SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 7, 2021

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Audrey Hou
Sukruth Krishnakumar
Jacob Lucke

ENTITLED

Smart Pantry

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

David C. Anastasiu Digitally signed by David C. Anastasiu Date: 2021.06.06 21:52:44 -07'00'

Thesis Advisor

Nam Ling
Nam Ling (Jun 7, 2021 08:49 PDT)

Department Chair

Smart Pantry

by

Audrey Hou
Sukruth Krishnakumar
Jacob Lucke

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 7, 2021

Smart Pantry

Audrey Hou
Sukruth Krishnakumar
Jacob Lucke

Department of Computer Science and Engineering
Santa Clara University
June 7, 2021

ABSTRACT

Two-thirds of food waste at home is due to food going bad before it is used. Mitigating this food waste would mean keeping track of groceries' shelf lives, which is a time sink. Our solution is a grocery monitoring system that tracks grocery items, saves time, and makes meal planning more convenient and efficient through recipe recommendation. Tracking groceries is accomplished through using a camera in tandem with weight sensors. Grocery items are identified using computer vision to avoid the tedium of manually inputting grocery information. With the data collected, a dashboard is generated with groceries currently on the scale and their associated image.

ACKNOWLEDGMENTS

We would like to thank our advisor, Dr. David C. Anastasiu, for his advice and for helping guide us in the right direction with this project. We would also like to thank the School of Engineering for providing the funding for this project, and the Maker Lab for 3D printing necessary parts.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The EPA says 40-50 percent of food waste comes from households, with about two-thirds of food waste at home being due to food going bad before it is used. Groceries and their expiry dates are often forgotten, resulting in spoiled food. To mitigate the amount of food waste would mean keeping track of groceries' shelf lives, a significant time sink for a household that already needs time to plan and prepare meals for the week.

There are currently some meal prep services like HelloFresh that offer pre-planned meal delivery. While these save time (and actually cost around the same as buying groceries), the options are limited and you do not have complete control over portions, possibly leading to more food waste. There are also some apps that help manage groceries and track expiry dates, but these are not automated - they rely on manual entry and are tedious to use and inconvenient.

1.2 Solution

Our solution is to provide a grocery monitoring system that avoids food waste, saves time, and makes meal planning more convenient and efficient. Unlike meal prep services like HelloFresh, our solution gives the user the freedom to decide both what to cook, and how much to cook. To introduce automation, we use a camera along with weight sensors connected to a Raspberry Pi. Computer vision will identify groceries, and the weight sensors will help determine the weight of each item. Grocery data will be stored in a database, which can be viewed via a website. Alongside the camera and scale, a recipe recommendation engine will suggest recipes based on items currently in the system. We also created an Alexa Skill to introduce smart home integration to Smart Pantry. Overall, our end goal is to provide the users convenience and lower consumer food waste.

Chapter 2

Literature Survey

In this chapter, we review some current solutions to the proposed problem. Also, we note some additional considerations relating to our recommendation system and possible alternatives to Vision APIs.

2.1 Commercial Applications

Currently, the only established large-scale product that resembles our proposed solution is the technology that powers the Amazon Go stores. For those unfamiliar with Amazon Go, Amazon created stores that allow you to walk in, grab an item, and leave without paying a cashier. This revolutionary concept is powered by computer vision and various sensors, as described in the patents presented by Gyori et al. [1, 2].

Amazon Go is a very impressive idea, but is difficult to directly port to a household appliance. Amazon Go requires many cameras and sensors to keep track of each item, and each item has to be placed in a predetermined spot. One of the goals of Smart Pantry is to provide convenience to the user, and convenience means being able to place items anywhere. Additionally, introducing multiple cameras could increase the cost and complexity of the project.

2.2 Current/Similar Solutions

Food wastage is not a new problem, and neither is wanting more convenience in the kitchen. Khan et al. [3] propose a solution similar to ours, except they use multiple load cells. By using multiple load cells, they partition off parts of the cabinet, thus removing the flexibility for the user to place their items wherever they want. Sharma et al. [4] propose another interesting approach to the problem by using thermal imaging to detect the amount of food left inside opaquely packaged items. We considered using thermal imaging to detect the amount of food remaining, but instead decided to measure the weight as it is much faster and power efficient. Fujiwara et al. [5] also propose a similar solution, except they include a voice interface for labeling items. While this is an improvement over manual input, saying the name of each and every item can still be tedious. Zunnurhain et al. [6] propose an alternative approach to labeling items by scanning each item's barcode. Again, scanning every barcode could still become tedious. Even though labeling by voice or by scanning barcodes is more convenient than typing, there is still the inconvenience of labeling every item. Our goal is to completely remove the inconvenience of labeling every item through the use of object detection.

As for similar projects, Boonnuddar and Wuttidittachotti [7] created a similar tracking system using weight sensors, but for medicine. While this is an interesting application, there are many ethical and health issues involved with tracking medicine usage. For more food-related applications, Rostami et al. [8] propose a food logging system that attempts to match image/voice/text input to nutrition facts about the food. Kitamura et al. [9] propose another logging system that identifies the images of prepared food and maintains a log that is useful for diet monitoring. Identifying images of prepared dishes could be a useful addition to Smart Pantry, but we focused on identifying the individual grocery items placed on the weight scale.

2.3 Additional Considerations

2.3.1 Recipe Recommendation

Currently, our recipe recommendation system is recommending recipes solely based on ingredients in the user's Smart Pantry. Franco [10] proposes personalized recommendations based on current diet quality, and this aims to help improve the users' diets. Building a nutrition system to help users of Smart Pantry have better diets could be an interesting route, but it would create more ethical and health and safety issues.

Wang et al. [11] discuss a mechanism to incorporate flavor profiles to further improve food and recipe recommendations. Depending on its accuracy, incorporating flavor into our recommendation system could help create a novel recommendation system that retains users.

2.3.2 Image-based Information Retrieval Systems

While researching topics related to Smart Pantry, there were many papers on image-based information retrieval systems. Fu et al. [12] use cross-modal retrieval to associate images of food with recipes. Wang et al. [11] use cross-modal retrieval between recipes and images of food, with an emphasis on retrieving important nutrition facts or calories. Myers et al. [13] propose an algorithm that aims to determine the amount of calories of food in an image. Hamdan et al. [14] attempt to predict the mass of elements from images using a trained deep neural network. Accurately predicting the mass of objects in an image could significantly impact our project, making our weight scale obsolete.

Because Smart Pantry depends on taking an image of the user's groceries, image-based information retrieval systems could help add more functionality to the project. Being able to detect the amount of calories in an item on the scale could help add more information, and accurately predicting the mass of objects in an image could make our weight scale obsolete.

2.3.3 Useful Datasets

Another area of consideration is surrounding object detection. Our project uses Vision APIs, but these have a delay that contributes to an overall wait time between adding items to the scale. Because of the delay of Vision APIs, custom models that run on the system could be beneficial in terms of timing. Hou et al. [15] introduce a domain-specific dataset called VegFru which contains more than 160,000 images. As the name suggests, VegFru contains images of vegetables and fruits. This dataset is intended for domestic cooking and food management, which aligns with the scope of our project. Salvador et al. [16] introduce Recipe1M, a popular dataset of over 1 million recipes and 800 thousand food images. This dataset has been referenced in other image-recipe retrieval related papers such as [12]. Using Recipe1M, we could add capabilities to Smart Pantry for retrieving recipes from images, and then following up with suggested recipes.

Chapter 3

Methods

3.1 System Architecture

Smart Pantry was designed with cloud computing in mind. Performing most of the computing on the cloud allows the Raspberry Pi to be dedicated to gathering and offloading data, without worrying about processing power.

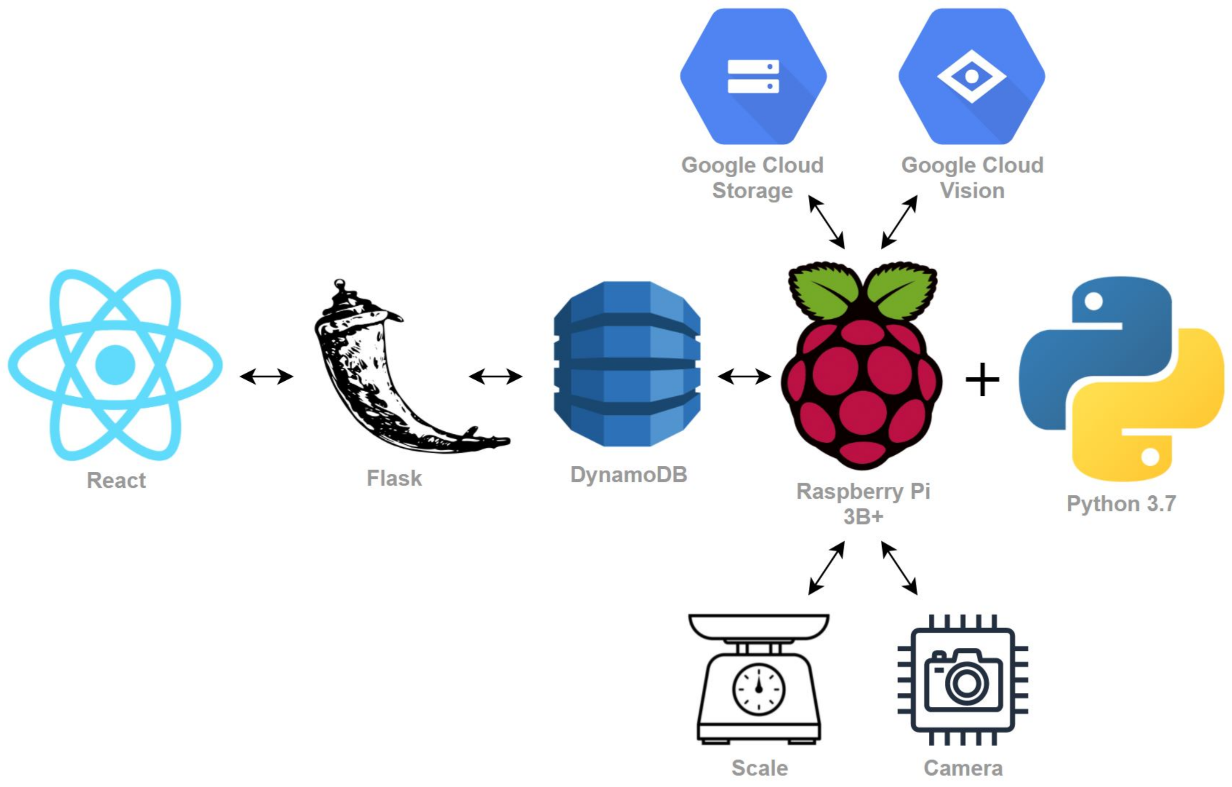


Figure 3.1: Final project architecture.

In the end, we discovered that cloud computing introduced delays which lowered convenience to the user, which is discussed more in Chapter 4. Our implemented project architecture can be seen in Figure 3.1. We decided to use DynamoDB instead of MongoDB and Elasticsearch as DynamoDB was easily accessible in our Alexa Skill. By using

Amazon Web Services, we believe Smart Pantry can be easily integrated into smart homes that may currently use an Alexa.

The Smart Pantry system starts with a scale where all the grocery items are added sequentially. When the Raspberry Pi Python script detects a weight change, the camera captures an image which is then stored in Google Cloud Storage. Google Cloud Vision runs object detection on the stored image, and sends the results back to the Raspberry Pi. Then, the Raspberry Pi determines which grocery item is the newest object placed on the scale. The newest object's data (weight, name, and date added) is uploaded to DynamoDB. On the website running React hosted in Flask, the user is able to see their grocery data and suggested recipes based on ingredients currently on their Smart Pantry scale.

3.1.1 Detecting the New Item

Since Google Cloud Vision is detecting every item in the image, it does not distinguish between new or old items. This meant we needed to find a way to determine which detected object is associated with the new item on the scale. To accomplish this, we took the structural similarity difference of the latest image with an image of the previous steady state (the scale minus the new item). Structural similarity looks at the color values and finds a contour around the most changed color values. This contour can be bound using a bounding box, thus finding our single new item.

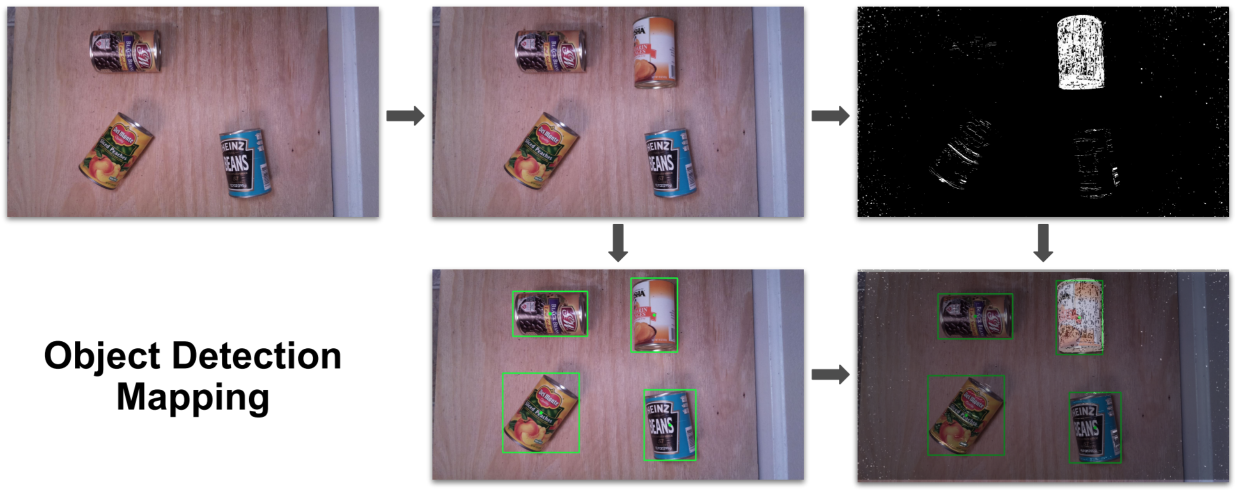


Figure 3.2: High-level representation of our object detection mapping algorithm.

Now that we have a list of all the detected objects and the bounding box of the single new item, we can determine the new detected object. To do this, we found the distance between the new item's centroid (center of the bounding box) to each detected item's centroid. The minimum distance between centroids determines the new object. A high-level representation of this algorithm is displayed in Figure 3.2.

The complexity of this algorithm is O(n), where n = number of items on the scale.

3.1.2 Individual Item Weights

Determining the weight of an item added to the scale is a simple equation, see 3.1. When an item is added to the scale, the Python script triggers the camera to take an image of the new item. This image is then sent to Google Cloud Storage, and then Google Cloud Vision runs object detection on the image.

$$weight_{item} = total_{current} - total_{previous} \tag{3.1}$$

3.1.3 Recipe Recommendation System

The objective of the recipe recommendation system was to suggest the user recipes based on their preferences from prior ratings and the items available in their pantry. We chose to implement a user-based collaborative filtering algorithm to generate the recommendations. Data was sourced from Food.com's recipe interactions dataset, which has about 230,000 recipes and ratings of over 25,000 users. After generating the recipe recommendations, the list was filtered by the list of available items in the pantry to make the recommendations more relevant.

Some of the main reasons why we chose to go with collaborative filtering is the fact that a user's food preferences are something that may evolve, and collaborative filtering enables a great adaption to this change. Collaborative filtering also allows for new recipes to be introduced which the user might not have tried on their own. At first, we began developing an item-based system, but we quickly realized the flaw of implementing this on our dataset. The users only rated a few recipes out of the 230,000 recipes, which made for a sparse distribution that was long-tailed. Also, since the ratings were concentrated in a few recipes, this was a use-case for which item-based filtering would not be ideal.

For the preprocessing steps, the users with less than five ratings were removed. Along with the dataset, there was a tokenized ingredient list that reduced down variations of the same ingredient name to its stem. Cosine similarity was used as the similarity score metric, and the formula is given below in 3.2. It is calculated from two vectors, A and B, and is bounded to the range [0,1]. In this case, the vectors represent the ratings that the two users recorded in order to calculate a similarity score between them.

$$sim = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.2}$$

Once the similarity scores were obtained, the k-nearest-neighbor approach was used to find the top 200 similar neighbors to generate recipe recommendations. After the recommended recipe list is generated, the recipes are filtered by these tokenized ingredients depending on the items in the user's pantry.

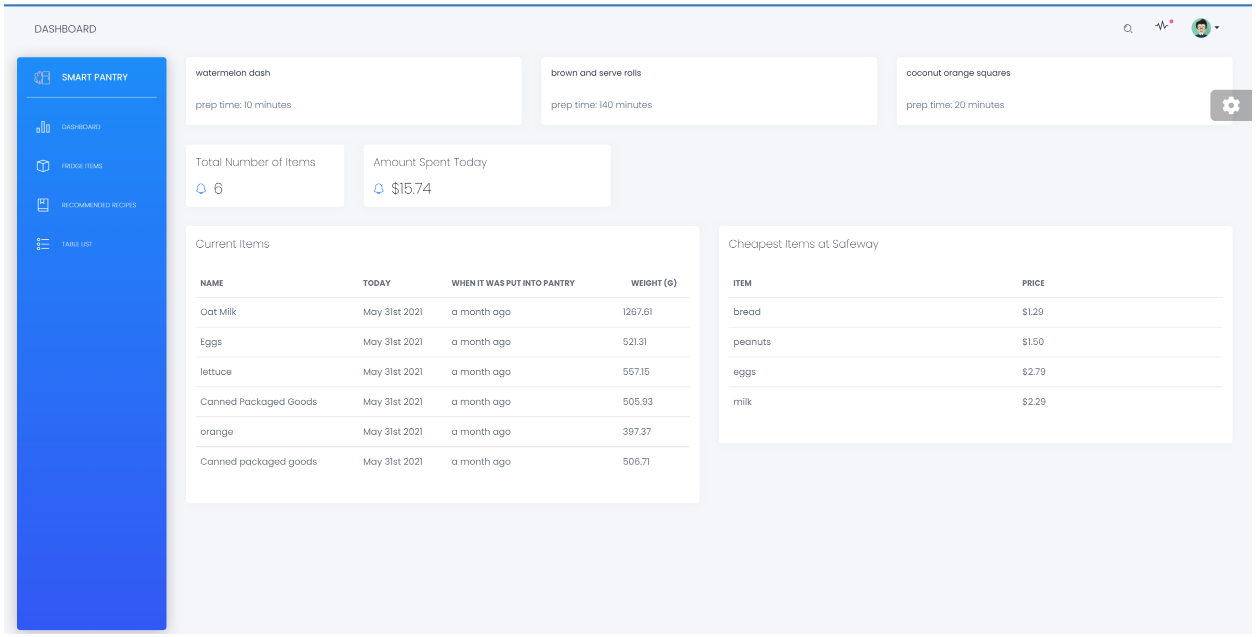
3.2 Web Application Design

3.2.1 Web Server

In order to display the information about items in the Smart Pantry system onto a web application, a server is required to send and retrieve the data to and from the database. This server is built with Flask, a web framework written in Python, where it can easily call the data from the database, DynamoDB, which is accessible via Python. As a result, all that is required to build the API for the server is to call a Python function and return the data as a JSON file.

GET and PUT requests were written for the API to fetch the pantry items from the database and the recipes that are generated by the recipe recommendation system. The front end can then make API calls from the server through Axios, a promise based HTTP client, and then generate ReactJS components based on each element from the data in the JSON file.

3.2.2 User Interface Design



watermelon dash
prep time: 10 minutes

brown and serve rolls
prep time: 140 minutes

coconut orange squares
prep time: 20 minutes

Total Number of Items
6

Amount Spent Today
$15.74

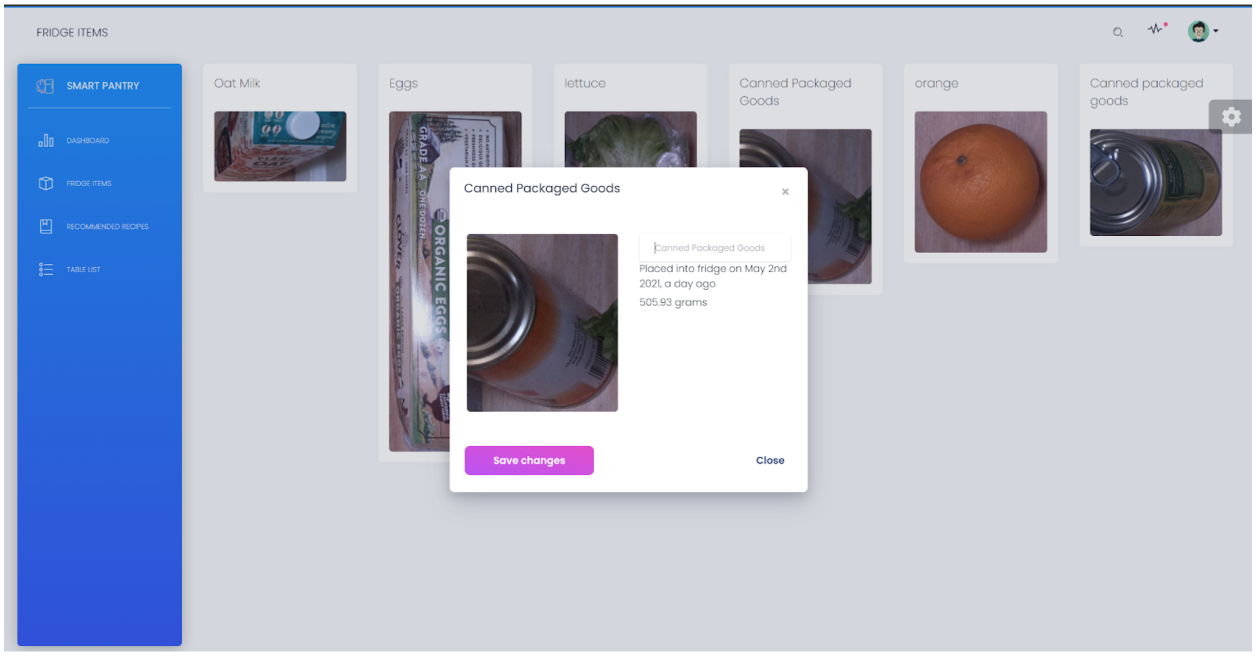
Current Items

NAME	TODAY	WHEN IT WAS PUT INTO PANTRY	WEIGHT (G)
Oat Milk	May 31st 2021	a month ago	1267.61
Eggs	May 31st 2021	a month ago	521.31
lettuce	May 31st 2021	a month ago	557.15
Canned Packaged Goods	May 31st 2021	a month ago	505.93
orange	May 31st 2021	a month ago	397.37
Canned packaged goods	May 31st 2021	a month ago	506.71

Cheapest Items at Safeway

ITEM	PRICE
bread	$1.29
peanuts	$1.50
eggs	$2.79
milk	$2.29

Figure 3.3: Homepage of the web application.



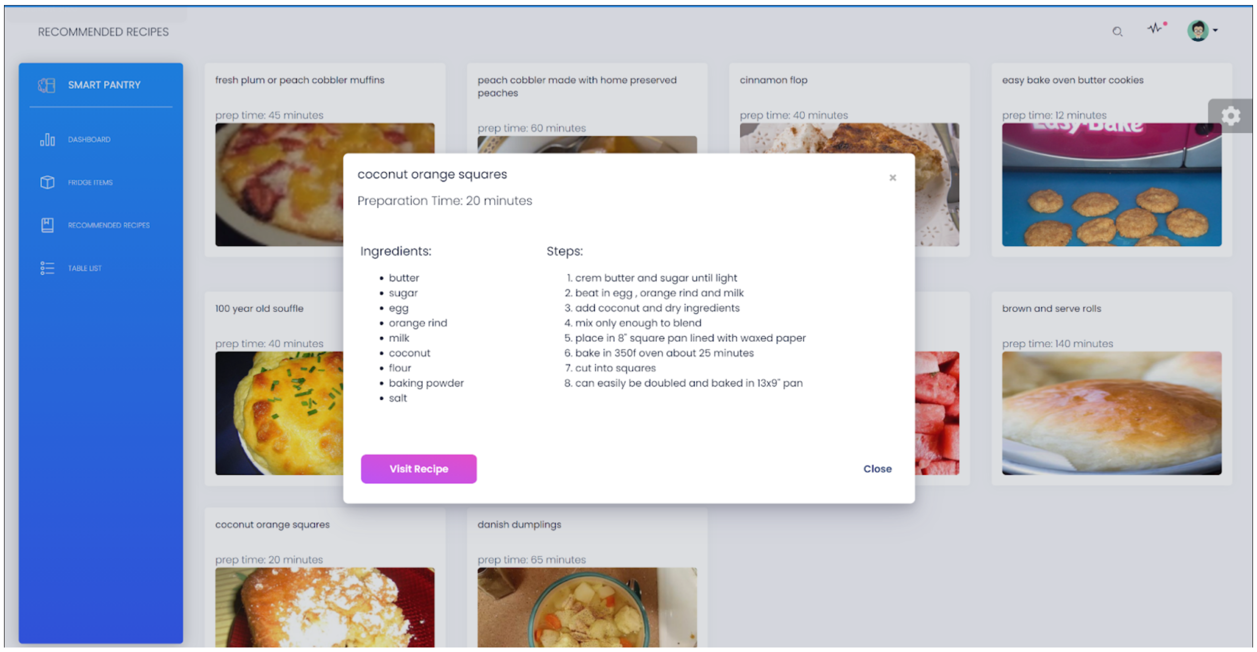
Canned Packaged Goods

Canned Packaged Goods

Placed into fridge on May 2nd 2021, a day ago

505.93 grams

Figure 3.4: Fridge items page of the web application.



coconut orange squares

Preparation Time: 20 minutes

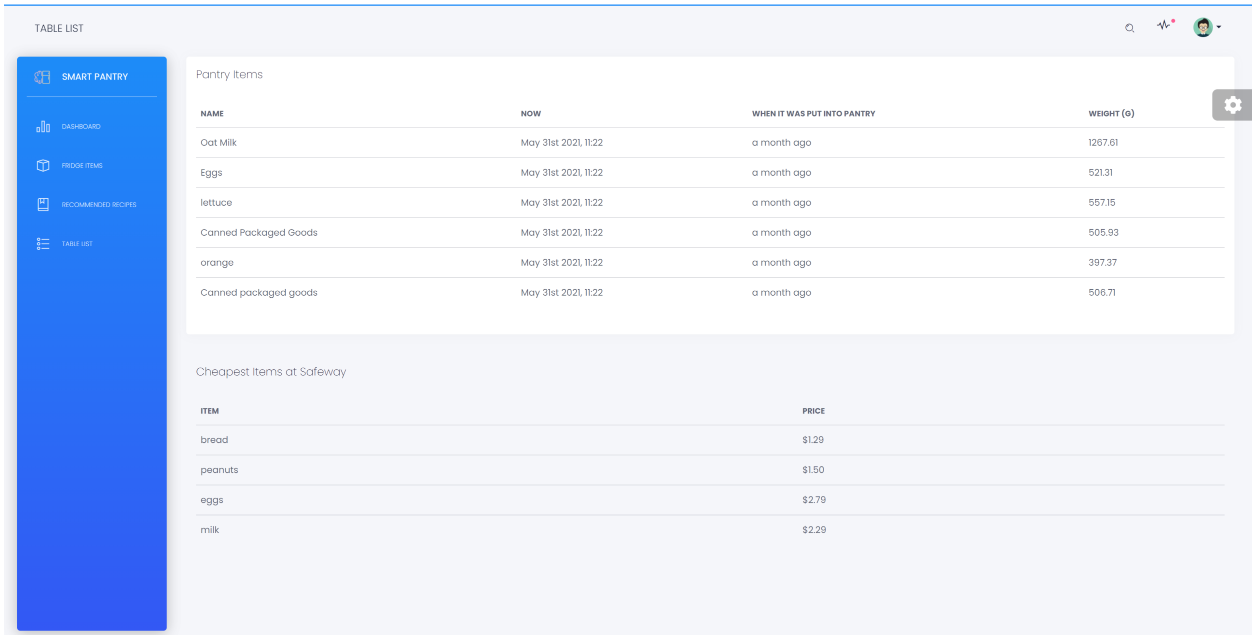
Ingredients:

- butter
- sugar
- egg
- orange rind
- milk
- coconut
- flour
- baking powder
- salt

Steps:

1. crem butter and sugar until light
2. beat in egg , orange rind and milk
3. add coconut and dry ingredients
4. mix only enough to blend
5. place in 8" square pan lined with waxed paper
6. bake in 350f oven about 25 minutes
7. cut into squares
8. can easily be doubled and baked in 13x9" pan

Figure 3.5: Recommended recipe page of the web application.



Pantry Items

NAME	NOW	WHEN IT WAS PUT INTO PANTRY	WEIGHT (G)
Oat Milk	May 31st 2021, 11:22	a month ago	1267.61
Eggs	May 31st 2021, 11:22	a month ago	521.31
lettuce	May 31st 2021, 11:22	a month ago	557.15
Canned Packaged Goods	May 31st 2021, 11:22	a month ago	505.93
orange	May 31st 2021, 11:22	a month ago	397.37
Canned packaged goods	May 31st 2021, 11:22	a month ago	506.71

Cheapest Items at Safeway

ITEM	PRICE
bread	$1.29
peanuts	$1.50
eggs	$2.79
milk	$2.29

Figure 3.6: Table list page of the web application.

The user interface (UI) was designed with the intention of making the data collected from the pantry easily available for users to view. As a result, a dashboard was chosen for the UI.

The main language that was used to build the front end was ReactJS, which makes it easy to create reusable web components. These components can be generated based on the data from the database, where for every item in the table, a component can be generated for display. The components, such as display cards, can contain the information and image of an item in the pantry that is fetched from the database.

The homepage of the dashboard, displayed in Figure 3.3, consists of the top recommended recipes for the user, an overview table of all the items in the pantry, basic statistics on the items in the pantry, and the navigation menu as a sidebar for easy navigation through pages. The recipes are automatically generated as cards based on the GET request to the database, where the user can click into it to view the whole recipe. The table of items contains the name, time it was put into the pantry, how long it has been in the pantry, and the weight of the item.

In the Fridge Items page, displayed in Figure 3.4, a grid of the items in the pantry will be displayed through a card (per item in the pantry) that contains the name of the item and an image of the item. These cards are generated based on the GET request to the database. Once the user clicks into the card, more information about the pantry item is displayed through a pop-up: the name of the item, when it was placed into the fridge, and the its weight. If the user thinks that the item is not named properly through the image detection, they can double click on the name of the item to change the item name. This will send a PUT request to change the item's name in the database once the user hits the ”Save Changes” button.

In the Recommended Recipe page, displayed in Figure 3.5, a grid of the recipes recommended by the recipe recommendation system will also be displayed through a card (per recommended recipe) that contains the name of the recipe and an image of the dish. These cards are generated based on the GET request that fetches the data from the recipe recommendation system. Once the user clicks into the card, more information about the recipe is displayed through a pop-up: the recipe name, the preparation time, ingredient list, and the steps to make the dish. The user can also choose to visit the original site of the recipe or close the pop-up.

In the Table List page, displayed in Figure 3.6, a table list of the items in the pantry is displayed. This table list is generated based on the GET request that fetches the items in the pantry from the database. Each row of the database is an item from the pantry which contains the item name, the current time, when the item was put into the pantry, and the weight of the item. This serves as a more detailed overview of the items in the pantry.

Chapter 4

Evaluation

4.1 Data Set

Food.com Recipe Interaction Dataset

This contained a set of over 230,000 recipes from food.com. Each recipe had details such as a list of ingredients, steps, a recipe name, and a unique id. A complimentary table contained anonymized user interaction data for about 25,000 users and contained fields such as user unique id, list of recipes rated, and the actual ratings. There was also a table that mapped raw ingredient names to processed (tokenized) ingredient names.

4.2 Evaluating the Recommendation System

A popular metric for evaluating recommendations is Mean Reciprocal Rank (MRR). This metric accounts for how far down the relevant recommendations occur in the generated recommendation list. In formula 4.1, Q is the number of sample queries and $rank_i$ is the position of the first relevant document for the i-th query.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{4.1}$$

For measuring this score, a subset of the highly-rated recipes was segregated into a test group. The intuition is that highly rated recipes are some of the most relevant for the user. A list of recommendations was generated and the list was searched to find the recipes that were part of the test group. The results were dismal and there were many zero scores observed. We hypothesized this was due to the long-tailed and sparse nature of the dataset; the recipe that one user rated highly may not be rated by many other similar users as there were ten times more recipes than the number of users, and most users rated only a handful of recipes.

An alternate metric that we used to evaluate our recommendations was the Mean Average Error (MAE). This accounts for the average discrepancy between the predicted rating (y_i) and the actual rating (x_i) for each recipe in the test group, see 4.2.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{4.2}$$

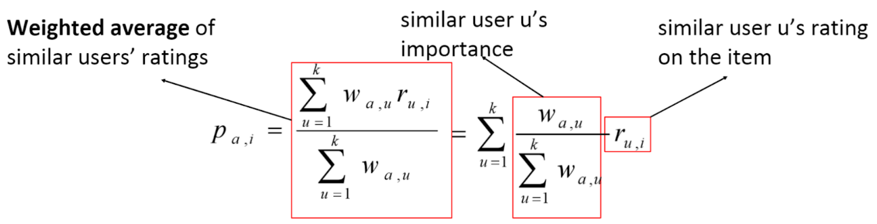

Weighted average of similar users' ratings
similar user u's importance
similar user u's rating on the item
$p_{a,i} = \frac{\sum_{u=1}^{k} w_{a,u} r_{u,i}}{\sum_{u=1}^{k} w_{a,u}} = \sum_{u=1}^{k} \frac{w_{a,u}}{\sum_{u=1}^{k} w_{a,u}} r_{u,i}$

Figure 4.1: Predict rating formula.

For each user, three recipes were set aside as part of the test dataset. Then, using the rest of the ratings as training, a predicted score was calculated using the formula as shown in Figure 4.1, factoring in the cosine similarity and k-nearest-neighbor approach.

These predicted scores were compared against the actual ones, and it was found that the average error was 0.16 in absolute terms. On a scale of 0-5 inclusive, this translates to around a 3.2% error margin. This means our recommendation engine was able to predict a given user's rating on recipes fairly accurately, missing the actual score by 3.2% on average.

4.3 Vision API Evaluation

We had two choices when deciding which Vision API to use for object detection: Microsoft Azure Custom Vision and Google Cloud Vision. To determine which Vision API to use, we tested the efficiency of both APIs and compared the results. Having an efficient Vision API was crucial for maintaining a small wait period between adding items to the scale.

4.3.1 Efficiency Results

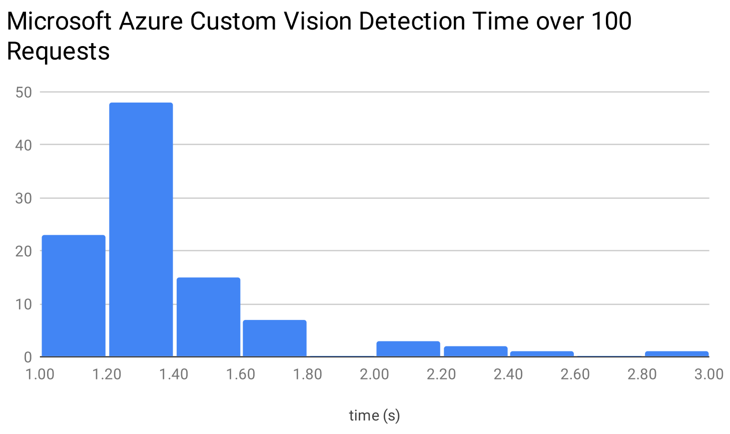


Figure 4.2: Histogram of request times to Microsoft Azure Custom Vision.

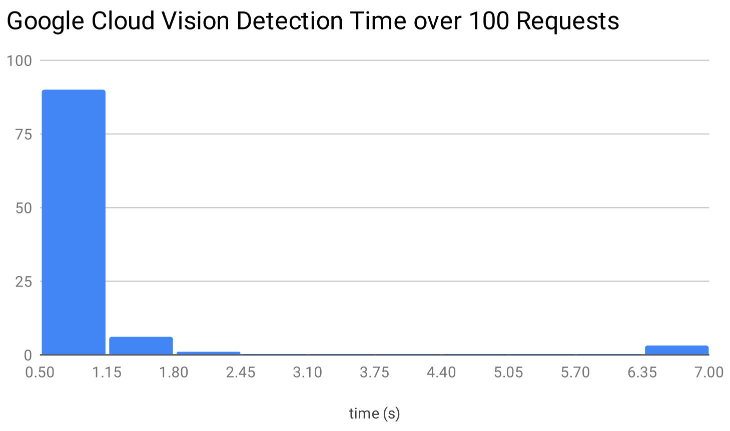


Figure 4.3: Histogram of request times to Google Cloud Vision.

For Computer Vision APIs, we tested Google Cloud Vision and Microsoft Azure Custom Vision. To compare the two APIs, we ran an object detection request 100 times. As seen in Figure 4.2 and Figure 4.3, the Google Cloud Vision average time was significantly faster than the Microsoft Azure Custom Vision average time. Because of our emphasis on convenience for the user, we decided to use the more time efficient Google Cloud Vision.

Chapter 5

Future Work

After reaching a minimum viable product, some future work is apparent. Our major focus for this project was to provide convenience for the user. However, as discussed in Chapter 4, the downtime required between placing items on the scale could be improved upon to further improve convenience for the user.

5.1 Time Efficiency

5.1.1 Multithreading

When implementing the new object association algorithm, multithreading was used to exploit the benefits of concurrency and reduce the delay between items. After completing this, we realized that by spawning a thread for each new item placed, we could significantly reduce the wait time between placing items. Incorporating multithreading into the core of the system is a possible future task to help improve convenience.

5.1.2 Custom Food Model

Another bottleneck we encountered while developing Smart Pantry was the time cost of API calls to Google Cloud Vision. As shown in Figure 4.3, API calls added an average delay of about 1 second. On top of calculating the structural similarity of images, determining the new object on the scale, and uploading to the database, this added second lowers convenience for the user by making them wait longer between adding items to the scale. By training a custom food model specifically designed to detect items on the Smart Pantry scale, items could be detected much faster, therefore lowering the delay between items.

5.1.3 Batch Uploads

Another design oversight is related to how the data collected by the camera-scale system was uploaded to the database. In an effort to have the website updated in real time, data was uploaded to the database as it was collected. Without multithreading, this meant an item's data had to be uploaded to the database before the system could start processing the next item to be added. One solution to this issue is to spawn a thread that uploads the data while the main process continues with the next item. Another solution is to maintain a local record of the data to be uploaded, and upload the batch of data when the user is finished adding items, thereby avoiding the database delay between items.

5.2 Web Application

5.2.1 User Authentication

User authentication and account information was one of the things we had in plan for the system, especially considering that a pantry and its items is based on the user. The plan was to create this so that there would be a database of pantry items per user, along with the recipe recommendations that are built on top of the items in their pantry.

The user's information should be stored in a database that contains the user's username and encrypted password from when they register and enter their password. The encrypted password will be saved for login, and then it will select the user and their information based on their user id. The flask server will have the API for managing the user ids and hashed passwords through a flask authentication package. The database should also contain the settings that the user has saved, as well as items that the user has used before.

On top of basic user authentication, OAuth can also be set up with the system for users to create profiles in Smart Pantry with their Google, Facebook, or other social media accounts instead of creating a username and login for the Smart Pantry platform.

5.2.2 Notification generation

Once user authentication is built into the system, notifications should also be built on top of the data from the user. The notification that we planned to incorporate into the system is to notify users of how long an item has been in the pantry. This would come from a daily check on the items in the user's pantry compared to a database of expiry times. If the item has been in the pantry for too long, it will be marked as a notification for the day, and it will show up in the application for the user to view as a reminder.

5.3 Recipe Recommendation

5.3.1 Hybrid Recommendation Engine

The current system is based on user-based collaborative filtering and a simple search algorithm to match ingredients of the recipe and the items in the pantry. A more sophisticated way would be to use a content-based recommender algorithm utilizing bag-of-words or TF-IDF to more effectively filter the recommendations based on items available in the pantry. A hybrid scoring model could be adopted with appropriate weights for both content and collaborative filtering methods to achieve a more robust recommendation engine.

5.3.2 Faster computations

In the current implementation, dense data structures are used for all computations. Using sparse data structures will help in making the recommender functions faster. Additionally, any pre-processing on the user interaction or recipes dataset can be stored as .pkl files, which are a way to quickly export dataframe structures in Python, so that these steps do not have to be repeated each time the API endpoint for recommendation is hit.

5.3.3 Explore other similarity measures

Cosine similarity with the k-nearest-neighbor approach was used in this project, but additional similarity measures or item-based filtering can be explored more in-depth. Testing the accuracy of predictions on various similarity measures and using it to tune hyper-parameters is the next step to optimize the recommendation engine.

Chapter 6

Societal Issues

Ethical. Our product could unintentionally contribute to unhealthy eating habits, eating disorders, or addictions by recommending unhealthy recipes to users. It could lead to unhealthy recipe suggestions if their pantry consists of unhealthy ingredients. On the other hand, our product is not a health application, and the function is not to provide healthier food suggestions. This puts our product in a moral gray zone, where our product could potentially reinforce bad habits, but it is in reality not trying to reinforce habits at all - good or bad.

Social. The target user group for Smart Pantry is broadly defined, but its use may be limited by the cost or learning curve required to effectively use the product. Another social issue would be the assumption that people have normal access to internet. The database of grocery items is stored in a cloud system where it allows for real time updates based on the internet.

However, the recommendation system also has a flaw. Promoting the healthiest option, or the option that is best fit for their pantry items is not necessarily the best one. A good recommendation must consider personal food preferences and healthiness to suggest the correct amount of ‘healthy and tasty’ food. The best recommendation should be given at the appropriate time to motivate people. Rostami et al. [17].

Political. We believe our project has a low political impact. Smart Pantry was designed to be used by any person, regardless of their political beliefs. Furthermore, Smart Pantry would not be distributed by the government. Because of this, we do not foresee any political issues with our project.

Economic. With the current stage of the project, the only economic impact is possibly reducing a user’s money spent on groceries. However, one possible addition to Smart Pantry could be automatic online grocery ordering. This introduces an economic impact on small businesses if many users opt-in to online ordering, an option usually only available for larger grocery stores.

Health and Safety. As discussed in the Ethical issues paragraph, our product poses some health and safety issues. Our product has the potential to unintentionally reinforce bad habits, eating disorders, or addictions.

Manufacturability. One manufacturing issue is the design of the scale. Every home is slightly different, so our current prototype may fit in some homes and not in others. Having different sizes of scales could increase the cost of manufacturing.

Sustainability. Our project's goal is to help sustainability by lowering food waste in the household. By providing the user with data on how long grocery items have been in their pantry, we hope to reduce food waste that results from forgotten food.

Environmental Impact. Lowering food waste lowers greenhouse gases, and aids positively in combating the global scarcity. However, more frequent restocks of individual items may mean higher trips to the grocery store, and this adds to the transportation carbon footprint.

Usability. The Smart Pantry system is being designed as an add-on device, but due to the variety of kitchens in all homes, compatibility may be an issue. There is a learning curve involved to making effective use of features such as tracking grocery consumption. This device would be intuitive for people who have used other smart home devices such as thermostats or fitness trackers.

Lifelong learning. This product helped us learn about the Internet of Things (IoT), weight sensors, camera sensors, machine learning, databases, user friendliness, and user design.

Compassion. Our project aims to relieve the stress of buying groceries, preparing, and making food by making the user's food information accessible.

Chapter 7

Conclusion

Identifying and tracking grocery usage can be a powerful tool to enable cutting down on inadvertent food waste. Through our proof of concept, we have shown the merits of a semi-autonomous process for grocery management which emphasizes user convenience. Adding on features such as tracking expiry and basing recommendations on this statistic can make the system a worthwhile alternative to how groceries are currently managed.

7.1 What We Have Learned

One of our most important realizations was about the importance of seamless integration of different software components. Our system had multiple components to it: an IoT device (Raspberry Pi) which was hooked up to the sensors and cameras, a cloud NoSQL database, a backend system to call image recognition APIs and generate the recipe recommendations, and the UI to present all the information in a digestible way. While building out the individual components was relatively straightforward, interfacing each part of the system to work together coherently was a challenge.

Another important lesson we learned was the challenge of identifying and defining the scope of the project. Like other real-world software projects, there is always a limitation on time and resources. Figuring out which features to focus on and defining the product's core competence is a challenge. Initially, there were many solution spaces we wanted to explore and build features for, but having objective discussions with the team was something that helped us tackle this challenge.

Technically, we learned that while image detection APIs are very advanced at identifying general objects, there were few plug-and-play solutions for specific use cases such as food identification.

7.2 Why it is Important

Our takeaways from this project were immense. We learned how to build all aspects of a functional consumer product and how to do so with given constraints. This is very applicable to engineering in the real world and we are grateful to have this exposure. Also, we particularly enjoyed creating a product designed to help people, and we look forward to having the opportunity to help others through our jobs after graduation.

Bibliography

[1] B. J. Gyori, I. Medrano, A. M. Frenkel, and P. N. Java, "Instrumented item stowage system with modular elements," June 19 2018. US Patent 10,001,402.

[2] B. J. Gyori, I. Medrano, A. M. Frenkel, and P. N. Java, "Shelf with integrated electronics," Sept. 4 2018. US Patent 10,064,502.

[3] M. A. Khan, M. H. B. Shahid, H. Mansoor, U. Shafique, M. B. Khan, and A. u. R. Khan, "Iot based grocery management system: Smart refrigerator and smart cabinet," in *2019 International Conference on Systems of Collaboration Big Data, Internet of Things Security (SysCoBIoTS)*, pp. 1–5, 2019.

[4] A. Sharma, A. Misra, V. Subramaniam, and Y. Lee, "Smrtfridge: Iot-based, *User Interaction-Driven* food item & quantity sensing," in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, (New York, NY, USA), p. 245–257, Association for Computing Machinery, 2019.

[5] M. Fujiwara, K. Moriya, W. Sasaki, M. Fujimoto, Y. Arakawa, and K. Yasumoto, "A smart fridge for efficient foodstuff management with weight sensor and voice interface," in *Proceedings of the 47th International Conference on Parallel Processing Companion*, ICPP '18, (New York, NY, USA), Association for Computing Machinery, 2018.

[6] K. Zunnurhain, K. Nayee, R. Patel, and P. Patel, "Always fresh: Tracking food expiry with mobile app," *J. Comput. Sci. Coll.*, vol. 35, p. 223, Oct. 2019.

[7] N. Boonnuddar and P. Wuttidittachotti, "Mobile application: Patients' adherence to medicine in-take schedules," in *Proceedings of the International Conference on Big Data and Internet of Thing*, BDIOT2017, (New York, NY, USA), p. 237–241, Association for Computing Machinery, 2017.

[8] A. Rostami, B. Xu, and R. Jain, "Multimedia food logger," in *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, (New York, NY, USA), p. 4548–4549, Association for Computing Machinery, 2020.

[9] K. Kitamura, T. Yamasaki, and K. Aizawa, "Food log by analyzing food images," in *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, (New York, NY, USA), p. 999–1000, Association for Computing Machinery, 2008.

[10] R. Zenun Franco, "Online recommender system for personalized nutrition advice," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, (New York, NY, USA), p. 411–415, Association for Computing Machinery, 2017.

[11] H. Wang, D. Sahoo, C. Liu, E. Lim, and S. C. H. Hoi, "Learning cross-modal embeddings with adversarial networks for cooking recipes and food images," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11564–11573, 2019.

[12] H. Fu, R. Wu, C. Liu, and J. Sun, "Mcen: Bridging cross-modal gap between cooking recipes and dish images with latent variable model," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14558–14568, 2020.

[13] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy, "Im2calories: Towards an automated mobile vision food diary," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1233–1241, 2015.

[14] M. Hamdan, D. Rover, M. Darr, and J. Just, "Mass estimation from images using deep neural network and sparse ground truth," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1987–1992, 2019.

[15] S. Hou, Y. Feng, and Z. Wang, "Vegfru: A domain-specific dataset for fine-grained visual categorization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 541–549, 2017.

[16] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba, "Learning cross-modal embeddings for cooking recipes and food images," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3068–3076, 2017.

[17] A. Rostami, V. Pandey, N. Nag, V. Wang, and R. Jain, "Personal food model," in *Proceedings of the 28th ACM International Conference on Multimedia (MM '20)*, MM '17, (New York, NY, USA), Association for Computing Machinery, 2020.