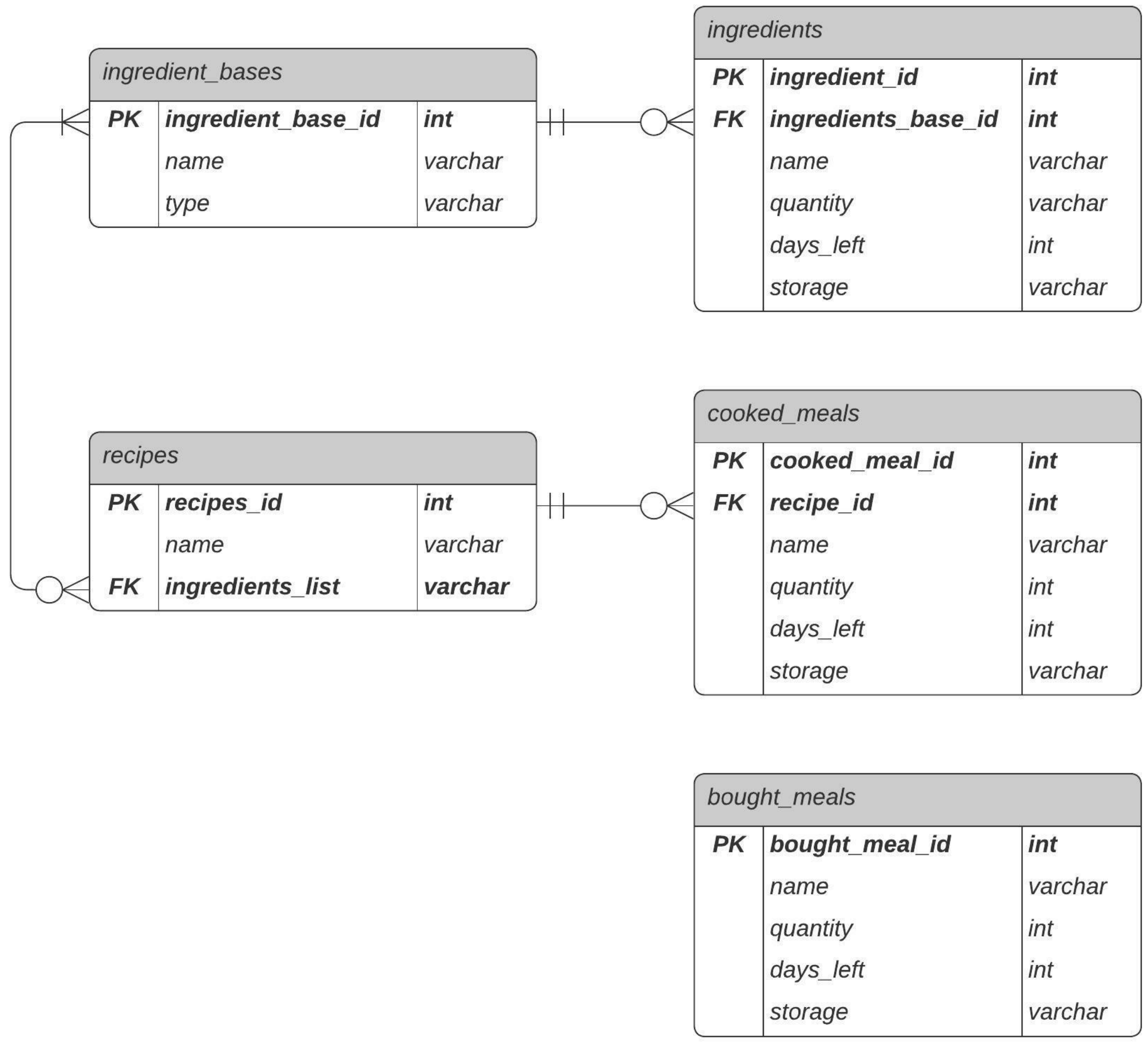
Criterion B - Design

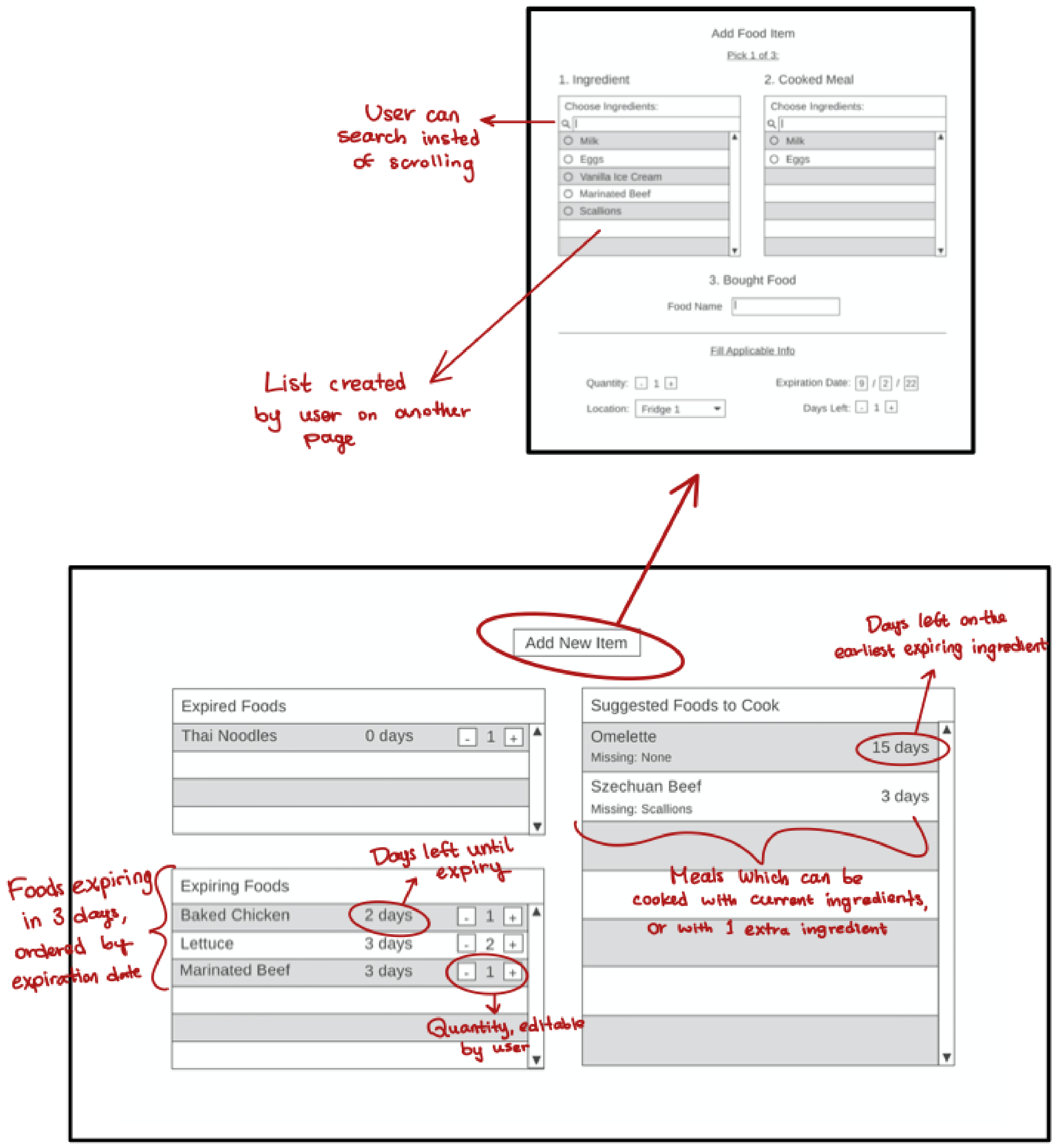
Entity Relationship Diagram



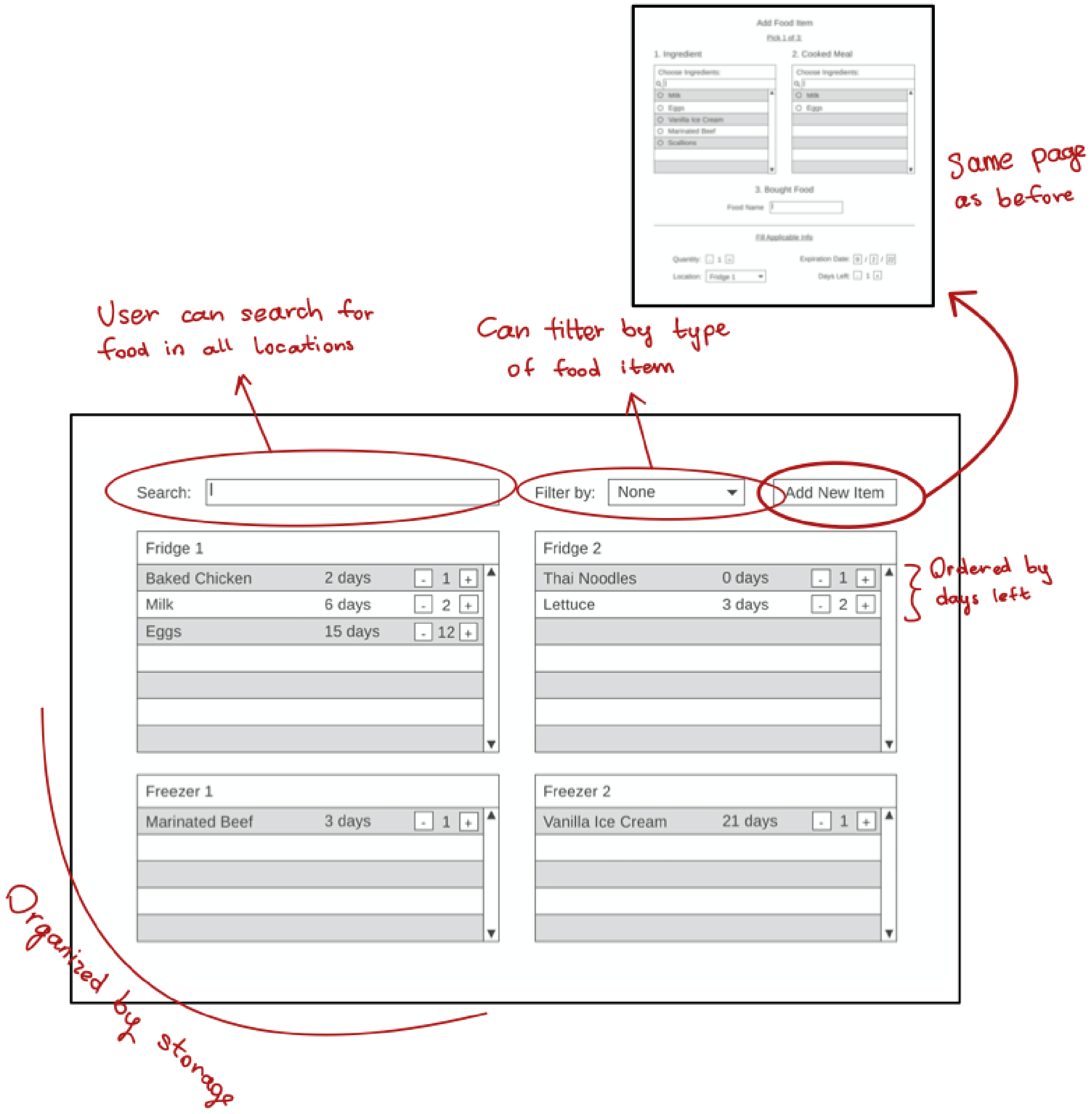
KEY:

- Circle and three prongs: zero to many
- Vertical line and three prongs: one to many
- Two vertical lines: one and only one

GUI Prototype



Add Food Item
Pick 1 of 3:
1. Ingredient
2. Cooked Meal
Choose Ingredients:
Milk
Eggs
Vanilla Ice Cream
Marinated Beef
Scallions
Choose Ingredients:
Milk
Eggs
3. Bought Food
Food Name
Fill Applicable Info
Quantity: - 1 +
Expiration Date: 9 / 2 / 22
Location: Fridge 1
Days Left: - 1 +
User can search insted of scrolling
List created by user on another page
Add New Item
Days left on the earliest expiring ingredient
Expired Foods
Thai Noodles 0 days - 1 +
Suggested Foods to Cook
Omelette Missing: None 15 days
Szechuan Beef Missing: Scallions 3 days
Meals which can be cooked with current ingredients, or with 1 extra ingredient
Days left until expiry
Expiring Foods
Baked Chicken 2 days - 1 +
Lettuce 3 days - 2 +
Marinated Beef 3 days - 1 +
Foods expiring in 3 days, ordered by expiration date
Quantity, editable by user


Same page as before
User can search for food in all locations
Can filter by type of food item
Ordered by days left
Organized by storage

Add Food Item

Pick 1 of 3:

1. Ingredient

Choose Ingredients:

- Milk
- Eggs
- Vanilla Ice Cream
- Marinated Beef
- Scallions

2. Cooked Meal

Choose Ingredients:

- Milk
- Eggs

3. Bought Food

Food Name: |

Fill Applicable Info

Quantity: - 1 +

Location: Fridge 1

Expiration Date: 9 / 2 / 22

Days Left: - 1 +

Search: |

Filter by: None

Add New Item

Fridge 1

Baked Chicken	2 days	- 1 +
Milk	6 days	- 2 +
Eggs	15 days	- 12 +

Fridge 2

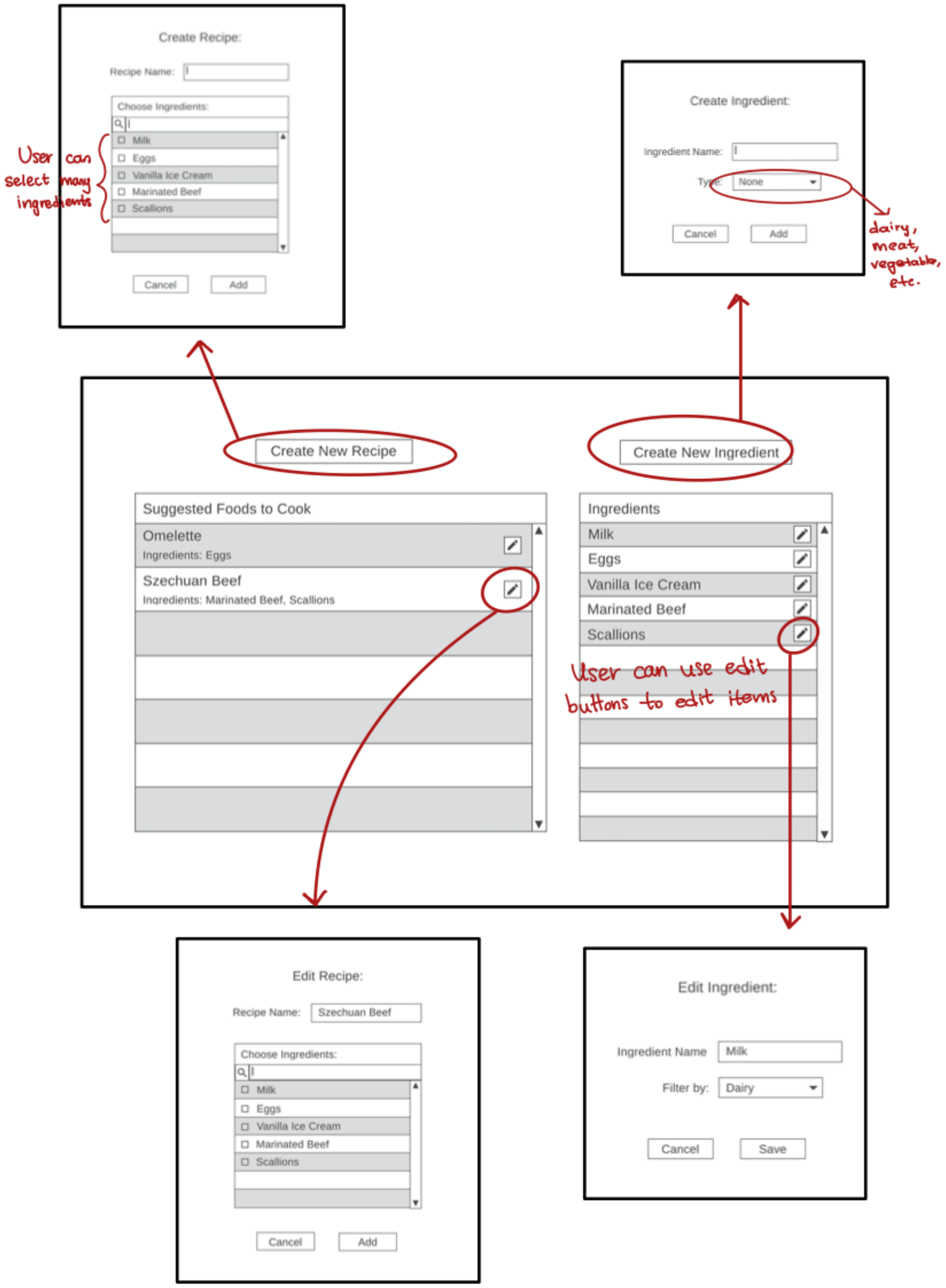
Thai Noodles	0 days	- 1 +
Lettuce	3 days	- 2 +

Freezer 1

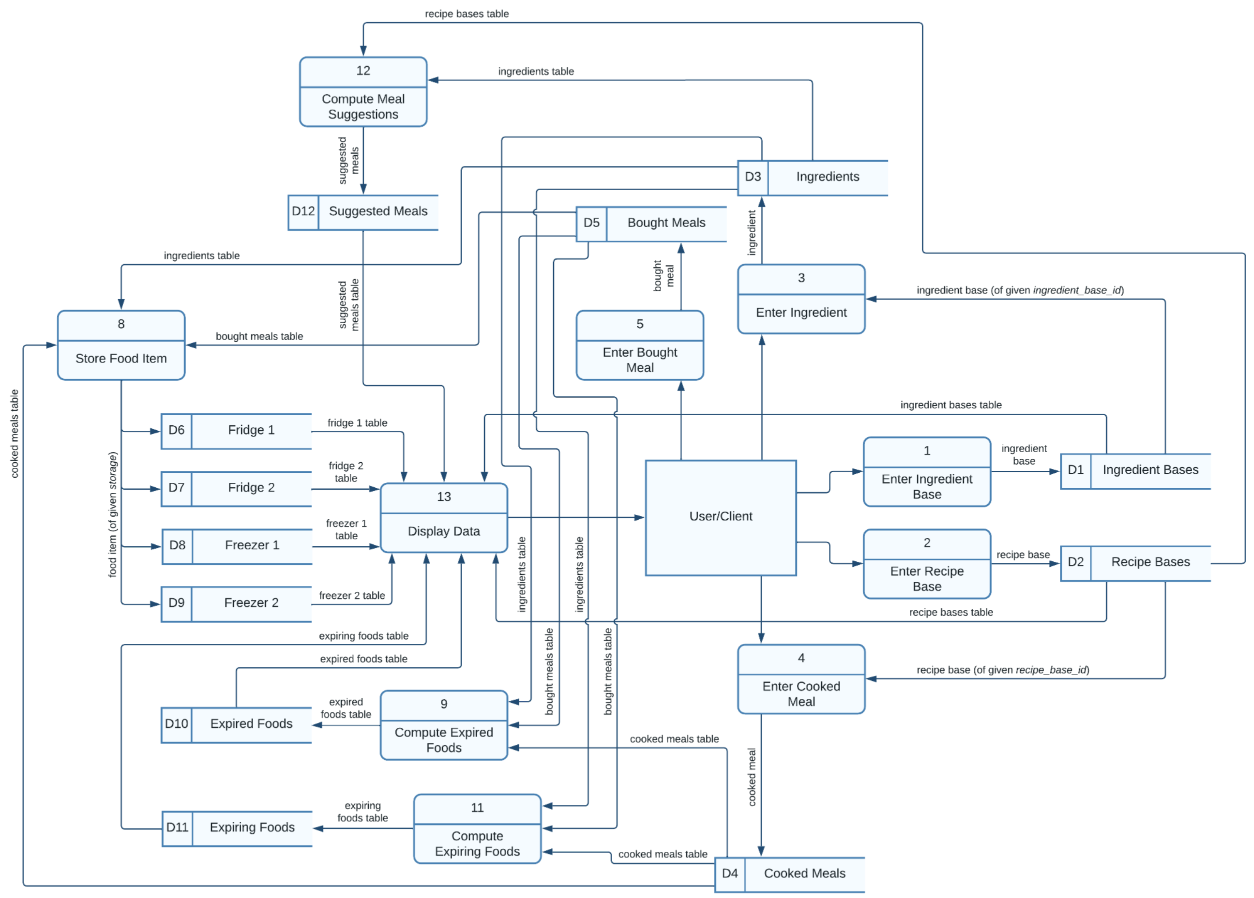
Marinated Beef	3 days	- 1 +

Freezer 2

Vanilla Ice Cream	21 days	- 1 +


Create Recipe:
Recipe Name:
Choose Ingredients:
Milk
Eggs
Vanilla Ice Cream
Marinated Beef
Scallions
Cancel
Add
User can select many ingredients
Create Ingredient:
Ingredient Name:
Type:
None
Cancel
Add
dairy, meat, vegetable, etc.
Create New Recipe
Create New Ingredient
Suggested Foods to Cook
Omelette
Ingredients: Eggs
Szechuan Beef
Ingredients: Marinated Beef, Scallions
Ingredients
Milk
Eggs
Vanilla Ice Cream
Marinated Beef
Scallions
User can use edit buttons to edit items
Edit Recipe:
Recipe Name:
Szechuan Beef
Choose Ingredients:
Milk
Eggs
Vanilla Ice Cream
Marinated Beef
Scallions
Cancel
Add
Edit Ingredient:
Ingredient Name
Milk
Filter by:
Dairy
Cancel
Save

Data Flow Diagram



*Please zoom in to see the detail

Testing Plan

Action to Test	Method of Testing & Expected Results
The user can add foods to the fridge, specifying name, quantity, expiration date, and location.	Add a new food, assigning values to name, quantity, expiration date, and location. Check if values show up correctly and in the correct table.
The user can edit foods, updating any of the aforementioned parameters.	Change the name, quantity, expiration date, and location of a previously-added item, one by one and/or in combination. Check if values are updated correctly.
The user can remove existing items.	Remove a previously-added item and check if it is successfully taken off all lists.
The program presents a list of all food items in fridge 1, fridge 2, freezer 1, and freezer 2, respectively.	On the storage page, check whether the name, quantity, and days left in the four tables are correct. Then cross-check this data with *fridge_1*, *fridge_2*, *freezer_1*, and *freezer_2* in MySQL, and cross-check those four tables with *ingredients*, *cooked_meals*, and *bought_meals*.
The user can order tables by a chosen parameter.	On the storage page, order the four tables by name, quantity, and days left, respectively, and check for each whether the data was ordered properly in the GUI. Then, move to the recipes/ingredients page and repeat, ordering the ingredients table by name and type.
The user can filter the data in tables by a chosen parameter.	On the storage page, filter the four tables by each of the filtering parameters (to be decided later, but possibilities include "meat," "dairy," "meal," etc.) and check whether only data matching that parameter is displayed.
The user can search for an item by name.	Search for an item and see if it shows up. If any other items show up too, check that they appear for a valid reason (e.g. searching "milk" may yield both "milk" and "milkshake," and that would be successful)
The program presents a list of food that has expired.	On the home page, check whether the name, quantity, and days left in the expired foods table are correct. Then cross-check this data with *expired_foods* in MySQL, and cross-check *expired_foods* with *ingredients*, *cooked_meals*, and *bought_meals*.
The program presents a list of food that expires within 3 days.	On the home page, check whether the name, quantity, and days left in the expiring foods table are correct. Then cross-check this data with *expiring_foods* in MySQL, and cross-check *expiring_foods* with *ingredients*, *cooked_meals*, and *bought_meals*.
The program presents a list of meal suggestions, where all or all but one	On the home page, check whether the name, ingredients, and days left in the table are correct. Then cross-check this data

ingredient are available.	with *meal_suggestions* in MySQL, and cross-check *meal_suggestions* with *recipes* and *ingredients*.
Data is stored permanently.	Close and reopen the program after making changes. Check to see if all data is still there.