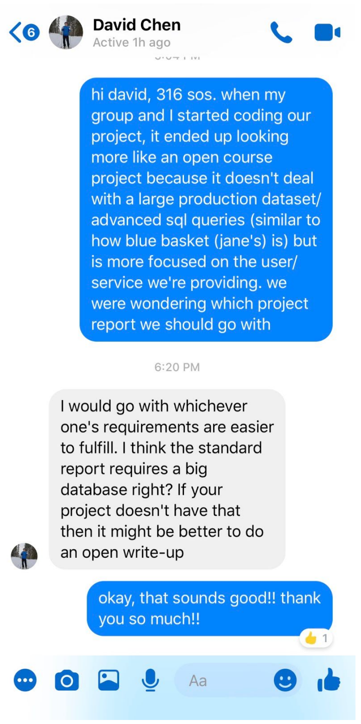
Final Report

Christina Le, Griffin Malm, Rebecca Shu, Jeevan Tewari, Jonathan Yu

Link to Web Application: https://virtualfridge-65ccf.web.app/

NOTE: SWITCH TO OPEN PROJECT
While our project was submitted as a Standard Project for Milestone 1 and Milestone 2, we have decided to submit our Final Report as an Open Project. We made this decision on the basis that our project is not data-heavy: we aimed to create an application that would allow users to keep track of their food items. In this sense, all of our data will be user-generated. Considering that we are outsourcing recipe generation to the Edamam Recipe API (opposed to storing recipes in an internal database), there isn't much data that our application will have to store aside from authentication information, general user information, and the food in each user's fridge. Additionally, as the majority of the coding for this project occurred after the submission of Milestone 2, we were only able to discover this issue during the building of our final report.

We confirmed the decision to switch to an Open Project with our project TA, David. He acknowledged that our project fit the form of an Open Project much better than that of a Standard Project.


David Chen
Active 1h ago

hi david, 316 sos. when my group and I started coding our project, it ended up looking more like an open course project because it doesn't deal with a large production dataset/ advanced sql queries (similar to how blue basket (jane's) is) but is more focused on the user/ service we're providing. we were wondering which project report we should go with

6:20 PM

I would go with whichever one's requirements are easier to fulfill. I think the standard report requires a big database right? If your project doesn't have that then it might be better to do an open write-up

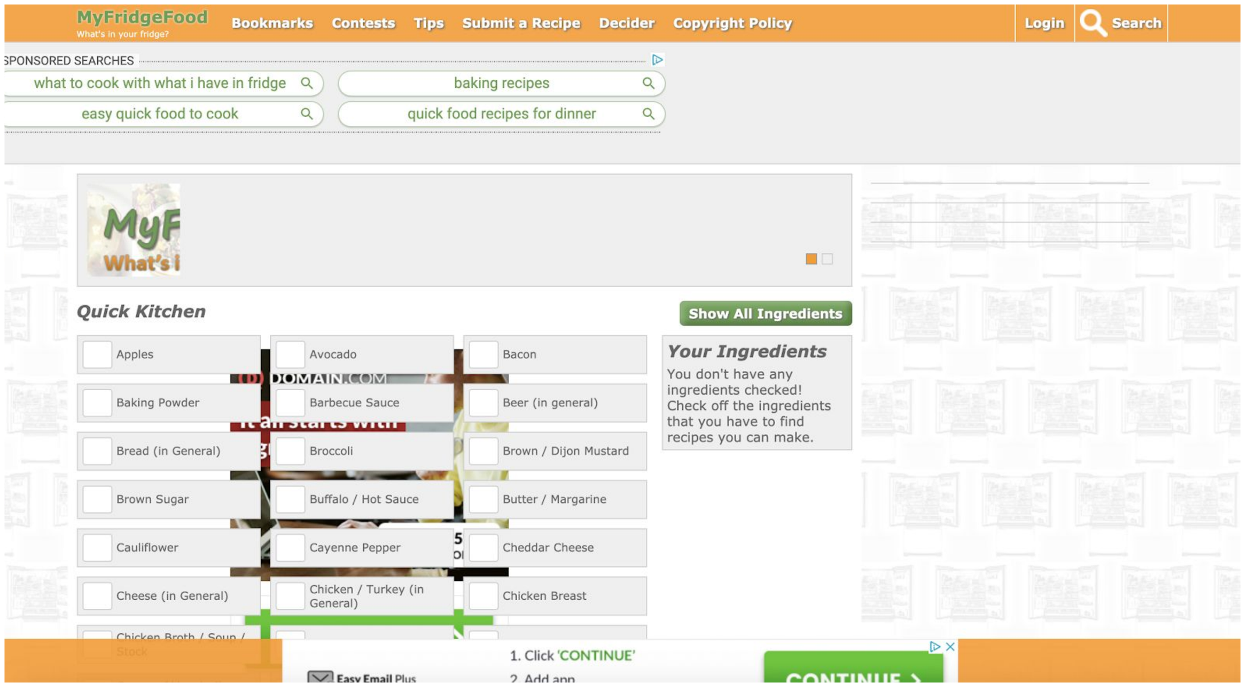
okay, that sounds good!! thank you so much!!

Problem Description and Motivation

According to the USDA, Americans waste about 30-40% of the food supply. In a single year, the average adult throws away 103 pounds of food. The knowledge of this prompted us to further research efforts to alleviate this waste so that we can all do our part to help the environment. The first step is managing what is going on in our own household, specifically in our fridges. WasteNoMore aims to manage users' fridges for them, prompting them to engage and use all of the food in their fridge before it expires.

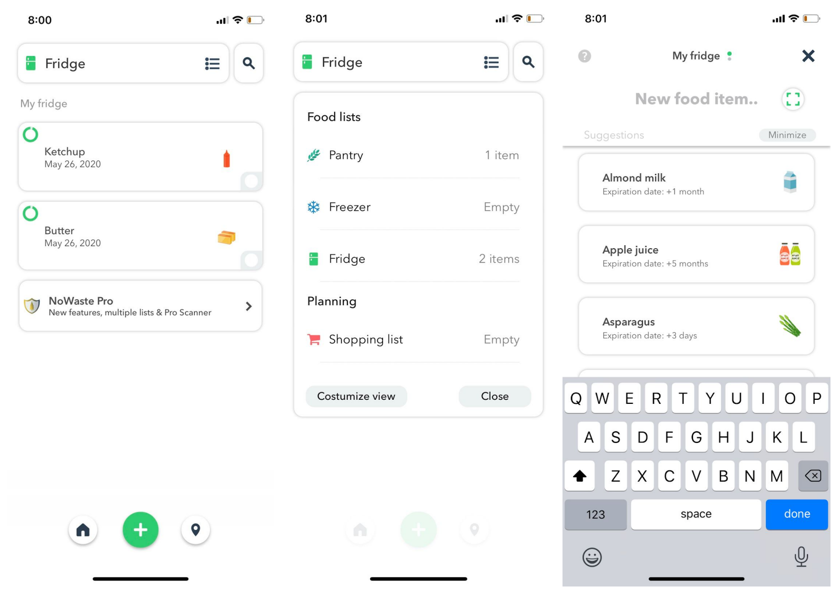
Survey of Related Work

There are a variety of fridge-management applications available to the public on mobile and the web, some of them being Fridge Pal, Best Before, Fresh Box, and MyFridgeFood. Fridge Pal, Best Before, and Fresh Box are all mobile applications, which shows our motivation to create a web application. Out of the four applications listed, MyFridgeFood was the only web application with a running site for fridge management. A quick search for fridge-management applications usually results in redirects to an app store. Below is a screenshot of MyFridgeFood's home page.



You don't have any ingredients checked! Check off the ingredients that you have to find recipes you can make.

NoWaste is the first recommendation on the iTunes App Store for fridge-management. Upon opening the application, a user is able to decide if they want to manage their fridge, pantry, and/or freezer. The free version allows you to select from a large list of popular items and has general terminology, such as "spices" for food items that might not be found on the list. In your own inventory, the user then is able to adjust the expiration date of their items accordingly. One of the most advanced features of this application is the implementation of software to read in food items from barcodes or receipts. An ongoing shopping list is also able to be created within the application. In the premium version, users are able to select various food items and see applicable recipes. Below are screenshots of the inventory on NoWaste.



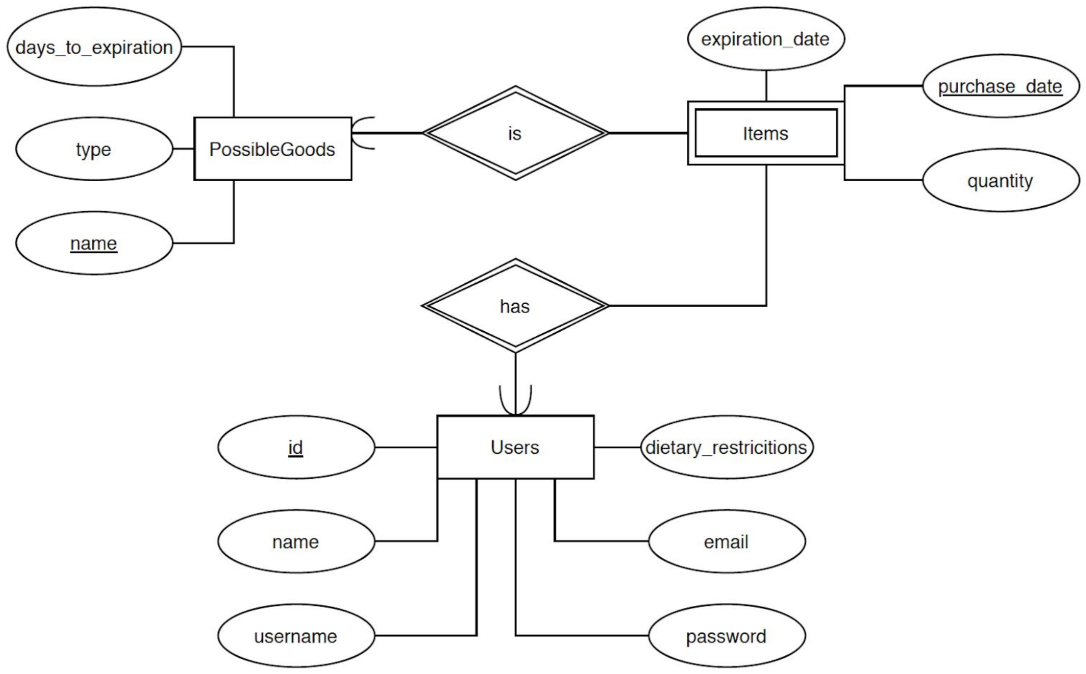
Application Description

Our application will allow users to create a virtual display of their refrigerator. A user can add items to their virtual fridge as they are obtained and delete items from their virtual fridge as they are consumed. The items in a user's virtual fridge will be sorted by their expiration dates by default, with the foods that are going to expire the soonest listed at the top. This allows users to see what food they should prioritize eating (in order to reduce the amount of food they waste).

E/R Diagram/Database Tables

Due to the structure and built-in features of Firestore and Firebase, an E/R diagram was not applicable for this project. Firebase generates unique keys for every user created through the application and has flexibility within its documents (synonymous to tuples) for any number of fields, so there was not a need to establish what fields were primary keys and what fields were not primary keys. Firebase was a very efficient way to manage users that eliminated the need to implement our original E/R diagram, which is shown below.

Our Original ER Diagram (not currently applicable)



Similarly, the flexibility within Firebase to define any number of fields for each collection (relation) eliminated the need to define database tables before creating the web application. However, an explanation for the collections in Firebase is provided below.

The Firebase API for authentication was used to store users. This API stores user tuples with emails and passwords. In order for this API to interact with the Firestore database, a users table was created in Firestore that solely contains the user's email. So every time a user is created, their email and password are stored in Firebase and a ‘user’ tuple is created in Firestore to log their email. Navigating these two stores was challenging. The foods relation/collection contains tuples that are generated each time a user adds a food to their fridge. These tuples log the user’s email (attribute: userId), the food’s name, the expiration date, and entry date of the food. Performing a search through the foods relation to find which food’s userId matches the current user’s email produces only the food that belongs to each user a.k.a. their fridge.

Description of Web Interface

The user signs up for an account on the website by clicking a button, where they will need a valid email and username, as well as a password. During the sign up process the user can also identify their dietary restrictions from a checklist menu. After the creation of an account, the user can then login to access their information using either their username or email, and their password. After logging in, the user is shown the items in their fridge and has the option of adding and deleting items in their refrigerator based on the name of the item and the expiration date, which can be manually adjusted. After additions of items along with their expiration date, the items in a user’s fridge are sorted by expiration date, with the items that expire sooner to be listed at the top.

Design Choices
We chose to use React, specifically, the create-react-app system for easy functionality purposes. This is because some group members had previous experience working in React, so we found that this method would be the quickest and most efficient way to work on the frontend. Additionally, create-react-app comes preinstalled with bootstrap template functions, as well as other needed files, so that we could quickly get started on this project without having to waste time downloading every essential file individually. While this is not as well suited for heavy, high-traffic websites as React.js Boilerplate, we felt that for this small-scale, starter project, it would be appropriate to use a platform that was easy to set up and work with, as well as perform the functions we needed.

For the backend, we chose to use Google's Firebase. This framework allows for straightforward user authentication and provides Firestore as a cloud database that implements NoSQL. This database stores all of our data, including authentication information, general user information, and the food in each user's fridge. We chose Firebase as it was quite simple to learn and use, with API calls easily implemented into our React.js code. Furthermore, by using Firebase, we avoided the issue of SQL injection attacks, as Firebase is not vulnerable to SQL injection attacks (this is because our queries are executed using an API provided by Firebase, which performs its own input sanitation).

Our motivation for choosing Firebase and its NoSQL framework over PostgreSQL was that Firebase provided convenient deployment of both the database tables, through Firestore, and the entire web application.

New Approaches and Algorithms
While the recipe suggestion functionality is not yet implemented (due to time constraints), significant progress has been made in research of how to formulate GET requests. In addition, an account has been registered with the API (id: 828de6c9; key: 2050bfcf297c354afff9c1c55f66ee6f). Since we signed up using a student account, our team has free access to all of the recipes provided by the Edamam Recipe API.

All GET requests to the API have the following base path: `https://api.edamam.com/search?` and must be supplemented by the q parameter (used for querying recipes based on the expiring food in a user's fridge), the `app_id` parameter (set to the id value for our team's API account), and the `app_key` parameter (set to the key value for our team's API account). Each GET request can be supplemented by various optional parameters, including `ingr` (indicating the maximum number of ingredients in the recipe), `health` (used for enforcing dietary restrictions), and `cuisineType` (used for selecting recipes of a user's preferred cuisine).

An example GET request to the Edamam Recipe API is as follows (this request generates recipes for a user with a peanut allergy that has carrots, potatoes, and onions expiring soon) i:
`https://api.edamam.com/search?q=carrot%20potato%20onion&app_id=828de6c9&app_key=2050bfcf297c354afff9c1c55f66ee6f&health=peanut-free`.

Evaluation of Our System

In comparing WasteNoMore to MyFridgeFood, we evaluated the two websites on functionality, accessibility, and the user experience. Both websites decided to use green and orange as their primary colors, a choice made on the healthy associations with green food and the inviting aspect of orange. When comparing the interfaces, MyFridgeFood has a cluttered design that is overwhelming to the user especially because every functionality is on display on the landing page. The design could be modernized to be more inviting to the user, similar to how NoWaste's interface is set up. WasteNoMore has a very simplistic design that relies on the use of whitespace to draw attention. The design, however, is not uniform across all pages, which can be confusing for the user. For accessibility, WasteNoMore's and NoWaste's onboarding experiences with registration are a lot more conducive of user retention/conversion than MyFridgeFood.

When it comes to functionality, MyFridgeFood has more functionality to its web application because it is able to generate recipes based on a selection of ingredients, a function that WasteNoMore sees as a future extension. However, WasteNoMore allows users to enter any food item instead of providing a limited list of food items, which means that WasteNoMore is more specified to the user. WasteNoMore also shows an expiration date of the items in the fridge. NoWaste's icons for food are a good decision for making the application interactive, something WasteNoMore should implement in the future.

At the end of the day, accessibility is the most important feature in determining which application is going to attract more users because most fridge-applications are a variation of each other. The point of development, then, is to design an application that is tailored towards users and their preferences, which WasteNoMore and MyFridgeFood ould use further research/work on. Offering a mobile version of WasteNoMore could also be of better service to the users who want to scan grocery receipts with their phones, like NoWaste offers.

Open Issues and Future Work

Given the time constraints and the lack of ability to meet up and work as a group easily and efficiently, there were many functions that we had originally hoped to implement, but could not get to. We had been pretty ambitious with our decision to use relatively new technology, such as React and Firebase, to create our application. However, given the recent circumstances and transition to remote learning, we lost the ability to meet in-person, and synchronous work became much more difficult. Thus, learning the new software and implementing it became much more difficult and time-consuming.

In our web application, we hoped to include a section in our sign up process, where the user could manually check off their personalized dietary restrictions, which included options such as dairy free, vegetarian, vegan, etc. While this was implemented on the website, its functionality is still in development, and therefore cannot be utilized at the moment. However, this functionality was meant to serve the purpose of restricting the recipes that our website could suggest and output to the user, another function that we unfortunately were not able to implement. Therefore, though there is an option for the user to customize and select their own dietary restrictions and preferences, having this option does not really do much for the user.

Also, we had hoped to be able to use the items in a user’s fridge and suggest recipes for the user to use based on what was in their existing fridge. This would be a function that we would hope to implement in the future. We had found a public API that provides functionality to find recipes based on ingredients and dietary restrictions (https://developer.edamam.com/edamam-recipe-api). We had hoped to make calls to this API but did not have the time to implement this feature. Another function we had hoped to implement was for there to be some kind of alert system used for if an item in the user’s fridge was about to expire. While we were able to get the items in the fridge to be sorted by expiration date (with the items that expired sooner listed at the top of the fridge), unfortunately this alert system was also not able to be implemented and could be another function we would hope to work on in the future. An additional function we had thought about but did not get around to was the option of auto-generating expiration dates based on a specific product. Currently, the user is required to manually implement an expiration date for each product they input, but in the future, perhaps from pulling from an API or online database this expiration date could be auto-generated or suggested, instead of having the user being required to manually implement every expiration date.