CS 125 Project Report

Dish It

Madison Thompson: madisoat@uci.edu

Kerry Cheng: kerryyc@uci.edu

Huu Lam: huul1@uci.edu

Benjamin Chen: benjamyc@uci.edu

Abstract

This paper will define the problem our system will attempt to solve as well as detail the specifications of our implemented system. The use of new and emerging approaches to generate data in our system has aided in the development of a next generation search systems. New approaches to data collection like micro-blogs and micro-reports has helped create a more personalized experience for our user. Location also has played a key role to creating a personalized experience. Using sensors to generate a dynamic data stream for our user will ensure we solve the problem efficiently.

Through studying related systems and their practices new and evolving approaches to data collection and analysis can be learnt. With the rise of social media many new approaches to relaying data to a user have been explored and studied in order to gain insight on new user evaluation on systems. The system uses multiple APIs many using next generation approaches to generate the data that is used to output personalized data to the user. Ensuring our users are satisfied with our system evaluating techniques are compared based on overall detection of user interface problems.

Introduction and Problem

Today, health is one of the elements that leads humans to decide what to eat. The topic has been researched by scientists, and businesses also involved to provide service that help consumer decide. Along with knowing what to eat, consumers also need to find out where to eat. Self-cooking was one of the best options, but in today busy life, consumers are more likely to

look for a restaurant as a source for meals. There also is a great amount of effort to find a suitable destination that meets a person's needs.

Understanding the demands of consumers, our group has developed an application that will automatically track and analyze users' eating habit and health conditions, then provide food recommendations based on those data. We will take advances of the sensor technology to help our app collecting users' data without getting input directly from users. There will be three key factors for the app giving recommendations which are location, calories, and user eating habits. The app will recommend specific food, and where to get it. We expects the app collect necessary input from users, APIs, system's real-time data (time, date, location,…), and from sensor devices (such as smartwatch,…).

The application that we built can automatically collect data, analyze it, and give feedback to user, so that user can take advances from their personal data with as less actions as possible.

Current State of the Art

Before and during the implementation of our application we looked at different aspects of current systems that solve similar problems. This aided us in finding a suitable approach for solving our problem.

Urbanspoon (Zomato):

This application generates restaurant recommendations based on location, reviews, and preferences entered by the user. This application has a unique way of collecting data. Zomato

employs a data collection team that uses a “feet-on-the street” approach where team members get first-hand experiences in a city so they do not have to rely on secondary resources. The user profile stores information like the reviews they have written, photos they have uploaded, and place they have bookmarked. In order to filter out bad reviews, users can be verified as a trustworthy user. To become verified the application uses a point system for users to move up a level as they enter information about restaurants they have visited.

The problem we are solving is similar to the one Urbanspoon solves, however there are many differences between the two. This application only learns about the users actions within the application like writing reviews and recent searches. Where as our application learns about the user’s actions outside the application like frequent locations and calorie intake. Our application automatically recommends a dish along with where they can get it based on the information retrieved from their profile and FitBit API. Urbanspoon requires the user to filter theri results each time the search their system for restaurants. Urbanspoon is more similar to Yelp then to our application.

Yelp:

Yelp allows you to window shop a restaurant before ever going. You can look at the menu, reviews, and see photos of the dishes prepared before ever going to the restaurant. In our application we use Yelp as a data source to generate a recommendation for the user based on their preference and past eating behavior. Yelp’s recommendation software is automated to filter reviews by users’ quality, reliability, and user activity. This ensures that advertisers and unreliable users don’t affect the recommendations and reviews generated by Yelp. This

approach to filter data will be useful to this project. The difference between our application and Yelp is that Yelp uses content-based analytics. "Content-based analytics focuses on the data posted by users on social media platforms, such as customer feedback, product reviews, images, and videos." [7] Yelp's main data source is user input. The manipulation of that data is where Yelp is successful. Our approach to the problem focuses more on predictive analytics. "At its core, predictive analytics seek to uncover patterns and capture relationships in data." Using historical and current data our system will attempt to provide a recommendation of a food dish based on a user's past and present food preferences as well as current calorie intake.

The Yelp gathers information from the user to produce better recommendations for the user. In a profiel a user can enter things like their favorite foods,restaurants, hobbies, websites, etc. In our system we modeled our profile similar to Yelp's as to get the most information about the user to produces the best recommendations. Of all the current applications that are similar to ours Yelp has been the most helpful to our implementation.

FourSquare:

FourSquare is a restaurant recommendation application that allows user to tailor their recommendation based on their taste. When user first create their profile they are able to add tastes to their profile that are dynamically generated by the system. These tastes range from specific dishes to restaurant atmospheres. When the system search for recommendations it filters them based on the users tastes. The tastes are saved to the user profile so they do not have to filter their search everytime. A user tastes can be edited to add new tastes or delete old taste. This use of tastes is similar to the use of preferences in our system. This reduces the amount of data the user has to input to generate recommendations.

FourSquare also has an industry-leading location technology, that "includes an always-updating map of 93M locations worldwide, Foursquare Analytics accurately measures visits to thousands of retail brands each month". Our system ideally would employ technology of this caliber to track restaurants the user goes to and locations they frequently visit. This also allows store fronts access dynamic foot traffic dashboards. By solving their problem to make timely recommendations to FourSquare users they were able to solve a larger problem for

ChefsFeed:

Chefs provide data to ChefsFeed. They are able to make specific dish recommendation to other users that they have enjoyed. The application sorts these recommendations based on the location the user has entered. ChefsFeed main data source is from the users, specifically a chef. Instead of having an automated filter like Yelp, ChefsFeed filters data based on the user type. Only chefs are allowed to make a recommendation. Aside from the basic description there isn't any documentation of any other data sources or analysis other then the clear presentation of user generated data. Their dish recommendations can possibly be used as a data source because our system will recommend dishes to the users rather then specific restaurant like Yelp and UrbanSpoon.

ChefsFeed API is used by applications like FourSquare and Urbanspoon. These applications use ChefsFeed to access recommendations directly from chefs. Even though we do not use Chefsfeed as a data source in the future this API could provide our application more interesting dishes to recommend to the user.

Data Analysis:

Image recognition and analysis will be a vital component to our application. The user is able to submit a picture of their meal to gain information about individual food's nutrition levels. For example, if the user uses a picture of pasta to search, the application is able to discern individual ingredients in the pasta such as the pasta noodles and pasta sauce. Such a feature would rely on object-based image analysis. Object-based image analysis in its essence partitions "remote sensing imagery into meaningful image-objects, and assessing their characteristics through spatial, spectral and temporal scale." [1] This type of image analysis would ultimately "result in increased repeatability and production, while reducing subjectivity, labour and time costs."[1] Our application will analyze what the user eats during a given meal and use the data to determine whether certain potential foods in later meals would be within the user's calorie count. Using an API that utilizes object-based image analysis would greatly improve the efficiency of analyzing images that a user would submit in the application. Additionally, object-based image analysis would also allow the application to better determine individual food items and thus make better calculations on the dish's nutritional value. Because "several OBIA methods/commercial software packages build upon the powerful object-oriented paradigm," [1] we can also utilize object-oriented programming methods in our code to take advantage of APIs that we choose to use in our application for performance and efficiency related reasons. One example of an API that we could potentially use is Clarifai. A study on hyper-resolution monitoring of urban flooding with social media and crowdsourcing data used the Clarifai API to analyze pictures of flood sites. The study explains that the API "service receives feedings of individual photos and feedbacks a list of tags that describe the objects present too busy to clean in the

photo." [2] In the feedback list includes the tags in which the Clarifai service determines what it thinks the image depicts and also includes the probability that "[quantifies] its confidence of the recognition."[2] Our application could effectively utilize such an API to categorize different types of foods and thus also create more accurate recommendations for other types of foods and restaurants that serve that particular type of food.

Our application's primary purpose is to recommend types of foods that is beneficial in maintaining the user's daily calorie count. Many people today rely on fitness tracking services to check up on the status of their overall health. It also provides a convenient means of tracking various health analytics that is easily accessed through the user's phone application. It is important to note that users should not be completely dependent on fitness trackers as overall indicators for accessing their health status. As Patel, Asch, and Volpp explain, maintaining good health comes from mostly self-motivation as "most health-related behaviors such as eating well and exercising regularly could lead to meaningful improvements in population health only if they are sustained."[3] Moreover, health data shown in any application "must be presented back to the user in a manner that can be understood, that motivates action, and that sustains that motivation toward improved health." [3] Thus, our team has taken these considerations into account when developing the application. The user needs to trust that the information presented to them is accurate and applicable to their daily habits. This in turn would encourage the user to use our application regularly and also potentially better their lifestyle. Everyone's health conditions are unique, therefore it is important to gather accurate health data from a data source that is both readily accessible and reliable. According to Kelman, Bass, and Holman, "linked health administration data can provide an unparalleled resource for the monitoring and evaluation of

health care services." [3] Having a standardized source of information would mean faster access to such data as well as yielding more accurate health data. In doing so, however, it is important to guarantee the user that their health data is secure and that they should not be concerned with "the possibility of unauthorized copies of data being made and distributed or data being used for other than agreed purposes." [4] Our team has taken advantage of existing fitness tracking APIs to gather information regarding the user's height, weight, calorie goals, etc. However, each fitness tracking application has different algorithm implementations in calculating certain fitness goals, not to mention the fact that different fitness tracking hardware could possibly cause some inaccuracies in calorie tracking. Potentially, future iterations of our application can pull data from public health records to better enhance health tracking.

Architecture

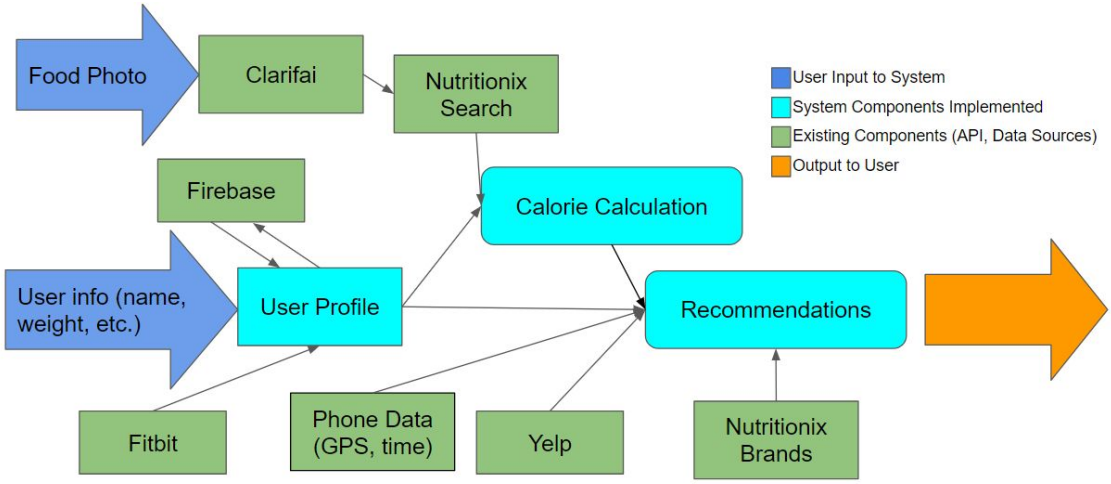


Figure 1: Software diagram of Dish It system

*All specifics on these components and how they work can be found in the Components and Algorithms section.

In our system, the only two manual inputs needed are the pictures of the food and the user information such as name, weight, and gender. The food photo is sent into the part of the system that calculates the total calories contained within the photo by sending it to Clarifai and then to Nutritionix. The user information is stored in the form of the user profile and is saved to Firebase. After initial profile creation, the system both reads and updates information from Firebase as well.

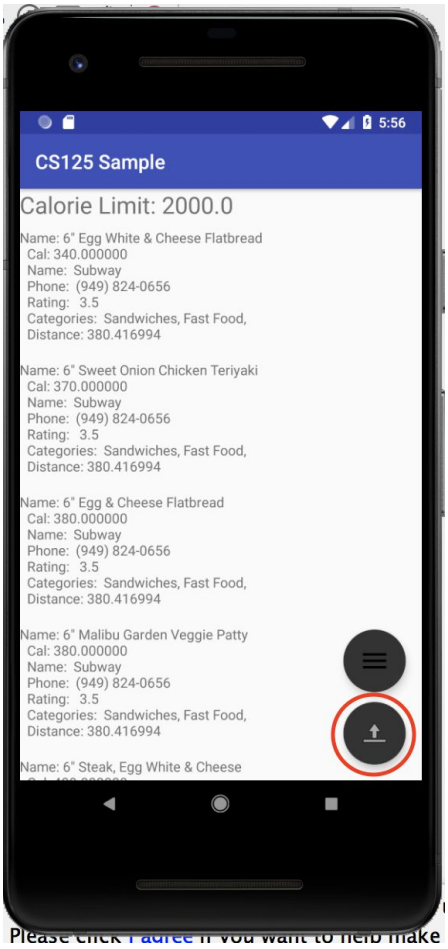
The two live data streams in the system are from Fitbit, where a user's health and activity information is pulled, and from their phone which has data on the user's current location and time. Both of these streams are used in the recommendation algorithm. Nutritionix serves as a persistent data source by providing constant information on the calories of different types of food. Yelp is both a persistent data source as it provides information on restaurants and an episodic data source as it provides reviews about that restaurant.

Query/Input Environment

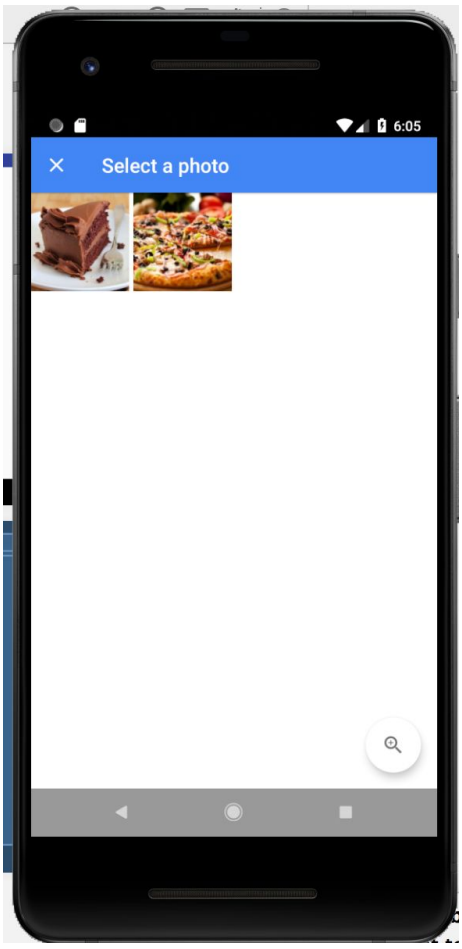
Our application allows user to search the total calories of their meal and create a profile to better filter their recommendations. The search system we have implemented allows the user to upload a picture to the application and it return a list of possible foods that their meal contains. From that list the user can select the correct food and the total calories of the meal is returned.

Our system also allows for the user to input information via their profile. The user can can select preference from a list or can enter their favorite foods and the food they dislike. The data is saved and used to generate a recommendation to the user.

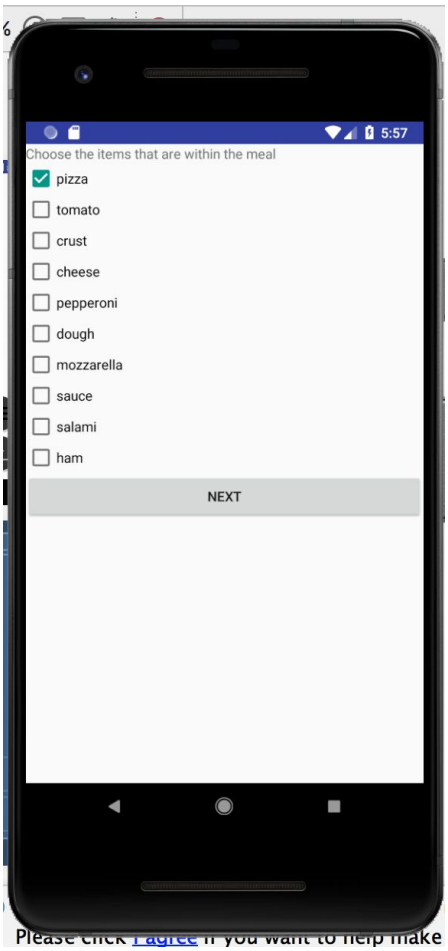
Search System



Step 1: Select upload button

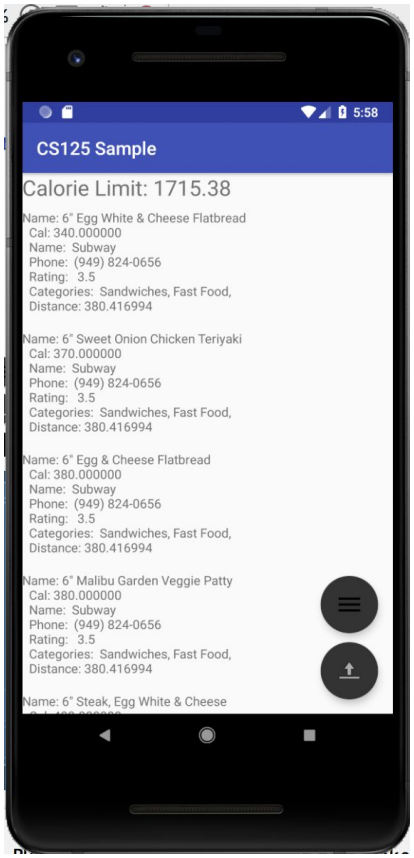


Step 2: Select Photo

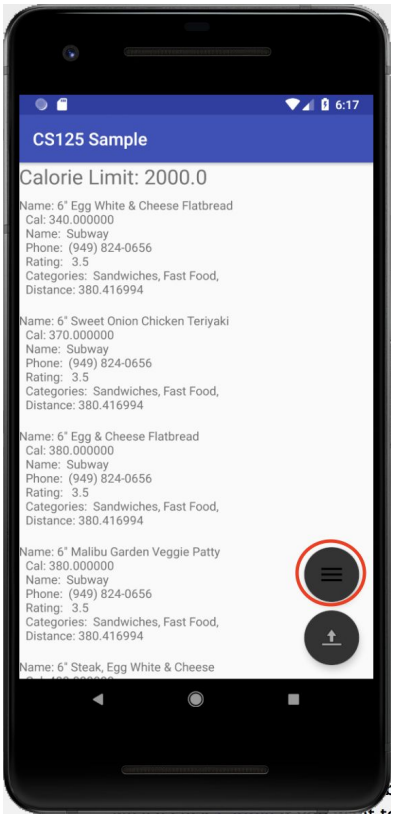


Step 3: Select Food Items

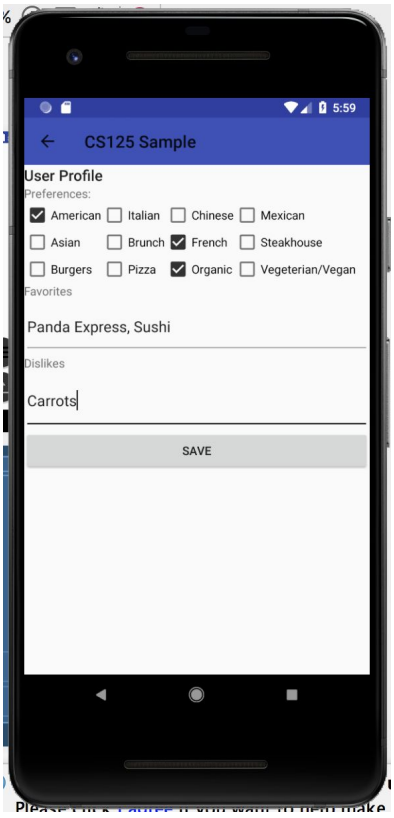
Profile System



Step 4: Updated Calorie Count



Step 1: Select Profile Menu Button



Step 2: Enter Preferences and Save

Presentation/Output

Most of the pertinent information for the user will be presented in the home page. Our application will present a newsfeed that presents the recommended food item and restaurant that serves that that particular food item. There is also an image upload button for easy access so the user can upload an image to the application. After uploading the image the user will be able to pick out search results as to what food items the application thinks is in the picture. Once the user picks from these results, the newsfeed will refresh with the updated total calories remaining

for the user for that particular day. (In the images above you can see the newsfeed and profile displayed.)

Details of Components and Algorithms

Components:

User Data Collection:

Though APIs offer many different solutions to otherwise complex problems, there is data that the system still needs that can only be collected from the users directly. The basics of this can be collected through the form of a user profile. Besides the obvious username and password to store and access specific user information, the system collects information about a user's gender, age, and weight in order to calculate the optimal amount of calories that can be eaten. Additionally, it will ask users to select their most liked and disliked foods which are used when calculating the best recommendations for the user.

Additionally, the system should learn about a user over time. The system has the ability to pull a user's GPS location and time stamp directly from the phone as well as ask for user feedback after each recommendation. Apart from being a necessary component for finding nearby restaurants, the location data will be saved in order to learn about a user's preferences when at that specific location over time(e.g. the user tends to go to In-N-Out when at a specific shopping center, so the system will then recommend In-N-Out more often whenever the user is at that location). The system will also be able to figure out common times that the user uses the application to find somewhere to eat by analyzing the time stamps when they open the app. When enough data is gathered, the application can begin sending a notification with a

recommendation attached at that specific time stamp. User feedback is used to further fine tune and improve upon the system as well as to learn a user's preferences apart from the liked and disliked food list attached to their profile.

Clarifai API:

Clarifai is a learning artificial intelligence program that is being trained to recognize the different contents of an image and video. For example, if the Windows XP Hills wallpaper is submitted to Clarifai, it would respond with landscape, hill, grass, and other aspects that it detects are contained as well as the probabilities that they appear in the image. It has models for general, faces, colors, travel, logos, apparel, and many other different image categories. Most importantly for our system, it also contains a model for food. As specified in the Clarifai API documentation, our system will provide the API with the key, the model being used, and the image submitted by the user. The API will respond with a list of all the predicted contents and their probabilities. The system then shows the first ten elements from this list to the user and asks them to select what they want factored into the meal calorie calculation. The items selected are then sent to Nutritionix to get their calorie information and then are added together to get a final meal calorie total.

Yelp API:

Yelp is a large and popular website that provides information and crowd-sourced reviews for businesses. It serves as the primary data source for all general restaurant information for the system. In particular, it provides the system with a restaurant's address, rating, phone number,

category, and if the restaurant is open. The system filters out restaurants that are closed and that are rated too low. It then utilizes the restaurant category in conjunction with a user's food preferences in the recommendations. This allows for the system to not only recommend food items that fit within their daily calorie limit, but also recommend food that the user will actually enjoy. All other information pulled from Yelp will be displayed in the newsfeed to provide the user with as much information as possible about a restaurant.

Fitbit API:

Fitbit provides resources for developers that help streamline the health aspects of the system. When creating an account, Fitbit asks users to provide information about their weight and weight goal which it then uses to calculate a daily calorie limit. Thus, instead of calculating that information internally, the system can instead pull it directly from Fitbit's database. Additionally, if a user is using a Fitbit accessory, the system will pull data about that user's daily activity and factor that into calorie calculation, allowing for a more accurate daily calorie limit and therefore more accurate food recommendations.

Nutritionix API:

The Nutritionix API allows for information about the nutrients in food to be searched with "natural language". For instance, it allows simple searches such as "1 cup mashed potatoes" to still provide approximate calorie counts without having to be too specific or search in an exact format. This is used in conjunction with the Clarifai API as despite Clarifai only providing the general categories of items in an image, Nutritionix still provides its best calorie estimate for

these categories thereby allowing for complete automation after the user selects which categories they want to be factored in to their calorie calculation.

Additionally, Nutritionix provides a branded food database. This database contains information on the calories of various menu items for over 600 restaurants. It is invaluable to the system as it is the primary source of data for finding different food items from nearby restaurants to recommend to the users.

Algorithms:

Getting API Responses (maybe not needed, not really an algorithm or component):

For every API call, the system makes use of Android Studio's Volley library. The method calls used to call Nutritionix and Yelp are based on the sample HttpRequest method provided in the sample project. The method calls also overrides GetParams and GetHeaders whenever an API requires the inclusion of parameters or headers for things such as API keys. The responses are often stored in a global string which other functions would access to parse the response.

Profile System:

Initially, the plan was for the user to be able to create an account and then set up their profile. This would include setting up a username and password, either manually entering user health information or logging in to Fitbit, and then an area where they would enter in their food preferences. However, the current system is only using one test profile and does not contain a robust implementation of account creation.

This test profile is currently saved in the form of a JSON file and is pulled and parsed by the system when the user visits the profile activity. This profile activity statically displays the test profile's name, gender, and height. The profile system currently saves the users preferences, favorites and dislikes intputed to an Arraylist for further use when the profile system is accessed. The user can update their information by inputting new information in the profile menu.

Getting Meal Calories from a Photo:

After the user successfully uploads an image, the system uses the same methods implemented in the sample project to send this image to Clarifai and then receive back a list of different concepts and their probabilities. After the system registers that this list is received, it selects the first ten items and dynamically creates and displays ten checkboxes to the user. After the user selects which items they want to be considered, the system sends each of these items to Nutritionix to get their calorie amounts. These calorie amounts are then added together to get a final total meal calorie total and then are sent back to the base activity so that it can be used in the recommendation algorithm.

Creating Recommendations:

Before the system can recommend food, it first needs to have a list of foods it can pull from to make these recommendations. In order to create this list, the system sends calls to both Nutritionix's restaurant database and Yelp to list nearby restaurants. The system saves the restaurant's brand id and name from Nutritionix to one list and relevant restaurant information from Yelp to another list. The restaurants in Yelp's list are then matched to those in Nutritionix's

list in order to create a final restaurant list to be used. After that, the system iterates over the restaurant list and, if the restaurant is open and has a rating above 2.5, sends that restaurant's brand id to Nutritionix's food database in order to receive the menu items from that restaurant. Finally, the system iterates over this list of menu items and, after filtering out food that is less than 250 calories and more than the user's calorie limit, adds them to a final master list of food from nearby restaurants.

Now that this list of food has been created, the system can now create recommendations. These recommendations are created through repeated sorting of the food list. First, the list is sorted by calorie amount in order to display the foods with the lowest amount of calories first. It is then sorted by distance in order to list food from the closest restaurants. Finally, it sorts the list by the user's food preferences in order to recommend food that the user will enjoy first and foremost. The end result is a list of food that is ordered in a way such that the closest preferred restaurants are displayed first with their menu items that fit within a user's calorie limit displayed from lowest to highest.

User Studies and Evaluation

Testing the System:

Many of the problems of the system boil down to the reliance on the Nutritionix database. The first is when using their search endpoint when searching for the amount of calories in a Clarifai concept. Though Nutritionix allows for broad searches that fit in with the broad concepts that Clarifai uses and therefore allow the most automation, this automation also resulted in more

inaccurate calorie calculation. Inevitably, a search for “rice” will be less accurate than a search for “two cups cooked white rice”.

Another issue comes when searching their database for branded foods. It has proved invaluable for providing the system with the menus of various restaurants and the calories of the items in those menus, but it does not classify what category these items are in. This did not seem like an issue at first, after all Yelp had categories for different restaurants, but it was found that unwanted items, such as drinks like Diet Coke or condiments like ketchup, always at the top of the recommendations as they inevitably have the fewest amount of calories. This is currently circumvented by avoiding business’ that sell mostly drinks such as Starbucks and only allowing the system to list items that are above 200 calories.

After testing, it was found that the Yelp JSON responses pulled from the search/business part of its API contain categories that are not the same as the ones currently used by the profile. For example, the app might say that the user enjoys American food, but the Yelp response only says if a restaurant serves sandwiches. This was unexpected and, in the time remaining, they could not be properly linked. Instead, the application currently uses hard coded preferences based off of Yelp’s responses in order to test the current recommendation algorithm.

The system was repeatedly tested throughout implementing its many features. As a result, the system runs as expected and outputs the correct responses.

User Testing:

Though the system is more of a prototype, we were able to find two user tester(s) to provide feedback on the current application. When shown the first screen of the app, one user

tester did not know what the upload button did until he tried it out and it was explained. The tester really liked the concept and felt that the calorie calculation was pretty close to the image. However, he did note that he would prefer some way to enter the actual amounts of different parts of food (e.g. “two cups of cooked rice” instead of just “rice”). For the recommendations, this tester claimed that the list sorted by calories then food preferences then distance was preferred over the others. He also suggested including a way that users themselves can select which sorting algorithm they want to use.

Another user found the UI to be a bit confusing. Before testing they could not predict what any of the buttons would do. After being explained the different components the user was able to upload a photo and found the calorie count feature helpful and useful for finding meals to eat. The user found the recommendations the sorting done by calories then distance then food preference the best for them because they do not have a car and prefers restaurants that are closer to them. Overall the user would prefer if they only had one dish recommended from each restaurant so they had a more diverse list to choose from.

Features Not Implemented

Learning About A User Over Time:

The original plan was for the system to constantly gather information about the user in order to learn about them and become a much more personalized application. This information included gathering which restaurants the user would frequent, what times that the user would use the app, and the user themselves rating the system’s recommendations. However, this plan was too ambitious and can not be completed within the time remaining.

Profile and Firebase:

Initially, the plan was for the user to be able to create an account and then set up their profile. This would include setting up a username and password, either manually entering user health information or logging in to Fitbit, and then an area where they would enter in their food preferences. However, the current system is only using one test profile and does not contain a robust implementation of account creation.

Ideally, the system will send and save data, most notably user profile information, to an online database. As recommended in lecture, Firebase seemed to be the best option available. However, though the application is connected to a Firebase database, it is currently not being used. Instead, the system reads and writes a user's profile to a locally saved JSON file. However, we understand that using an online database would be the most optimal method.

Fitbit:

An implementation of Fitbit API would have allowed the system to be able to constantly pull accurate health information about a user while also speeding up the process of creating a profile. However, getting this API working was much trickier than expected as it required use of the OAuth 2.0 authentication protocol and receiving a token before being able to pull specific user information. The app is instead simply using a static calorie daily limit of 2000 for use in the recommendations.

Conclusion

After receiving user feedback on our system we found there are many components of the project we can improve on. Due to the time constraint of this course the system is not as robust as we detailed in our first report and proposal. Though through researching current systems and user feedback we found more components that could better solve the problem proposed. The main improvement we would look into would be generating a better calorie count of food uploaded by the user. Without an accurate calorie count the system may not generate suitable dishes for the user to eat. Also better tracking of a user's health information would also be ideal to produce the best recommendations. A users heart rate, sugar levels and blood pressure are all health aspects that can affect a user health and diet. Overall more time and resources was needed to adequately solve the problem and implement the system we proposed.

References

1. Hay, G. J., and G. Castilla. "Object-based image analysis: strengths, weaknesses, opportunities and threats (SWOT)." *Proc. 1st Int. Conf. OBIA*. 2006.

2. Wang, Ruo-Qian, et al. "Hyper-resolution monitoring of urban flooding with social media and crowdsourcing data." *Computers & Geosciences* 111 (2018): 139-147.

3. Patel, Mitesh S., David A. Asch, and Kevin G. Volpp. "Wearable devices as facilitators, not drivers, of health behavior change." *Jama* 313.5 (2015): 459-460.

4. Kelman, Christopher W., A. John Bass, and C. D. J. Holman. "Research use of linked health data—a best practice protocol." *Australian and New Zealand journal of public health* 26.3 (2002): 251-255.

5. Jeffries, Robin, et al. "User interface evaluation in the real world: a comparison of four techniques." *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1991.

6. Nielsen, Jakob, and Rolf Molich. "Heuristic evaluation of user interfaces." *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1990.

7. Gandomi, Amir, and Murtaza Haider. "Beyond the hype: Big data concepts, methods, and analytics." *International Journal of Information Management* 35.2 (2015): 137-144.

8. Walia, Shelly, and Saptarishi Dutta. "How Zomato plans to eat Yelp's lunch in America." *Qaurtz*, 14 Jan. 2015, qz.com/325976/how-zomato-plans-to-eat-yelps-lunch-in-america/. Accessed 5 Feb. 2018.

9. Glueck, Jeff. "Introducing Foursquare Analytics: a dynamic foot traffic dashboard for brands." *Medium*, 20 Mar. 2017, medium.com/foursquare-direct/introducing-foursquare-analytics-a-dynamic-foot-traffic-dashboard-for-brands-9ce60a393b42. Accessed 15 Mar. 2018.

10. Hayne, Susie. "Chefs Feed Introduces its API; Launches Content Partnerships with Foursquare, Urbanspoon, Red Bull and Virgin America." *Nasdaq*, 12 Sept. 2014, globenewswire.com/news-release/ 2014/09/12/665757/10098433/en/Chefs-Feed-Introduces-its-API-Launches-Content-Partnerships-with-Foursquare-Urbanspoon-Red-Bull-and-Virgin-America.html. Accessed 15 Mar. 2018.