that wasting food is literally throwing money away.

2. Implementation

2.1. Backend Implementation

2.1.1. Database Implementation

2.1.1.1. MySQL Database

As previously mentioned in the Requirements and Design Report, this database will be created using MySQL. The database will contain the following tables with the following columns as seen in Table 3.

2.1.1.2. Deliverables

2.1.1.2.1. Contents

The final database product will be a package of files with several directories' worth of files. The top-level directory would include two files. The first file will be a bash script to automate the installation of the database. The second file will contain some of the SQL queries required to perform the first half of that installation, i.e. the creation of the database. The rest of the SQL queries in that same file would perform the second half of the installation by importing data in CSV files.

There will then be two directories of CSV files. These directories be used to populate the tables formed within the MySQL database. The source of those CSV files will be the nutrients and prices spreadsheets from two separate studies done by the USDA. The nutrition spreadsheets come from the USDA *National Nutrient Database for Standard Reference, Release 28* (Beltsville Human Nutrition Research Center, 2015). The prices spreadsheets come from the *Quarterly Food-at-Home Price Database* (Todd, et. al., 2010). There will be some modifications to the organization of the original spreadsheets so that their columns match columns in

the database's tables. There will also be subsequent modifications to the resulting, exported CSV files to remove extraneous characters in the data entries that may cause issues during the import, such as double quotation marks around price values.

2.1.1.2.2. Usage

A readme file will be provided in the installation package to help the user perform the basic setup.

2.1.1.2.2.1. Importing the Database

The importing of the database will occur through the execution of a single bash script (Fig. 7). This assumes that not only is MySQL 5.5 the platform driving the SQL server, the web server itself would be running on a Linux (or, at least, bash-capable) operating system. This also assumes that, to ensure optimal performance, the MySQL server will reside in the same node/baremetal as the web server. To use the bash script, the user must specify a username. Because MySQL servers may or may not contain a password, specifying a password would be optional.

This script would also allow this database to be remotely imported to the server from another location. This would also require the user to specify the IP address or URL. If the hostname is specified, then the bash script will require the user to also specify the port on which the MySQL server would be listening.

The SQL queries used to import the database will reside in a .sql file (Fig. 8). The queries assume that no such database and/or corresponding, populated tables currently exist in the target MySQL server.

Once the import is completed, the import package need not stay on the server, if the import was done locally.

2.1.1.2.2.2. Querying

Queries will be passed to the MySQL server through the web server using the web server's MySQL client. The client provided by the framework powering the web server should additionally, automatically format the output to the desired form for further processing by the web server.

2.1.1.3. Testing

The scripts and files will be created and edited on Microsoft Window's Linux subsystem. MySQL 5.5 will also be installed on the Linux subsystem for testing whether or not the database is properly populated with all the desired datasets as a result of the queries. Select tuples and data points across all of the original spreadsheets used to populate the database will be queried for, to obtain reasonable assurance.

Afterward, the files will be moved to the intended baremetal that will ultimately run the final product. Further testing will occur on the baremetal to ensure to that no major issues will occur on production. There will be additional query testing on the MySQL server, but through the web server's MySQL client to ensure that integration is successful.

2.1.1.4. Integration with Web Server

Once the database has been tested to show reasonable robustness, the file package will be used to install the database onto the baremetal secured by the client. If the baremetal already comes with a more recent version of a MySQL server pre-installed, and there are issues with the queries specified the installation files, then the appropriate changes will be made accordingly to ensure a successful database installation. If there is no MySQL server pre-installed, MySQL 5.5 will be installed. If there is a need, or the client sees a more recent MySQL server version as more beneficial, a more recent one will be installed instead.

2.1.2. Web Server Implementation

2.1.2.1. Technologies Utilized

For this project, we are utilizing Node.js to serve as the web server. Conventionally, one would have a HTTP web server that would serve pages from the backend service of your choosing. Note that each request to a HTTP server is served on a different thread.

Node.js operates on a single thread, which might seem worse. However, Node.js is event-driven, so it waits for an *event* before it does anything. This also makes it extremely efficient, so long as it is coded properly. (Foundation, Node.js.)

2.1.2.2. Architecture

For a Node.js application to be coded properly, it must be understood what keeps Node.js so efficient. Node.js does not work well with computationally intensive tasks, but it does work well with I/O. To overcome this issue, we can pass computationally high tasks to another thread. This way Node.js can keep its I/O thread free for events.

We will set up a HTTP server that is created by Node.js, that will serve the front-end pages that we have created. For the calculation of the food waste, this will be done on a separate thread. This way, the mono-thread will not be blocked, as this is the biggest issue.

2.1.2.3. Deliverables

We will have a web server that is connected to the database, being created by Node.js. This server will send properly configured HTTP responses along with the HTML/JS to serve as the front end.

2.2. Front-end Implementation

2.2.1. Food Logger Page

This is the front page of the website. It will allow the user to select the food for which they want to record waste. After this webpage they can see the analysis page, which will reflect the amount of money wasted based on the food they entered and the

amount of it they wasted. The envisioned page design will both satisfy what the client wants from a colorful interactive website and a convenient tool for the user. This page will incorporate a text entry box that allows for autocompletion based on the food choices from our database, and a grid structure to store images based on the top autocomplete results. The grid structure was incorporated into the website's design after the last report to provide a better visual experience, per client request and will look similar to figure 17.

The technologies that would be most appropriately used would be Flickr API, HTML, CSS, and JavaScript (Flickr). The Flickr API can be used to generate images based on the autocomplete results from user entry into the text box. The Flickr API could provide all of the images that appear in the grid, and seems like the best choice for a dynamic webpage. The Flickr API could be a good choice in the case where there are a lot of images to be generated and you need them to be flexible (Flickr). HTML and CSS will be used for basic styling and formatting as well as JavaScript for basic functionality. JQuery could also further be used to extend the functionality of JavaScript for modifying elements of the front page such as when the user searches a food and an image appears.

2.2.2. Personal Analysis Page

After users enter their information in the food logger page, they will be presented with a personal analysis page that provides them with information about how their food waste adds up. This page will include graphing visualizations that show the user how much their food waste compares to the average American, and among different users of our site. These graphs will be rendered using a d3.js Javascript library using data provided from the server after the user enters their data (D3). Other charts will be displayed on the analysis page that tell the user the monetary value of the food they throw away. Using different images to indicate small, medium, and large amounts, we will show the user how much their food amounts and money will add up over a month, a year, and five years.

The technologies that will be used to power the personal analysis page will be React to handle the data and build the page, d3.js to handle the graphing and charting needs, Bootstrap's CSS library to structure and design the page, and HTML templating. Ultimately, this page will provide data in a visual manner similar to figure 2.

2.2.3. Global Statistics Page

The Global Statistics page is the final page that you are able to see. You are able to see it after the analysis page and you can return to the Food Logger page from here. It will use basic technologies to accomplish the task of presenting the information gathered from everyone who has used this site and presenting it in an organized fashion. This page will feature a table that lists out the following information:

- Food Waste for
    - Today
    - This week
    - This month
    - This year
- Waste Statistics
    - Standard Deviation by day
    - Average per day
    - Trend over the year (increasing or decreasing)

This page as much, as the last page, will use d3.js for all of the graphing needs to present this information in a structured and organized way as well. The first thing users will see is a graph of money wasted as a result of food wasted, extrapolated over various durations of time, up to a year. Following that will be several examples of what could have been done with that money, such as how many other people in other countries may be fed, instead. This page will look somewhat similar to figure 3.

2.3. Video Production

2.3.1. Equipment and Technologies

The video will be captured in first person view with a GoPro Hero 3 and GoPro Hero 4, provided by Virginia Tech's Innovation Space. The videos will be edited in Sony Vegas Professional and iMovie.

2.3.2. Process

Raw footage will be captured over time to collect scenes of cooking, cleaning, and food waste. In order to visually engage the audience, statistics will be animated in an infographic manner. Generic landscape timelapse footage will also be used. There are an abundance of tutorials online of how to use Vegas Professional, how to make the visually animated statistics, and Innovation Space staff to help with iMovie if needed (NorthernLights, "How to make Chart Animations for Statistics using Sony Vegas").

2.3.3. Deliverable

The theme of the short 1-2 minute video is to motivate people to waste less through a public service announcement by showing infographics and the action of wasting food. There will be no narration. Music will be from the Free Music Archive, selecting only instrumentals that fit with the video (Free Music Archive). Statistics to include: (Consumer Reports)

- The amount of food waste costs a U.S. family of four per year: $1,500
- Homes waste 43% out of all the food wasted {Consumer facing businesses: 40%, Farms 16%, Food processing companies: 2%}
- Food Eaten vs. Food Tossed: Meat {48%/52%}, Produce {78%/ 22%}, Milk {80%/ 20%}, Seafood {50%/50%}, Grains {62%/ 38%}
- 63 million tons per year is wasted in the U.S.
- 28% of land is used to produce food that never gets eaten
- 25% of water in the US grows food that's ultimately tossed
- 4% of total US consumption is used for transporting food that goes uneaten

3. Prototype

3.1. Backend Prototype

3.1.1. Database Prototype

The database was created with the file structure similar to that seen in Figure 9 with the exception of the presence of test_files and a README. The main focus of the prototype was to ensure that all of the files in the folder had the capability of automatically generating the desired MySQL schema and fully import the dataset.

3.1.1.1. Import Script

The import script was created as a bash script, as mentioned in the implementation section for database. The import script had a few additional functionalities added that were not mentioned in the design section, but it also was missing a few functionalities as a result of the minimal goal from the onset.

The import script was able to take in the user's username and password, if supplied, to log into the MySQL server for executing the queries in the SQL file. The script needed to be manually made aware of the SQL file by passing the path to it as an argument to the script. The script then parsed the SQL script line by line, concatenating them into a single string that would be passed to the MySQL command-line tool.

Immediately a one major issue was encountered. The queries needed to load the dataset into the created database assumed that the files were present in the machine on which the script was run. The MySQL command-line tool, by default does not allow such an act. And so, this was solved by simply adding the flag `--local-infile` to the string that was to be gradually built into the final MySQL command for the Bash shell to execute.

The script was also designed to fail with various messages, if the user did not provide all of the necessary information like the login password, to prompt the user to rerun the script with the missing information. This

particular function ensured that the user always provided the user's MySQL username. If the password flag was provided, it ensures that an actual password was also provided. The script flag for entering a remote MySQL server's address for remote import was optional. However, if a remote MySQL server's address was provided, then the port number would be required.

3.1.1.2. SQL File

In order to create the schema in the target MySQL server, the creation and load queries were all put into a single SQL file (Fig. 8). The SQL creation queries each had line of catch code to create a new database or table only if one by the same name does not already exist. All of the tables mentioned in the implementation section were created by these queries (Table 3).

Loading the data into the tables, at the time, appeared to be doable only either locally or remotely, but not both. And so, the decision was to assume that loading will occur with a dataset present locally in the same machine as the one hosting the MySQL server.

3.1.1.3. USDA Spreadsheets

The source for the dataset, the spreadsheets from the two USDA studies, have slightly different column names than what were decided for the database, and in a different configuration. Accordingly, several spreadsheets, such as `nutrition_foodDescriptions.gsheet`, had to be broken up into spreadsheets with columns that perfectly matched the columns of the tables in the database, such as into `foods_table.gsheet` and `unit_conversion_table.gsheet`. Other spreadsheets only had certain values implied by their filename, such as food subgroup ID information. Hence, for example, `prices_grainsanddairy.gsheet` needed the food subgroup ID to be explicitly entered into each row.

3.1.1.4. Dataset

The dataset used for importing the data into the MySQL database are in the tab-separated-values format. The comma-separated-values format was initially

considered, but some values in the spreadsheets were surrounded by double quotes, while others were not. This inconsistency created a slight challenge that was further complicated by the CSV format, where all double quotes had to be somehow automatically deleted from the thousands of entries. Consequently, tabs were used as the column-delimiting character to circumvent this issue. The TSV files were created, one corresponding to each table in the MySQL schema; all double quotes were deleted through the use of the substitute command in the VIM text editor.

3.1.2. Web Server Prototype

3.1.2.1. Web Server Prototype

The web server was initially created to serve as a location that a user could have visited, and would have served a single page. In the earliest form of the prototype, everyone was used their own web server to test the functionality on their pages. A figure below has been created to demonstrate.

- Step 1 – In Figure 11, the client requests a webpage from the first server, which is the index.
- Step 2 – The client receives the index page, which allows the user to select the food groups.
    - The user will submit both a food sub-group and the mass.
- Step 3 - When selected, the user submits a get action to another web server.
    - The web server prepares data, but in the earlier phases of prototyping, it was hard coded in.
- Step 4 – The web server sends back a webpage with the necessary statistical information.

3.1.2.2. Technologies implemented

While the multi-server setup was good in terms of having quick reiterations, i.e. people could make a change and then see the results, it was a poor foundation on which

to lie on.

While this setup did serve as an end to end prototype, much of the time invested in this prototype phase was implementing it in Node.js. Node is a server side platform on which many applications run, and the minimum amount of time it takes to "hit the ground running" is quite low.

In order to have a functioning Node js program, three Node modules are necessary. In this case, the web server was made with:

- The Express module, which is the foundation for the web server, and allows for easy handling of create, read, upload, and delete operations.
- The MySQL module, which allows one to connect to a MySQL DB and submit queries.
- The body-parser module, which allows one to easily receive specific data from the buffer stream from the get action.

3.2. Front-end Prototype

3.2.1. Front Page prototype

The design for the front page of this website had developed over time and per the wants of the client (Fig. 12). The prototype reflects all of this change and the initial design requirements.

The design is simple right now and is awaiting client approval before agreeing upon a style on top of the components that are already apart of this website. This website features a food search bar where you enter in some characters and the suggestion bar populates an autocomplete along with corresponding pictures below. There is a max of 8 that are shown at any given time. There is also a field where you can enter the number of grams of that category of food.

The search bar's main feature is its autocompletion. This autocompletion is what will allow the user to easily follow the database's categories and show the appropriate images for the user so that it can help them select what they want in case a food item confuses them (such as solid fat, which is mainly referring to

butter). It autocompletes based on a jQuery autofill method and the autofill comes from a static list within the code.

The images are generated as the user types characters into the search bar. They come from a static image dictionary embedded within the code itself; this was the optimal solution as opposed to loading from a file from the server because it would be an expensive operation to be doing that repeatedly for a dataset small enough to be represented statically. The dictionary has a key-value pair of categories and image URLs.

These data fields will be sent in a get request to the server as parameters and will be used to calculate the corresponding monetary food waste. The next page that would be displayed after this would be the results page that shows you how much you have wasted with the data you have entered in the previous page.

3.2.2. Statistics page

When developing the initial statistics page, we chose to implement functionality first before worrying about design and styles. After the user selects their food groups and fills out how much they have wasted in the past day, the statistics page will be served with the calculated extrapolated food waste data from the backend. While less complicated than the front page in that this statistics page doesn't rely on complex user interaction, a comparable amount of care will still be needed while implementing this part of the prototype to ensure that our message is well received by the user.

As seen in Figure 13, the initial prototype simply spewed out data statistics obtained from the backend. This design was more of a placeholder than a final prototype, but was used by us mainly to determine what statistics the user cared about seeing. After some user testing, we iterated on our design to include a more graphical representation of the food wasted. While this design was still generally rough, it was due to the fact that we relied on many small iterations of changes from working with multiple users to see what worked and what didn't.

Figure 14 represented a shift from our end to beginning to focus on design. We realized at a certain point that no matter what statistics we choose to show the user, presentation played a key

part in getting our message across. We initially experimented with working strictly with raw HTML, JavaScript, and CSS before using libraries like Bootstrap CSS and D3 for document styling and graph generations, respectively. This was where the website was before we met with Susan to decide how to further proceed.

3.3. Video Prototype

The video prototype was 1 minute and 41 seconds long and showed the action of wasting food. Through a first-person view, the video progresses and shows multiple scene cuts of opening the fridge, picking out ingredients or leftovers, cooking, and trashing varying amounts of food. The raw video footage was filmed with the Go Pro 3, Go Pro 3+, and Go Pro 4. Sony Vegas Professional 11 was used to edit, compile, and render the video prototype. The video prototype does not include narration audio, but it does include an instrumental soundtrack from freemusicarchive.org. The background music used was *Moonlight Reprise* by Kai Engel, because of its use of piano (Engel, 2013). The music intensifies slightly as food is being wasted in the video. The video ends with the message “Don’t Waste Food”.

4. Initial Refinement

4.1. Backend Initial Refinement

4.1.1. Database Initial Refinement

Once the import script was found working with the rest of the files in the directory, a few initial changes were made to the following sections:

4.1.1.1. Import Script

The prototype was developed with only a local import in mind. The remote import features, such as supplying a host and port flag pointing to the target remote MySQL server to the script, were removed. Additionally, a much simpler way was found to execute the SQL creation and load queries in the SQL file. The line-by-line parsing function was removed. Afterward, the bash script looked like Figure 7 with the exception of the presence of the testing portion.

4.1.1.2. USDA Spreadsheets

These source spreadsheets were only used to generate the TSV files. That is, the import queries of the SQL file only make use of the TSV files. Their inclusion in the import package only serves to take up space. Therefore, they were all deleted from the package.

Once these changes were made, the structure of the import package transformed into the one as depicted in Figure 10.

4.1.2. Web Server Initial Refinement

The web server was refined in numerous ways, in order to streamline it and make it more stable. One of the most significant changes that was made was that there is only one current web server on which everything runs. This not only lessens the number of servers involved, it also reduces the overhead.
For example:

- The back end has no longer to keep track of which user asked for what data, as a web server can maintain a single session.
- Less data has to be passed through another web server, which reduces latency.
- Using fewer web servers also reduces the potential attack vector of our system.

We also included the use of the MySQL database, as opposed to hard coding values into the web server application. This was a great boon, as we no longer had to deal with static values. We could dynamically gather the values we wished, and made the implementation abstract. And so, any database could be hooked up to our system with little modification.

4.2. Front-end Initial Refinement

4.2.1. Frontpage refinement

A lot of refinement had occurred with the intervention of our client, Susan. Here are some of the things she focused on about the prototype that were fixed:

1. Renaming the fields

This had been changed because the way the field names were described was inappropriate. It was an easy fix to simply rename the fields.

2. Create a design for the website

The prototype looked very plain. There is now a central theme to the website overall and the flow of the website is still appropriate. The page looks better.

3. Add mobile support

A lot of the page was already good for mobile support. The only thing that seemed necessary to change for the sake of being mobile was the location of the box where you enter the mass of the food you are wasting. It was moved to beneath the search box.

4. Change the black boxes to something more visually appealing

The black boxes were substitutes for search results that weren't existent. These were simply changed to clear boxes that fits in better with the theme of the website as well. They are more visually recognizable as an empty data set.

The result of these changes is shown in Figures 15 and 16, which is believed to be close to the final version of this front page.

4.2.2. Statistics Page Refinement

Much refinement also went into the statistics page after meeting with Susan. Much of this was in the form of small visual adjustments and tailoring a webpage that was friendlier toward novice Web users. Susan requested that we opt for a more colorful look with bigger text, but left most of the actual implementation details open for us to decide. As a result, we decided to build a general template for our website that we could apply to all of the other webpages. We also looked more into D3's libraries to utilize its animation and mobile friendly features. Previously, loading on mobile would render the graph statistics off the viewable area and show very small text. We adjusted our webpage to be fully responsive to the browser window sizes. Finally, we concentrated on adding more fun facts to our statistics page like how many whole chickens, bananas, and hamburgers the user would have wasted over an entire year. Figures 15-17 show the result of our focus on improving the design and experience of the statistics page and the addition of a general web template.

4.3. Video Initial Refinement

The 1 minute 41 second video felt too long, according to our client, and should be shorter and more focused to retain the attention of the audience. Scenes were trimmed or cut to reduce the video length to 1 minute and 29 seconds. Quality of the video should also be improved upon but because the raw footage was captured using different versions of Go Pros, refining the visual quality of the scenes is nearly impossible. Currently, we are exploring video filters or video event fx offered in Sony Vegas to mask the video quality. Because of the general nature of public-service-announcement videos, it was also difficult to include the

statistics in a meaningful context with the initial video prototype, so our client and our team decided to split the information into two videos. The first video should show the action of wasting food, while the second video would show infographics of food waste. Instead of taking the scenes gathered from the Go Pro, the second video will have generic landscape time lapse scenes to give a sense of higher importance. The combination of time lapse scenes and infographics will instill a sense of responsibility in audiences. Most of the statistics are percentages, so an animation of a count up of numbers will be used to show statistics (starting at 0% and counting up to target percentage). The video will end with a message and a dramatic statistic to leave viewers thinking. The target length of the second video is less than one minute so that it is short and to the point.

5. Testing and Further Improvements

5.1. Backend

5.1.1. Database

To determine which areas of the database import process required testing, the steps involved in importing the database were considered. This resulted in the following changes:

5.1.1.1. Import Script

The import process handled by the import script first creates the tables as specified. Next, it imports the data from the dataset into the tables. During a few sessions of manual testing by running the script and dropping the database after, most errors that occur with creating the tables are already printed out by the MySQL command-line tool. However, if the dataset failed to load into the database, then no error message is printed to the console. A testing portion was created in the bash script.

An import was done through the MySQL installed on the Bash on Ubuntu on Windows Subsystem. The tables were manually checked to ensure that the number of non-null rows for each table match the number of entries in each of their corresponding TSV files. Then, a query was done on the number of non-null rows for each table through the MySQL command-line tool, and the results were piped to the corresponding `.expected` file in the test_files folder.

The testing portion does the same query to count the number of non-null rows for each table, and compares that to the count in the corresponding `.expected` file. It does this silently if the script was not run in verbose mode. If the script was run in verbose mode, the MySQL command-line tool also runs in verbose mode. However, if the count of non-null rows for any table does not match the expected count, then an error message will be printed to the console regardless of verbose mode or not.

5.1.1.2. SQL File

Issues that may occur in the SQL queries are either already handled by the MySQL command-line tool during creation or the testing portion of the bash script during data import.

Further refinements to the bash script would involve implementing the ability to load the data from a remote machine. This would involve re-including the ability to pass a host and port flag to the bash script. This would also require the separation of the SQL load queries from the SQL create queries. This is because a separate set of load queries would be required for load that occurs remotely. Logic in the bash script would have to be included to determine which SQL file with the load script, remote or local, should be used depending on whether the user wants to use the script to load data locally or remotely.

To test this new feature, a separate machine would be needed to attempt to load the data from the first machine.

5.1.2. Web Server

5.1.2.1. Web Server Testing and further Improvement

Several tests were run, to ensure that the server did not fail. One test was to ensure that our inputs could handle different data. In the beginning, it failed if one to zero inputs were put in. Thus we had to validate input, as we also run the risk of a SQL injection attack, or a cross site scripting attack. To prevent attacks like these, we decided to sanitize our inputs. This was paramount in order to get closer to the final product.

We also made tests to see how many threads the Node JS web server could handle, just in case we received a larger than usual amount of traffic. Thankfully, Node JS is designed so it can have an enormous number of events running, because of its asynchronous nature. So long as any one event is not computation heavy, Node JS will hardly be affected.

In our case, we have to calculate the amount of food wasted by the client. This calculation can be done client side, as it not confidential or at risk. If this does become an issue, we can spawn another thread to do the heavy computational work, so that the primary Node JS thread is not blocked.

5.2. Front-end

5.2.1. Front Page

The testing of the front page was not as bad as it could have been because its design was appropriate from the beginning. This helped to reduce the amount of potential errors for the future. The front page was tested to be able to handle:

1. Bad data
   It was made sure that the front page could handle and not pass on data that wasn't appropriate such as data that wasn't autocompleted to comply with a database entry.
2. Incorrect data
   It was made sure that the front page could handle incorrect data types as well, such as non-numerical characters in the grams section.

Different aspects of the page were tested at different times so that they could come together in the best, least conflicting way possible. In addition, usability tests were conducted to determine if the front page was usable. It was shown that users responded very well to the frontpage design. The pictures really helped them to not only accomplish the tasks of entering in the correct information, but also to enhance the usability experience in general. The users responded really well to the fact that there were updating pictures every time their search query updated. The users found the interface to be both helpful and aesthetically pleasing. They also found the autocomplete to make it easy for them to pick out what category their food is, where a user might have trouble defining otherwise.

5.2.2. Statistics Page

Testing of the statistics page focused mostly on seeing whether we made an impact on users who interacted with our website. On the implementation side, the statistics page was tested to see if it could handle:

1. Huge data points
   We manually set huge food waste weights in order to stress test our graphing tool libraries.
2. Different devices
   We also loaded our browser on both desktop and mobile devices to see if the images and text loaded properly.

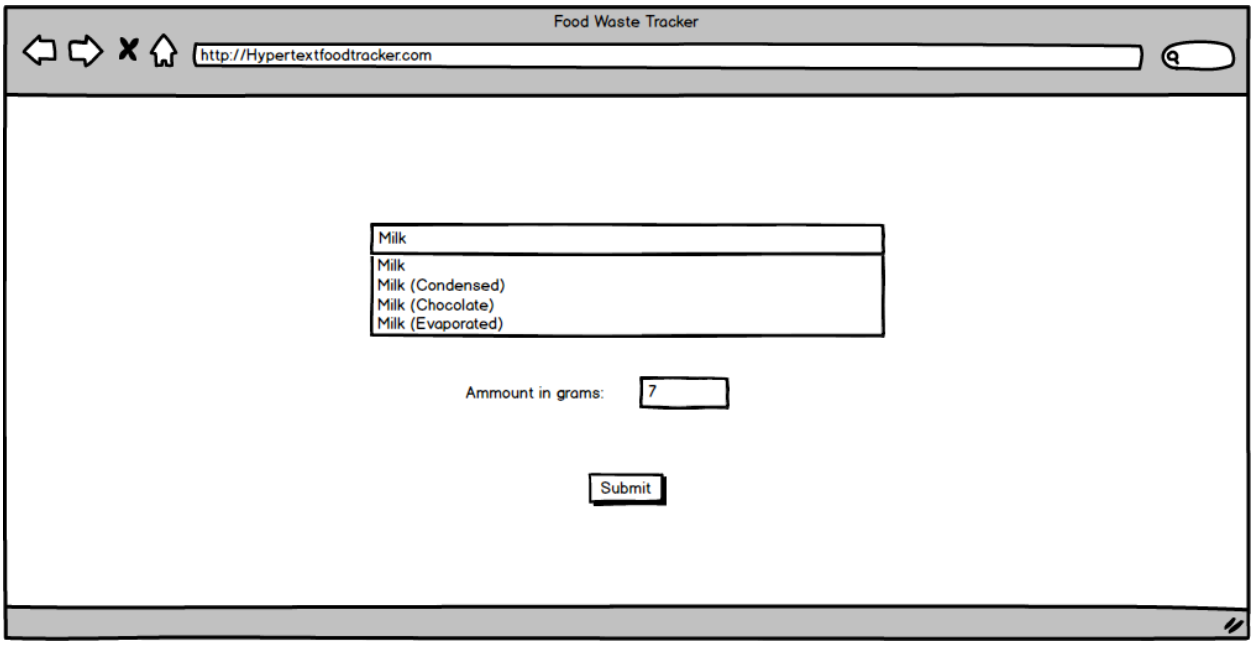
Much of the further improvement to be made is examining the user experience of our website. We've scheduled both meetings with Susan and multiple users of different ages to see how they interact with our website and how they interpret our statistics page that we present to them. Key points that we'll be paying attention to is whether or not our website is easily to use and interpret, whether users understand the statistics we generate, and whether our website provides meaningful commentary for people to change and adjust their food waste habits.

5.3. Video

To determine if the video was effective, feedback was acquired by showing the video prototype to random viewers. An important feedback was that audiences lost focus towards the middle of the video prototype. To further improve the videos, scenes have to be clear in what it is trying to convey while retaining the attention of the audience. The audience should be able to relate or logically process the video's plot. Another feedback was that the ending message was not as impactful as it could be. This could be solved by either displaying an eye-catching statistic or a dramatic scene.

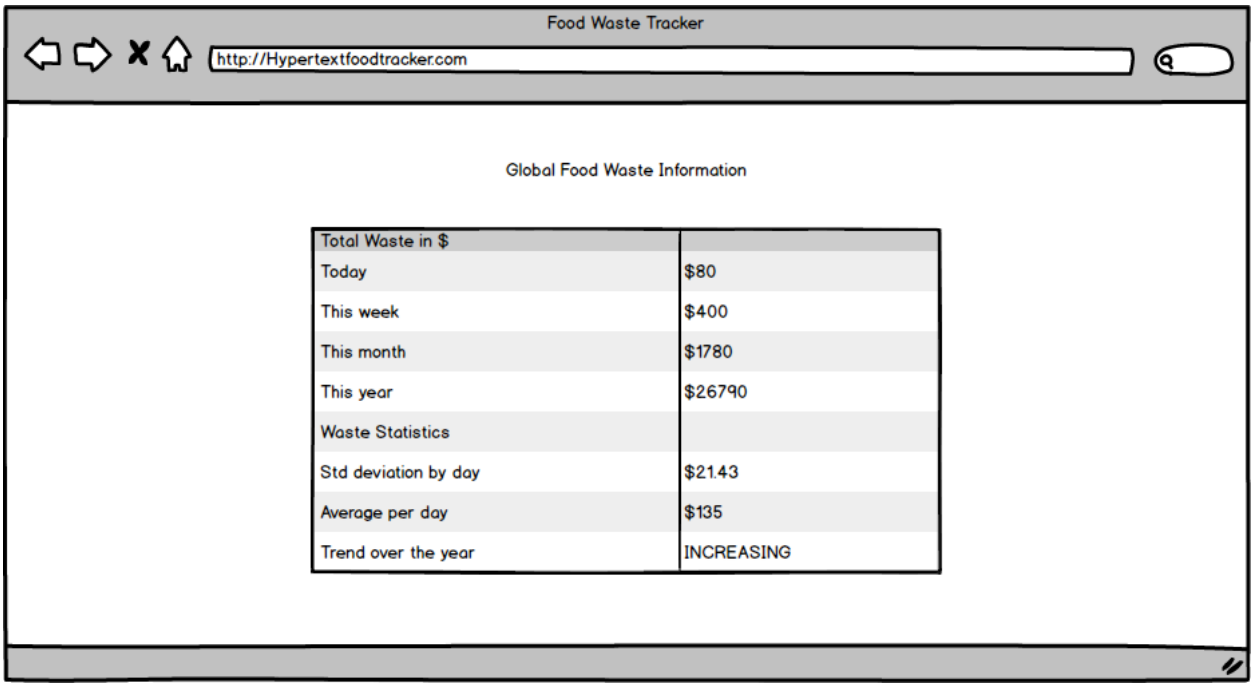
Figures

1. Wireframe, homepage

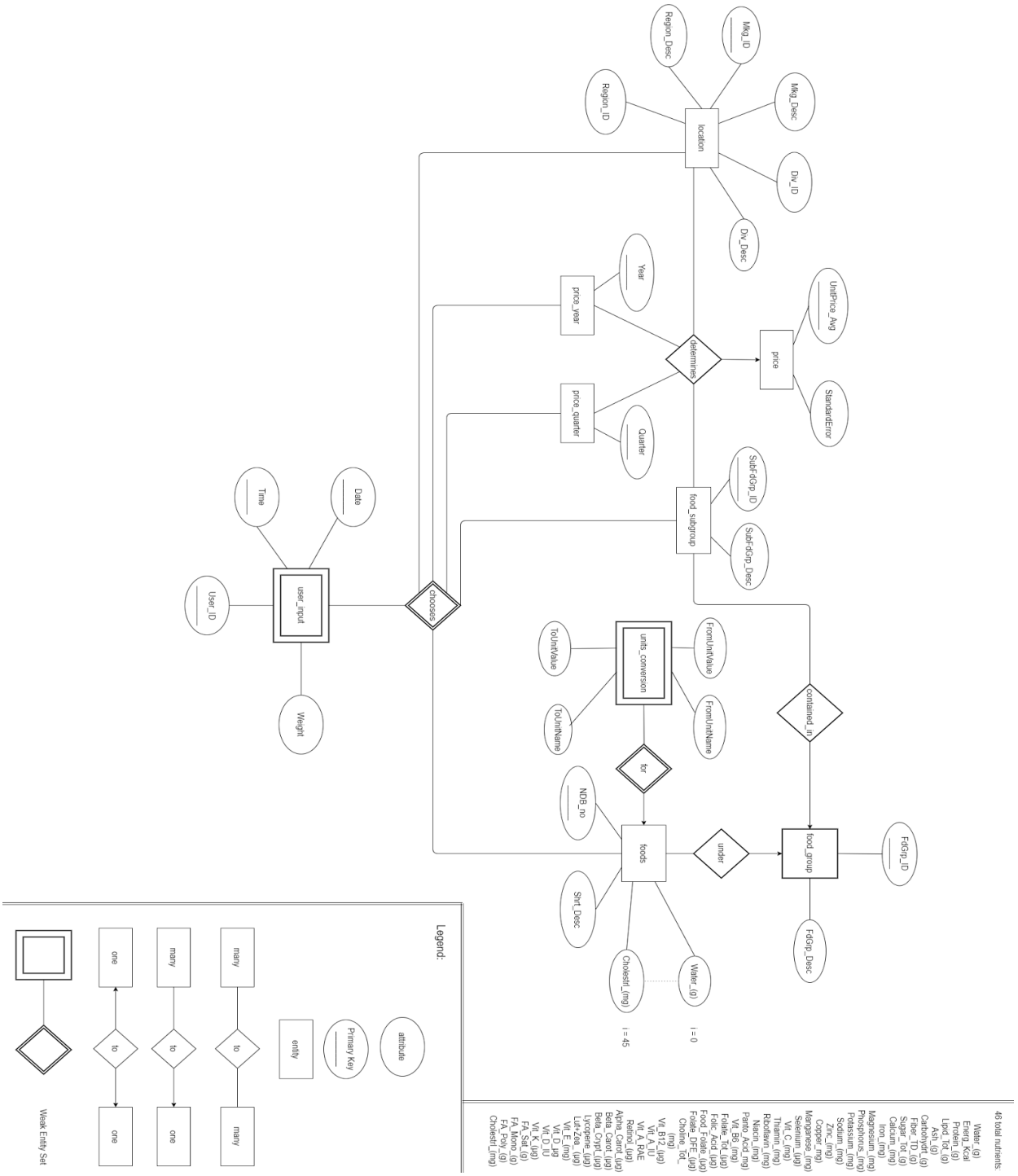


2. Wireframe, food wasted page

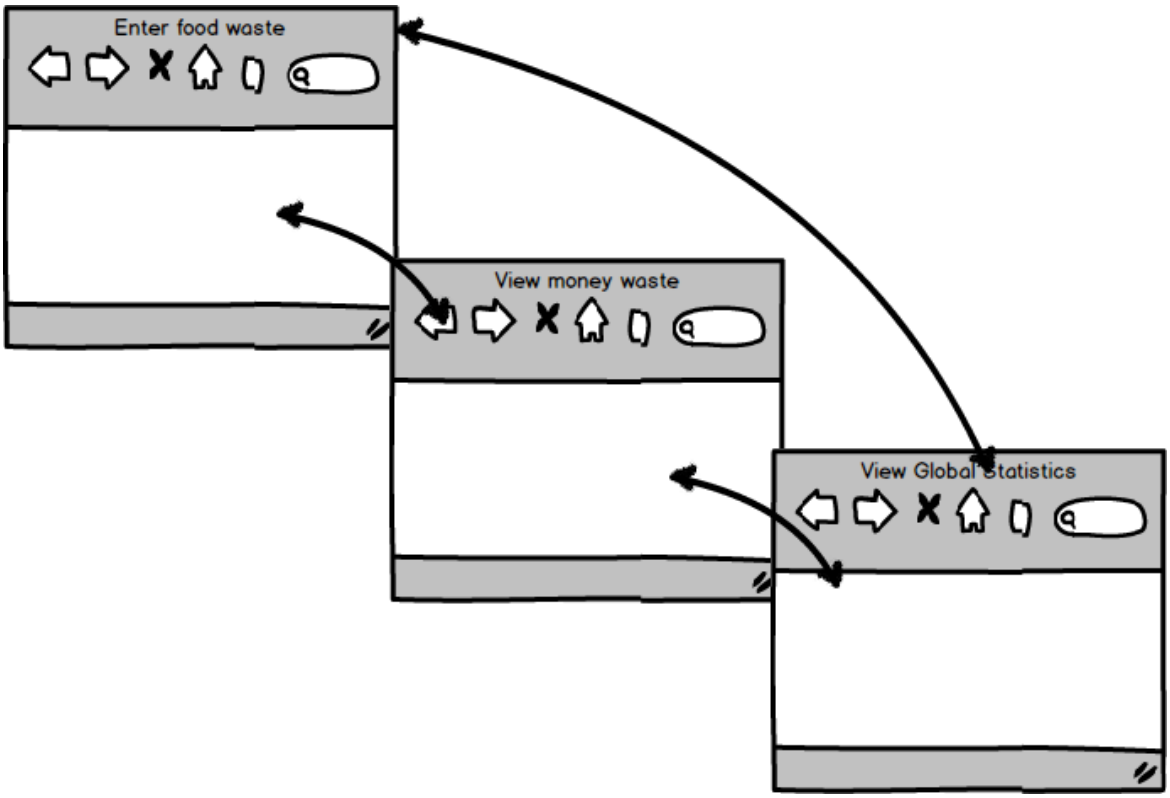
3. Wireframe, statistics page



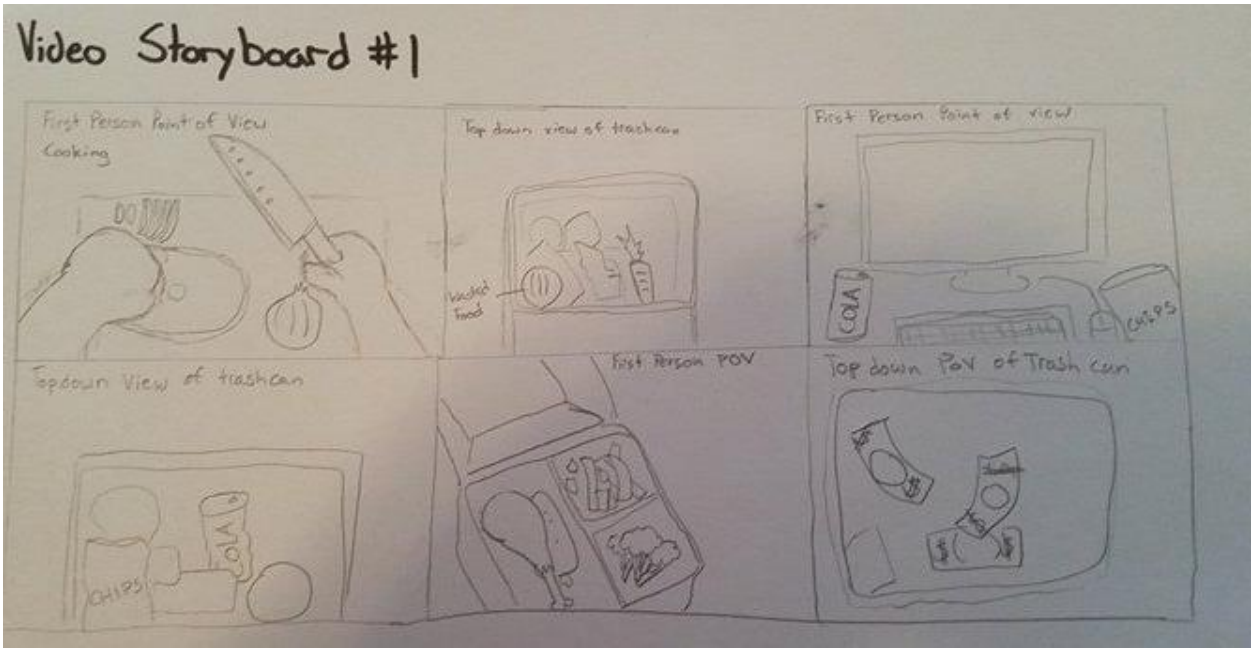
4. Database entity-relationship diagram



5. User interface action flow



6. Video storyboard



7. Bash script, db_import.sh

```
#!/bin/bash

# This script creates the database 'food_waste_tracker' and its
# associated tables.
#
# Then, it loads the database with tab-separated datasets, and
# checks whether all of the tables were successfully populated.
#
# Version   2016.04.09

USAGE[0]="Usage: $0 <FLAGS>"
USAGE[1]="Flags:"
USAGE[4]="    -u <USERNAME>       REQUIRED"
USAGE[5]="    -p <PASSWORD>"
USAGE[6]="    -v                  Creates and imports in verbose mode"
CMD="mysql --local-infile"
VERBOSE=false

# Used to indicate a piece of information the user did not provide
function highlight_line {
   USAGE["$1"]="$(echo "${USAGE[$1]}" | sed -e 's/../**/1')    ****"
}

# Adds the credentials required to login into the mysql server
function load_conn_info {
   if [[ "$VERBOSE" = true ]]; then
      # Runs MySQL in verbose mode
      CMD="$CMD --verbose"
   fi

   if [[ -z "$USERNAME" || "$USERNAME" == "-p" ]]; then
      # The user enabled the "use a password" flag, but did not give one
      highlight_line 4
      printf '%s\n' "${USAGE[@]}"
      echo "$USERNAME"
      echo "$CMD"
      exit 1
   fi

   # The user's username for managing the MySQL server
   CMD="$CMD -u $USERNAME"

   if [[ ! -z "$PASSWORD" ]]; then
      # The user wants to login to an account that requires a password
      CMD="$CMD --password=$PASSWORD"
   fi
```

```
}

# Processes the user's inputs
while getopts ":u:p:v" flags; do
  case "${flags}" in
    u)
      USERNAME="$OPTARG"
      ;;
    p)
      PASSWORD="$OPTARG"
      ;;
    v)
      VERBOSE=true
      ;;
    /?)
      echo "Invalid options: -$OPTARG" >&2
      ;;
    *)
      printf '%s\n' "${USAGE[@]}";
      exit 1
      ;;
  esac
done
shift $((OPTIND - 1))

# Adds the credential required to login into the mysql server
load_conn_info

# Copies dataset to temporary folder for SQL commands to access
mkdir -m 700 -p /tmp/db_fwt_dataset_tsv
cp dataset_tsv/*.tsv /tmp/db_fwt_dataset_tsv/

# Executes the included .sql file
bash -c "$CMD < db_fwt.sql"

# Clean-ups temporary files created for the import process
rm -drf /tmp/db_fwt_dataset_tsv

# Test that all data has been imported correctly
cd ./test_files
if [[ $VERBOSE = true ]]; then
  # Only notifies the user of test in progress when verbose run is enabled
  echo "Testing dataset load success:"
fi
EXIT_CODE=0
# Checks each table's non-null row count
for f in ./*.expected; do
  COUNT_CMD="$(head -n 1 ./$f)"
```

```
  TABLE_NAME=$(basename $f .expected)
  OUTPUT=$($CMD -e "use food_waste_tracker; select $COUNT_CMD from $TABLE_NAME;")
  OUTPUT=${OUTPUT##*$'\n'}
  EXPECTED=$(tail -n 1 $f)
  if [[ -z $(diff <(echo "$OUTPUT") <(echo "$EXPECTED")) ]]; then
    # OUTPUT matches EXPECTED
    # The table has the expected number of non-null rows
    if [[ $VERBOSE = true ]]; then
      # Provides feedback only when verbose run is enabled
      STATUS_PROMPT="'$TABLE_NAME' load:"
      printf "%-32s" "$STATUS_PROMPT"
      printf "[OK]\n"
    fi
  else
    # When the count of rows in a given table does not match
    # what is expected, a message will be printed regardless of
    # verbose run or not
    STATUS_PROMPT="'$TABLE_NAME' load:"
    printf "%-32s" "$STATUS_PROMPT"
    printf "[FAIL]\n"
    echo "--------------------"
    echo "EXPECTED:"
    echo "\"$EXPECTED\""
    echo "--------------------"
    echo "RESULT:"
    echo "\"$OUTPUT\""
    echo "========================================"
    EXIT_CODE=-1
  fi
done

# Exits without error (0) if all the data was loaded, after
# the schema was created properly; exits with error (-1 or 255) if otherwise
exit $EXIT_CODE
```

8. SQL file, db_fwt.sql

```
CREATE DATABASE IF NOT EXISTS food_waste_tracker;
USE food_waste_tracker;

CREATE TABLE IF NOT EXISTS food_subgroup (
  fsg_id INT (5) NOT NULL,
  fsg_desc VARCHAR (50),
  PRIMARY KEY (fsg_id)
);

CREATE TABLE IF NOT EXISTS food_group (
```

```
   fg_id INT (5) NOT NULL,
   fg_desc VARCHAR (50),
   PRIMARY KEY (fg_id)
);

CREATE TABLE IF NOT EXISTS price (
   fsg_id INT (5) NOT NULL,
   mkg_id INT (5) NOT NULL,
   year YEAR NOT NULL,
   quarter INT (1) NOT NULL,
   unit_price_g_avg DOUBLE (10, 2),
   standard_error DOUBLE (10, 9),
   PRIMARY KEY (fsg_id, mkg_id, year, quarter)
);

CREATE TABLE IF NOT EXISTS location (
   mkg_id INT (5) NOT NULL,
   mkg_desc VARCHAR (50),
   div_id INT (5),
   div_desc VARCHAR (50),
   region_id INT (5),
   region_desc VARCHAR (50),
   PRIMARY KEY (mkg_id)
);

CREATE TABLE IF NOT EXISTS food (
   ndb_no INT (5) NOT NULL,
   food_desc VARCHAR (100),
   #46nutrients

   water_g DOUBLE(10, 3),
   energ_kcal DOUBLE(10, 3),
   protein_g DOUBLE(10, 3),
   lipid_tot_g DOUBLE(10, 3),
   ash_g DOUBLE(10, 3),
   carbohydrt_g DOUBLE(10, 3),
   fiber_td_g DOUBLE(10, 3),
   sugar_tot_g DOUBLE(10, 3),
   calcium_mg DOUBLE(10, 3),
   iron_mg DOUBLE(10, 3),
   magnesium_mg DOUBLE(10, 3),
   phosphorus_mg DOUBLE(10, 3),
   potassium_mg DOUBLE(10, 3),
   sodium_mg DOUBLE(10, 3),
   zinc_mg DOUBLE(10, 3),
   copper_mg DOUBLE(10, 3),
   manganese_mg DOUBLE(10, 3),
   selenium_g DOUBLE(10, 3),
   vit_c_mg DOUBLE(10, 3),
```

```
   thiamin_mg DOUBLE(10, 3),
   riboflavin_mg DOUBLE(10, 3),
   niacin_mg DOUBLE(10, 3),
   panto_acid_mg DOUBLE(10, 3),
   vit_b6_mg DOUBLE(10, 3),
   folate_tot_g DOUBLE(10, 3),
   folic_acid_g DOUBLE(10, 3),
   food_folate_g DOUBLE(10, 3),
   folate_dfe_g DOUBLE(10, 3),
   choline_tot_mg DOUBLE(10, 3),
   vit_b12_g DOUBLE(10, 3),
   vit_a_iu DOUBLE(10, 3),
   vit_a_rae DOUBLE(10, 3),
   retinol_g DOUBLE(10, 3),
   alpha_carot_g DOUBLE(10, 3),
   beta_carot_g DOUBLE(10, 3),
   beta_crypt_g DOUBLE(10, 3),
   lycopene_g DOUBLE(10, 3),
   lutzea_g DOUBLE(10, 3),
   vit_e_mg DOUBLE(10, 3),
   vit_d_g DOUBLE(10, 3),
   vit_d_iu DOUBLE(10, 3),
   vit_k_g DOUBLE(10, 3),
   fa_sat_g DOUBLE(10, 3),
   fa_mono_g DOUBLE(10, 3),
   fa_poly_g DOUBLE(10, 3),
   cholestrl_mg DOUBLE(10, 3),

   PRIMARY KEY (ndb_no)
);

CREATE TABLE IF NOT EXISTS unit_conversion (
   ndb_no INT (5) NOT NULL,
   value_a DOUBLE (10, 9),
   name_a VARCHAR (20),
   value_b DOUBLE (10, 9),
   name_b VARCHAR (20),
   PRIMARY KEY (ndb_no)
);

CREATE TABLE IF NOT EXISTS food_food_group (
   ndb_no INT (5) NOT NULL,
   fg_id INT (5),
   PRIMARY KEY (ndb_no)
);

CREATE TABLE IF NOT EXISTS food_subgroup_group(
   fsg_id INT (5) NOT NULL,
   fg_id INT (5) NOT NULL,
```

```
   PRIMARY KEY (fsg_id, fg_id)
);

CREATE TABLE IF NOT EXISTS user_input (
   uid INT (5) NOT NULL,
   date DATE NOT NULL,
   time TIME NOT NULL,
   ndb_no INT (5),
   fsg_id INT (5),
   mkg_id INT (5),
   year YEAR,
   quarter INT (1),
   weight DOUBLE (10, 9),
   PRIMARY KEY (uid, date, time)
);

CREATE TABLE IF NOT EXISTS visitors (
   identifier VARCHAR (512) NOT NULL,
   uid INT (5) NOT NULL AUTO_INCREMENT,
   PRIMARY KEY (uid)
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/food_subgroup_table.tsv' INTO TABLE
food_subgroup
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
   fsg_id,
   fsg_desc
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/food_group_table.tsv' INTO TABLE
food_group
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
   fg_id,
   fg_desc
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/price_table.tsv' INTO TABLE price
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
```

```
(
   fsg_id,
   mkg_id,
   year,
   quarter,
   unit_price_g_avg,
   standard_error
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/location_table.tsv' INTO TABLE location
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
   mkg_id,
   mkg_desc,
   div_id,
   div_desc,
   region_id,
   region_desc
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/food_table.tsv' INTO TABLE food
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
   ndb_no,
   food_desc,
   water_g,
   energ_kcal,
   protein_g,
   lipid_tot_g,
   ash_g,
   carbohydrt_g,
   fiber_td_g,
   sugar_tot_g,
   calcium_mg,
   iron_mg,
   magnesium_mg,
   phosphorus_mg,
   potassium_mg,
   sodium_mg,
   zinc_mg,
   copper_mg,
   manganese_mg,
   selenium_g,
```

```
  vit_c_mg,
  thiamin_mg,
  riboflavin_mg,
  niacin_mg,
  panto_acid_mg,
  vit_b6_mg,
  folate_tot_g,
  folic_acid_g,
  food_folate_g,
  folate_dfe_g,
  choline_tot_mg,
  vit_b12_g,
  vit_a_iu,
  vit_a_rae,
  retinol_g,
  alpha_carot_g,
  beta_carot_g,
  beta_crypt_g,
  lycopene_g,
  lutzea_g,
  vit_e_mg,
  vit_d_g,
  vit_d_iu,
  vit_k_g,
  fa_sat_g,
  fa_mono_g,
  fa_poly_g,
  cholestrl_mg
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/unit_conversion_table.tsv' INTO TABLE
unit_conversion
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
  ndb_no,
  value_a,
  name_a,
  value_b,
  name_b
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/food_food_group_table.tsv' INTO TABLE
food_food_group
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
```

```
IGNORE 1 LINES
(
   ndb_no,
   fg_id
);

LOAD DATA LOCAL INFILE '/tmp/db_fwt_dataset_tsv/food_subgroup_group_table.tsv' INTO
TABLE food_subgroup_group
FIELDS TERMINATED BY '\t'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
   fg_id,
   fsg_id
);
```

9. Database, directory structure, prototype

```
.
├── dataset_tsv
│   ├── desktop.ini
│   ├── food_food_group_table.tsv
│   ├── food_group_table.tsv
│   ├── food_subgroup_group_table.tsv
│   ├── food_subgroup_table.tsv
│   ├── food_table.tsv
│   ├── location_table.tsv
│   ├── prices
│   │   ├── desktop.ini
│   │   ├── prices_fatsandpreparedfoods_38.tsv
│   │   ├── prices_fatsandpreparedfoods_39.tsv
│   │   ├── prices_fatsandpreparedfoods_40.tsv
│   │   ├── prices_fatsandpreparedfoods_41.tsv
│   │   ├── prices_fatsandpreparedfoods_42.tsv
│   │   ├── prices_fatsandpreparedfoods_43.tsv
│   │   ├── prices_fatsandpreparedfoods_44.tsv
│   │   ├── prices_fatsandpreparedfoods_45.tsv
│   │   ├── prices_fatsandpreparedfoods_46.tsv
│   │   ├── prices_fatsandpreparedfoods_47.tsv
│   │   ├── prices_fatsandpreparedfoods_48.tsv
│   │   ├── prices_fatsandpreparedfoods_49.tsv
│   │   ├── prices_fatsandpreparedfoods_50.tsv
│   │   ├── prices_fatsandpreparedfoods_51.tsv
│   │   ├── prices_fatsandpreparedfoods_52.tsv
│   │   ├── prices_fruitsandvegetables_10.tsv
│   │   ├── prices_fruitsandvegetables_11.tsv
```

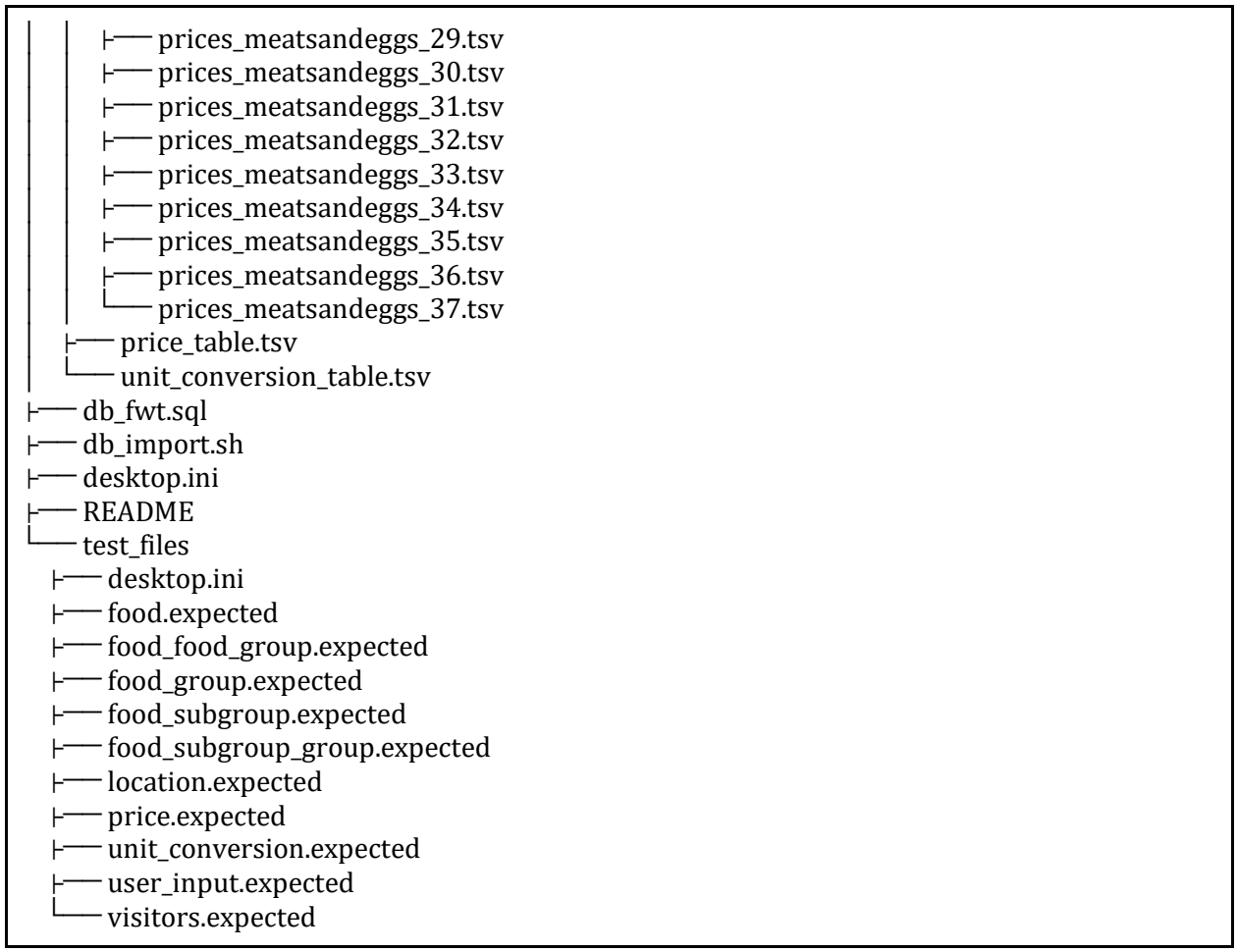
```
│   │   ├── prices_fruitsandvegetables_12.tsv
│   │   ├── prices_fruitsandvegetables_13.tsv
│   │   ├── prices_fruitsandvegetables_14.tsv
│   │   ├── prices_fruitsandvegetables_15.tsv
│   │   ├── prices_fruitsandvegetables_1.tsv
│   │   ├── prices_fruitsandvegetables_2.tsv
│   │   ├── prices_fruitsandvegetables_3.tsv
│   │   ├── prices_fruitsandvegetables_4.tsv
│   │   ├── prices_fruitsandvegetables_5.tsv
│   │   ├── prices_fruitsandvegetables_6.tsv
│   │   ├── prices_fruitsandvegetables_7.tsv
│   │   ├── prices_fruitsandvegetables_8.tsv
│   │   ├── prices_fruitsandvegetables_9.tsv
│   │   ├── prices_grainsanddairy_16.tsv
│   │   ├── prices_grainsanddairy_17.tsv
│   │   ├── prices_grainsanddairy_18.tsv
│   │   ├── prices_grainsanddairy_19.tsv
│   │   ├── prices_grainsanddairy_20.tsv
│   │   ├── prices_grainsanddairy_21.tsv
│   │   ├── prices_grainsanddairy_22.tsv
│   │   ├── prices_grainsanddairy_23.tsv
│   │   ├── prices_grainsanddairy_24.tsv
│   │   ├── prices_grainsanddairy_25.tsv
│   │   ├── prices_grainsanddairy_26.tsv
│   │   ├── prices_grainsanddairy_27.tsv
│   │   ├── prices_meatsandeggs_28.tsv
│   │   ├── prices_meatsandeggs_29.tsv
│   │   ├── prices_meatsandeggs_30.tsv
│   │   ├── prices_meatsandeggs_31.tsv
│   │   ├── prices_meatsandeggs_32.tsv
│   │   ├── prices_meatsandeggs_33.tsv
│   │   ├── prices_meatsandeggs_34.tsv
│   │   ├── prices_meatsandeggs_35.tsv
│   │   ├── prices_meatsandeggs_36.tsv
│   │   └── prices_meatsandeggs_37.tsv
│   ├── price_table.tsv
│   └── unit_conversion_table.tsv
├── db_fwt.sql
├── db_import.sh
├── desktop.ini
├── README
├── test_files
│   ├── desktop.ini
│   ├── food.expected
│   ├── food_food_group.expected
│   ├── food_group.expected
│   ├── food_subgroup.expected
│   ├── food_subgroup_group.expected
│   ├── location.expected
```

```
│   ├── price.expected
│   ├── unit_conversion.expected
│   ├── user_input.expected
│   └── visitors.expected
└── usda_spreadsheets
    ├── desktop.ini
    ├── nutrition
    │   ├── ABBREV.xlsx
    │   ├── desktop.ini
    │   ├── food_food_group_table.gsheet
    │   ├── food_group_table.gsheet
    │   ├── foods_table.gsheet
    │   ├── nutrition_foodDescriptions.gsheet
    │   ├── nutrition_foodDescriptions.ods
    │   ├── nutrition_foodSubGroups.ods
    │   └── unit_conversion_table.gsheet
    └── prices
        ├── desktop.ini
        ├── fatsandpreparedfoods_q1.xls
        ├── food_subgroup_group_table (1).gsheet
        ├── food_subgroup_table.gsheet
        ├── fruitsandvegetables_q1.xls
        ├── grainsanddairy_q1.xls
        ├── location_table.gsheet
        ├── meatsandeggs_q1.xls
        ├── prices_backup
        │   ├── Copy of prices_fatsandpreparedfoods.gsheet
        │   ├── Copy of prices_fruitsandvegetables.gsheet
        │   ├── Copy of prices_grainsanddairy.gsheet
        │   ├── Copy of prices_meatsandeggs.gsheet
        │   └── desktop.ini
        ├── prices_fatsandpreparedfoods.gsheet
        ├── prices_fruitsandvegetables.gsheet
        ├── prices_grainsanddairy.gsheet
        ├── prices_meatsandeggs.gsheet
        ├── qfahpd1codebook.xls
        ├── qfahpd1codebook.xls.gsheet
        └── sr28_doc.pdf
```

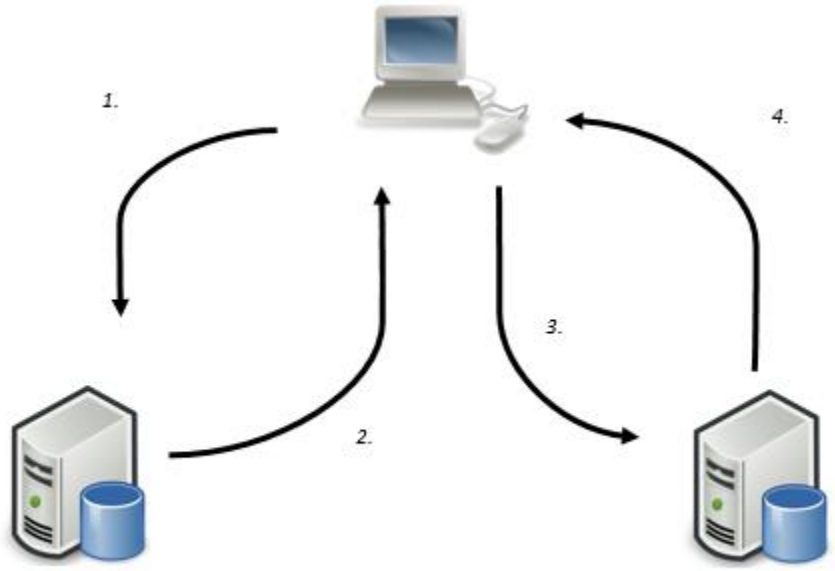
10. Database, directory structure, after initial refinement

```
.
├── dataset_tsv
│   ├── desktop.ini
│   ├── food_food_group_table.tsv
│   ├── food_group_table.tsv
```

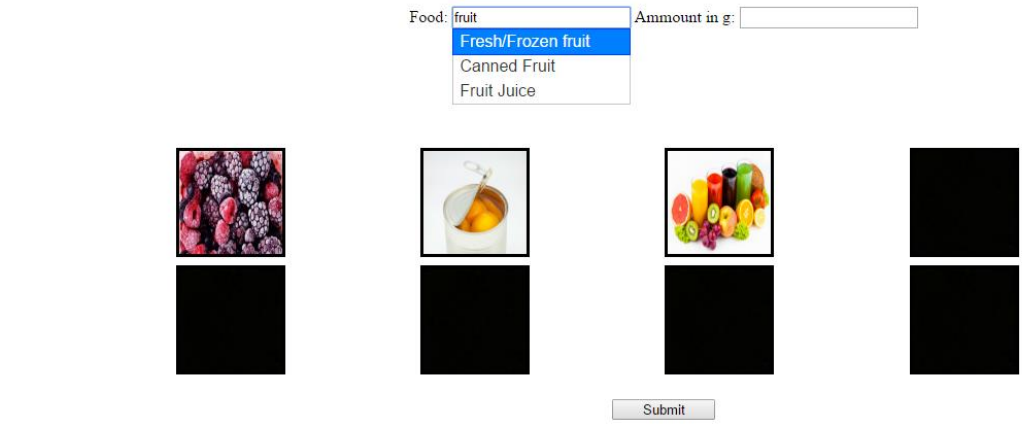
```
├── food_subgroup_group_table.tsv
├── food_subgroup_table.tsv
├── food_table.tsv
├── location_table.tsv
├── prices
│   ├── desktop.ini
│   ├── prices_fatsandpreparedfoods_38.tsv
│   ├── prices_fatsandpreparedfoods_39.tsv
│   ├── prices_fatsandpreparedfoods_40.tsv
│   ├── prices_fatsandpreparedfoods_41.tsv
│   ├── prices_fatsandpreparedfoods_42.tsv
│   ├── prices_fatsandpreparedfoods_43.tsv
│   ├── prices_fatsandpreparedfoods_44.tsv
│   ├── prices_fatsandpreparedfoods_45.tsv
│   ├── prices_fatsandpreparedfoods_46.tsv
│   ├── prices_fatsandpreparedfoods_47.tsv
│   ├── prices_fatsandpreparedfoods_48.tsv
│   ├── prices_fatsandpreparedfoods_49.tsv
│   ├── prices_fatsandpreparedfoods_50.tsv
│   ├── prices_fatsandpreparedfoods_51.tsv
│   ├── prices_fatsandpreparedfoods_52.tsv
│   ├── prices_fruitsandvegetables_10.tsv
│   ├── prices_fruitsandvegetables_11.tsv
│   ├── prices_fruitsandvegetables_12.tsv
│   ├── prices_fruitsandvegetables_13.tsv
│   ├── prices_fruitsandvegetables_14.tsv
│   ├── prices_fruitsandvegetables_15.tsv
│   ├── prices_fruitsandvegetables_1.tsv
│   ├── prices_fruitsandvegetables_2.tsv
│   ├── prices_fruitsandvegetables_3.tsv
│   ├── prices_fruitsandvegetables_4.tsv
│   ├── prices_fruitsandvegetables_5.tsv
│   ├── prices_fruitsandvegetables_6.tsv
│   ├── prices_fruitsandvegetables_7.tsv
│   ├── prices_fruitsandvegetables_8.tsv
│   ├── prices_fruitsandvegetables_9.tsv
│   ├── prices_grainsanddairy_16.tsv
│   ├── prices_grainsanddairy_17.tsv
│   ├── prices_grainsanddairy_18.tsv
│   ├── prices_grainsanddairy_19.tsv
│   ├── prices_grainsanddairy_20.tsv
│   ├── prices_grainsanddairy_21.tsv
│   ├── prices_grainsanddairy_22.tsv
│   ├── prices_grainsanddairy_23.tsv
│   ├── prices_grainsanddairy_24.tsv
│   ├── prices_grainsanddairy_25.tsv
│   ├── prices_grainsanddairy_26.tsv
│   ├── prices_grainsanddairy_27.tsv
│   ├── prices_meatsandeggs_28.tsv
```

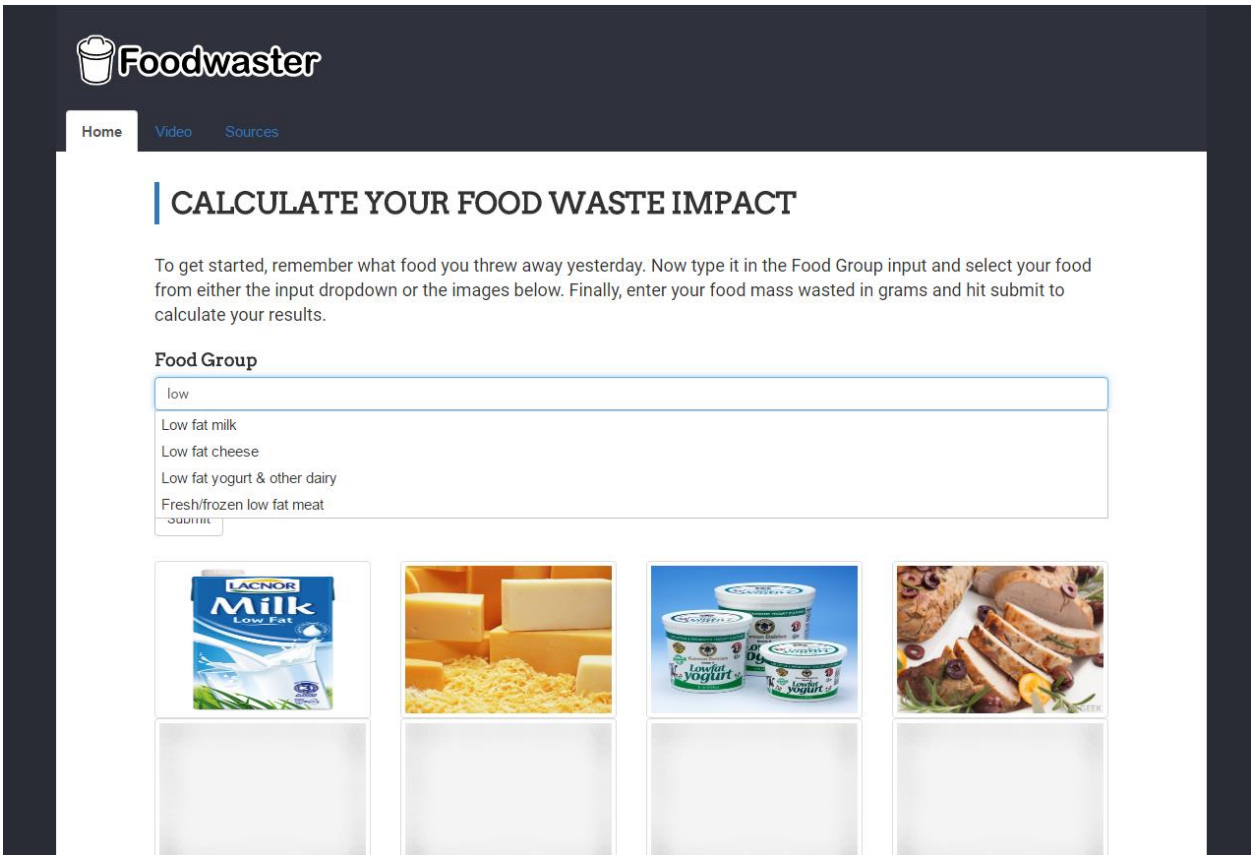

│ │ ├── prices_meatsandeggs_29.tsv
│ │ ├── prices_meatsandeggs_30.tsv
│ │ ├── prices_meatsandeggs_31.tsv
│ │ ├── prices_meatsandeggs_32.tsv
│ │ ├── prices_meatsandeggs_33.tsv
│ │ ├── prices_meatsandeggs_34.tsv
│ │ ├── prices_meatsandeggs_35.tsv
│ │ ├── prices_meatsandeggs_36.tsv
│ │ └── prices_meatsandeggs_37.tsv
│ ├── price_table.tsv
│ └── unit_conversion_table.tsv
├── db_fwt.sql
├── db_import.sh
├── desktop.ini
├── README
└── test_files
├── desktop.ini
├── food.expected
├── food_food_group.expected
├── food_group.expected
├── food_subgroup.expected
├── food_subgroup_group.expected
├── location.expected
├── price.expected
├── unit_conversion.expected
├── user_input.expected
└── visitors.expected

11. Initial end-to-end prototype

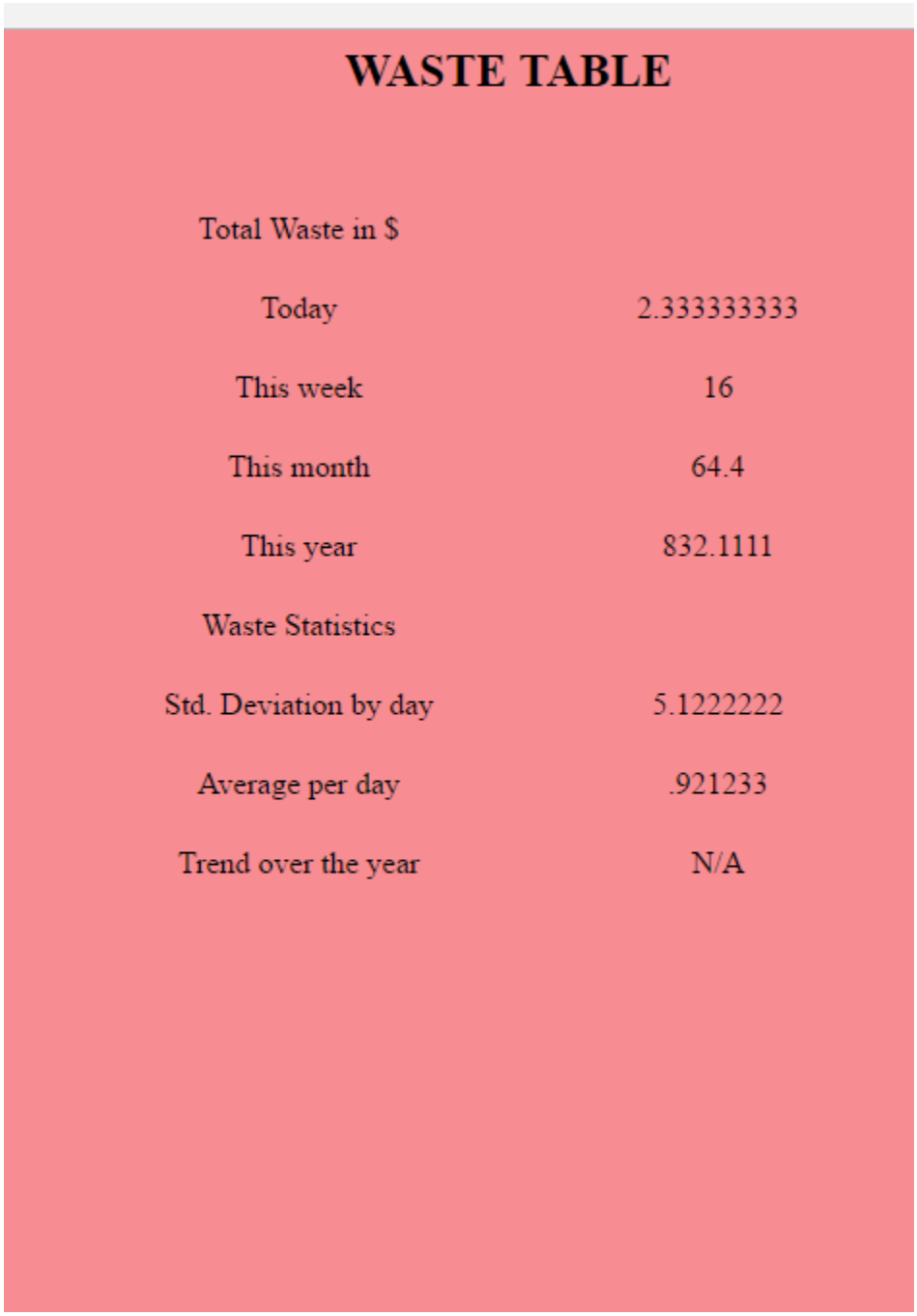


12. A prototype of the front page - before and after

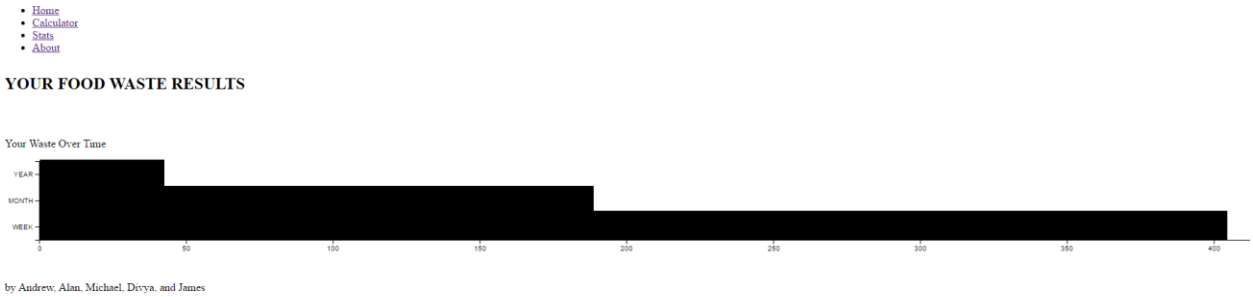




13. Initial frontend statistics page

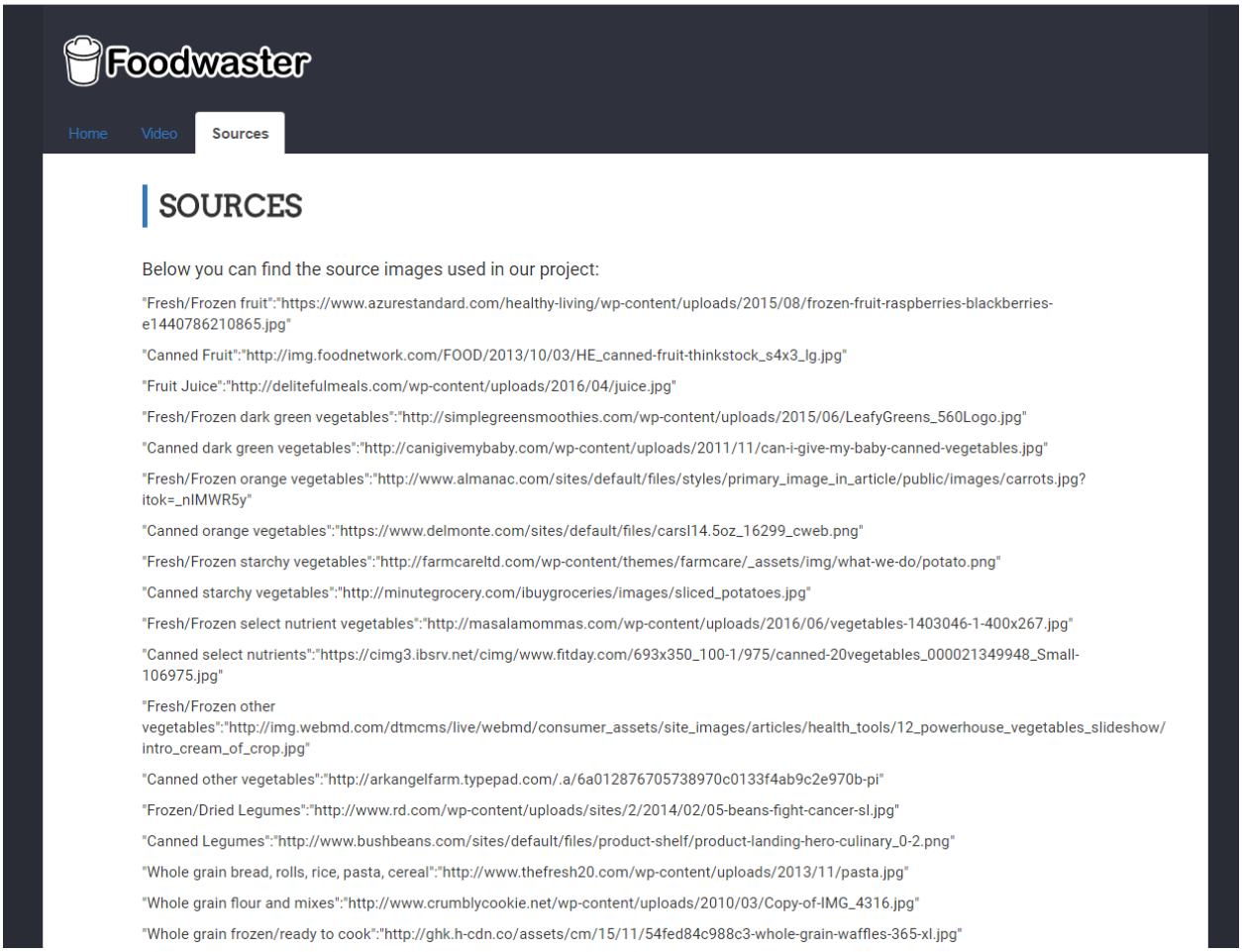


14. Second prototype front end statistics page

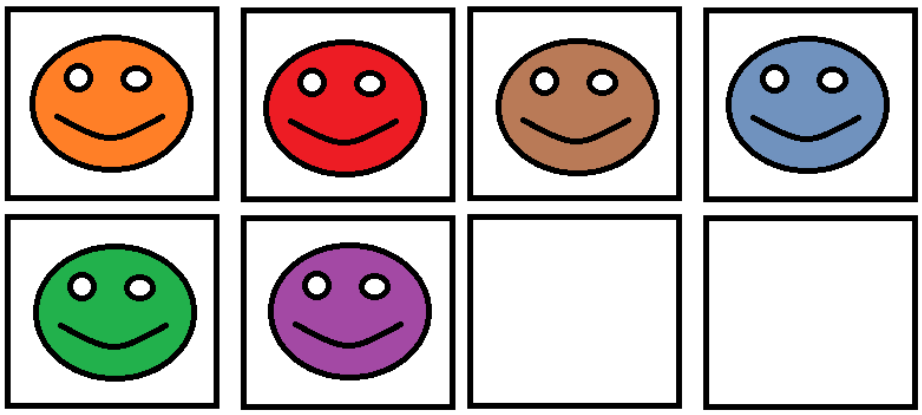


15. Refinement of the statistics page

16. Refinement of the general template and sources page



17. Grid Structure for front page displaying images.



18. Inventory of web server files

Tables

1. Deadlines and milestones

.... - Task completed
.... - Task not completed

Dates	Objectives	Lead
1/31	Milestones Drafted	Alan
2/2	Meet with Client; Finalize requirements and design	Client
2/9	Wireframes and Database Design	Michael + James
2/16	Prototype of website without interaction with DB	Michael + Andrew
2/17	Drafting Video	Alan
2/17	Wire Frames and DB Diagrams	Andrew + James
2/18	Review progress with Client	
2/19	Requirements and Design Report Deadline	
2/23	Meet with client	Client
2/24	Film a First Draft of the Video	Alan
2/25	Review progress with Client	Client
2/28	Front end draft	Michael
3/4	Implementation Report deadline	
3/6	Review progress of front end with Client	Michael
3/15	Video(s) finished and included	Alan
3/19	Review progress with Client	Client
3/23	Implementation: Summary Statistics	Divya + James
3/24	Testing	Andrew + Divya
3/26	Review Progress with Client	Divya + James

3/30	Prototype Refinement Report deadline	
4/1	UX Testing	Alan + Client
4/5	Mobile Compatibility	Andrew
4/7	Backend Testing	James
4/9	Review Product with Client	Client
4/13	Documentation as Needed	Divya
4/16	Testing Report Deadline	
4/19	Final Review with Client	
4/28	Final Report Deadline	

2. Database functional dependencies

Entities And/Or Relations	Functional Dependencies in the ER diagram	Minimal Functional Dependencies Cover
food_subgroup	$SubFdGrp\_ID \rightarrow SubFdGrp\_Desc$	$SubFdGrp\_ID \rightarrow SubFdGrp\_Desc$
determines	$SubFdGrp\_ID, Mkg\_ID, Year, Quarter \rightarrow UnitPrice\_Avg, Standard\ Error$	$SubFdGrp\_ID, Mkg\_ID, Year, Quarter \rightarrow UnitPrice\_Avg, Standard\ Error$
location	$Mkg\_ID \rightarrow Mkg\_Desc, Div\_ID, Div\_Desc, Region\_ID, Region\_Desc$	$Mkg\_ID \rightarrow Mkg\_Desc, Div\_ID, Div\_Desc, Region\_ID, Region\_Desc$
foods	$NDB.no \rightarrow Shrt\_Desc, \{46\ nutrients\}$	$NDB.no \rightarrow Shrt\_Desc, \{46\ nutrients\}, FromUnitValue, FromUnitName,$ $ToUnitValue, ToUnitName, FdGrp\_ID, FdGrp\_Desc$
unit_conversions for	$NDB.no \rightarrow FromUnitValue, FromUnitName, ToUnitValue, ToUnitName$	
under	$NDB.no \rightarrow FdGrp\_ID$	
food_group	$FdGrp\_ID \rightarrow FdGrp\_Desc$	
contained_in	$SubFdGrp\_ID \rightarrow FdGrp\_ID$	$SubFdGrp\_ID \rightarrow FdGrp\_ID$
user_input chooses	$User\_ID, Date, Time \rightarrow NDB.no, SubFdGrp\_ID, Mkg\_ID,$ $Year, Quarter, Weight$	$User\_ID, Date, Time \rightarrow NDB.no, SubFdGrp\_ID, Mkg\_ID,$ $Year, Quarter, Weight$

3. Database schema

Table Name	**Column Name**	**Data Type**	**Primary Key**	**Description**
food_group	fg_id fg_desc	INT VARCHAR	fg_id	Contains all food groups by id and name.
food_subgroup	fsg_id fsg_desc	INT VARCHAR	fsg_id	Contains all subgroups of food groups by id and name.

price	fsg_id mkg_id price_year price_quarter unit_price_100g standard_error	INT INT YEAR INT DOUBLE DOUBLE	fsg_id mkg_id price_year price_quarter	Contains the unique unit price (dollars/100g) of a given food subgroup found at a given location at a given time.
location	mkg_id mkg_desc div_id div_desc region_id region_desc	INT VARCHAR INT VARCHAR INT VARCHAR	mkg_id	Contains all the locations where the prices were surveyed.
food	ndb_no {46 nutrients}	INT {46 DOUBLEs}	ndb_no	Contains all foods surveyed for their nutritional data.
unit_conversion	ndb_no from_name from_value to_name to_value	INT VARCHAR DOUBLE VARCHAR DOUBLE	ndb_no	Contains what unit conversions were available from the dataset.
food_food_group	ndb_no fg_id	INT INT	ndb_no	Relates foods to their food groups through id pairs.
food_subgroup_group	fsg_id fg_id	INT INT	fsg_id	Relates food groups to food subgroups through id pairs.
user_input	uid date time ndb_no fsg_id mkg_id price_year price_quarter weight_g	INT DATE TIME INT INT INT YEAR INT DOUBLE	uid date time	Can be used to store a query made by a particular user of UID at a particular DATE and TIME.
visitors	identifier uid	VARCHAR INT	identifier	Can be used to track visitors, possibly for persistent statistics over time for the user.