FridgeBuddy

VICTOR YE, University of California, Santa Cruz
PETER HUANG, University of California, Santa Cruz

ACM Reference format:
Victor Ye and Peter Huang. 2019. FridgeBuddy. *ACM Trans. Graph.* 37, 4, Article 111 (August 2019), 4 pages.
https://doi.org/10.1145/1122445.1122456

1 ABSTRACT

FridgeBuddy is a mobile application that helps the user manage their refrigerator and pantry inventory. Users are able to quickly scan the barcodes of their groceries into the app and keep track of what food items are stored in their refrigerator for later use. This report will cover our objective for the app as well as the details behind the development.

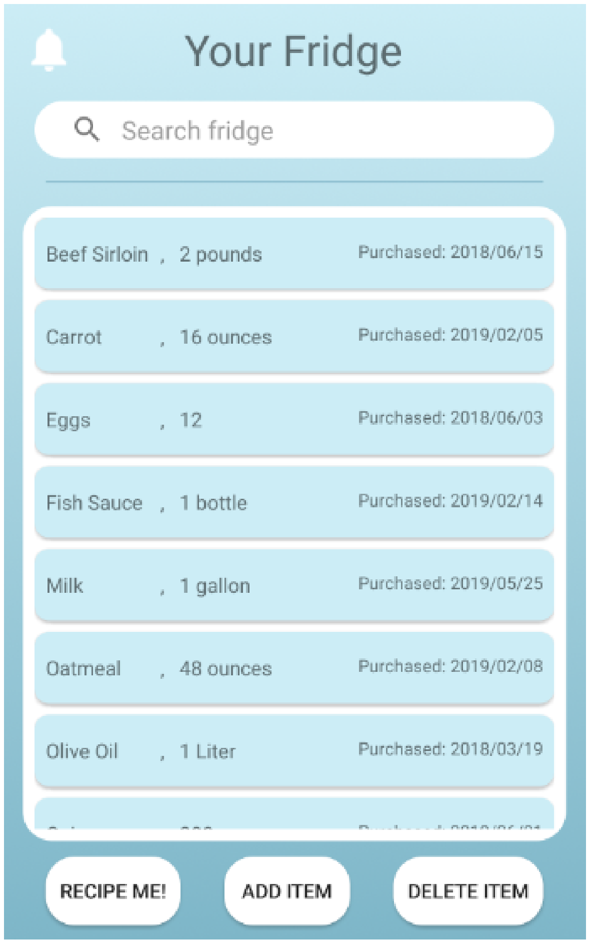
2 OBJECTIVE

The overarching purpose of FridgeBuddy is to provide users with an up-to-date look at what food they currently have available in their refrigerators, to better serve them when they go shopping for groceries. Using FridgeBuddy to log groceries would one, encourage users to create meals based on their currently stored food items, and two, prevent food waste by displaying what food items are near expiration.

Of course, similar applications exist in the form of shopping lists and whatnot. Users can manually search for and input items into a list and sometimes even share them with other users. However, the top apps on Google Play lack the ability to directly scan the barcodes of purchased items and save those items to the app. One app that did have this functionality is nearly ten years old and severely lacking in other capabilities. Due to the dearth of updated apps that had the desired functionality, Fridge Buddy was born.

3 COMPONENTS

FridgeBuddy contains all the basic components necessary for the digital storage and usage of food products. As the user opens the app, they are presented with the main screen *(Fig. 1)*. This screen's main feature is a list of the items stored in the user's fridge. Each item is displayed within a card and includes the item name, stored amount, and date of purchase. The item cards are automatically sorted alphabetically each time the list is updated, whether by adding or deleting items.



Beef Sirloin	, 2 pounds	Purchased: 2018/06/15
Carrot	, 16 ounces	Purchased: 2019/02/05
Eggs	, 12	Purchased: 2018/06/03
Fish Sauce	, 1 bottle	Purchased: 2019/02/14
Milk	, 1 gallon	Purchased: 2019/05/25
Oatmeal	, 48 ounces	Purchased: 2019/02/08
Olive Oil	, 1 Liter	Purchased: 2018/03/19

Fig. 1. FridgeBuddy Main Screen. This is the core of FridgeBuddy, where the user can view, modify, or generate a recipe with their food selection. UI design by Peter Huang.

3.1 Adding and Deleting Items

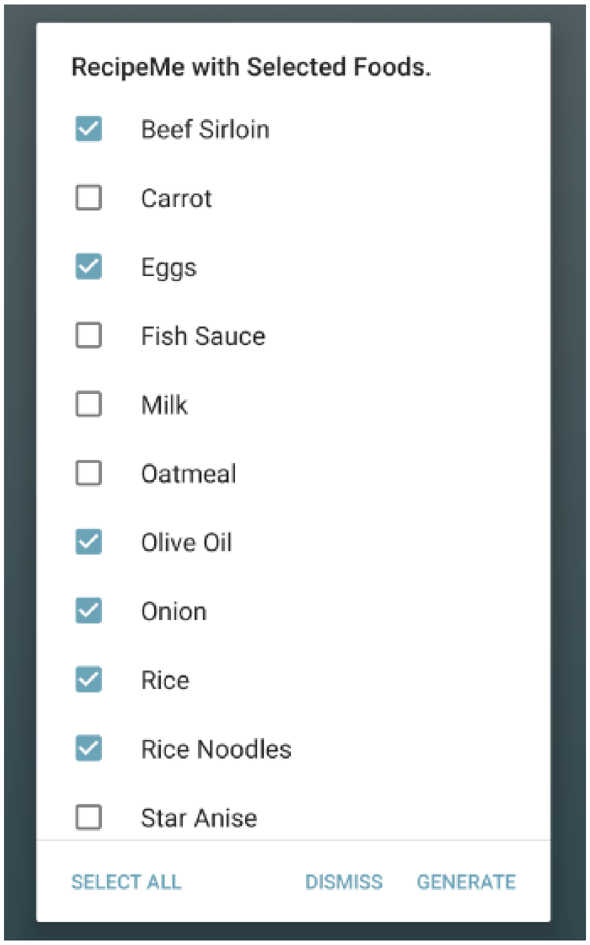
The user is able to add items to this list with the "`Add Item`" button. After clicking the button, the user is directed to another screen (AddFood) that gives them the option to either manually input items, or for food items with barcodes, scan them into the list. After entering the food name as well as the amount into the provided spaces, the user can click the "`Add Item`" button to insert the item into the database. Upon the successful storage of said item, the user is shown a Toast to confirm the updated contents of the fridge.

In addition to manually inserting an item into the fridge, users can click on "Scan Food" to bring up a new screen (ScanCode). Here, the user's front camera is accessed to scan product barcodes. On scanning a barcode, the affiliated item is added to the database and the scanner pauses for 1 second to prevent double scanning. When the user returns to the main screen via their phone's back button, the list is refreshed and displays the newly inserted item.

Of course after consuming food items, users must also be able to delete items from their fridge. Upon clicking the "`Delete`" button, a pop-up appears that allows users to select the items individually or altogether to be deleted from the database.

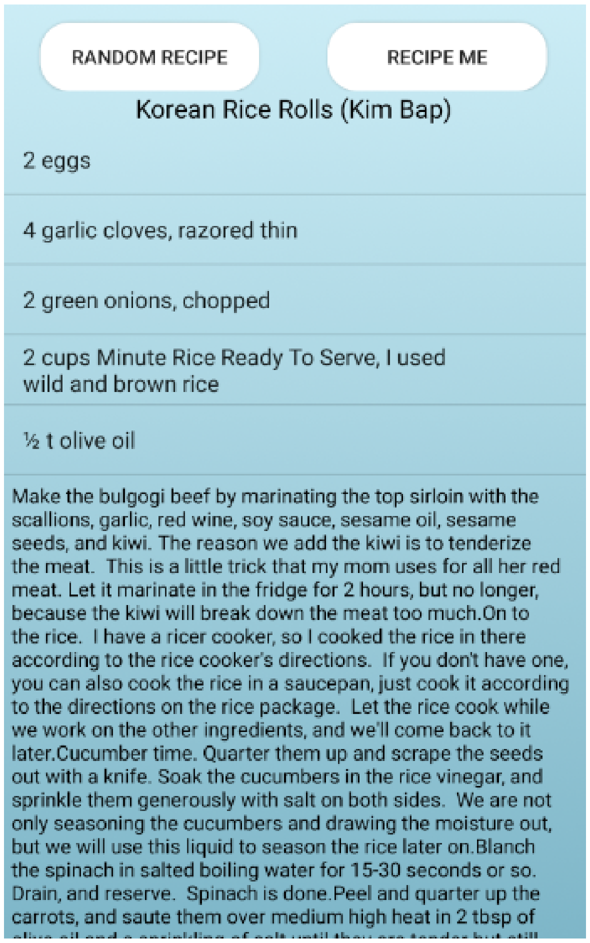
3.2 Generating Recipes

The "`Recipe Me!`" button allows the user to look up recipes from the items stored in the fridge. Users are taken to a new screen (RecipeMe) on clicking the button and shown two new buttons, "`Random Recipe`" and "`Recipe Me`" *(Fig. 3)*. "`Random Recipe`" does just what a user would expect; it displays a random recipe in the area below the buttons, generated by our recipe and food product API, Spoonacular. "`Recipe Me`" on the other hand allows users to select items from their current fridge inventory and generate a meal recipe from those selected ingredients(Fig. 2).


RecipeMe with Selected Foods.

- [x] Beef Sirloin
- [ ] Carrot
- [x] Eggs
- [ ] Fish Sauce
- [ ] Milk
- [ ] Oatmeal
- [x] Olive Oil
- [x] Onion
- [x] Rice
- [x] Rice Noodles
- [ ] Star Anise

Fig. 2. FridgeBuddy RecipeMe Food Selection. The user is prompted to select which food items they want to generate a recipe with.



Korean Rice Rolls (Kim Bap)

2 eggs

4 garlic cloves, razored thin

2 green onions, chopped

2 cups Minute Rice Ready To Serve, I used wild and brown rice

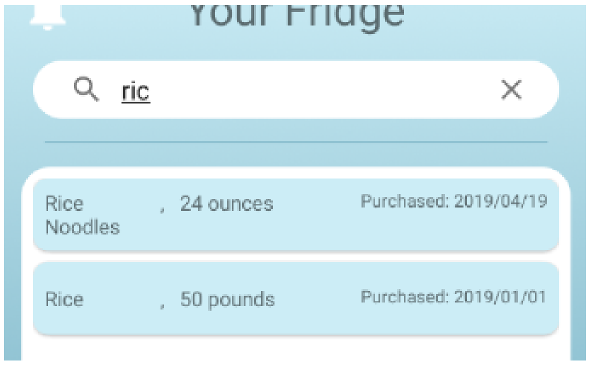
½ t olive oil

Make the bulgogi beef by marinating the top sirloin with the scallions, garlic, red wine, soy sauce, sesame oil, sesame seeds, and kiwi. The reason we add the kiwi is to tenderize the meat. This is a little trick that my mom uses for all her red meat. Let it marinate in the fridge for 2 hours, but no longer, because the kiwi will break down the meat too much.On to the rice. I have a ricer cooker, so I cooked the rice in there according to the rice cooker's directions. If you don't have one, you can also cook the rice in a saucepan, just cook it according to the directions on the rice package. Let the rice cook while we work on the other ingredients, and we'll come back to it later.Cucumber time. Quarter them up and scrape the seeds out with a knife. Soak the cucumbers in the rice vinegar, and sprinkle them generously with salt on both sides. We are not only seasoning the cucumbers and drawing the moisture out, but we will use this liquid to season the rice later on.Blanch the spinach in salted boiling water for 15-30 seconds or so. Drain, and reserve. Spinach is done.Peel and quarter up the carrots, and saute them over medium high heat in 2 tbsp of

Fig. 3. FridgeBuddy Generated Recipe. The app returns a best suited recipe based off of the food items that were selected.

3.3 Search

Back on the main screen, the search bar allows the user to filter the fridge to display items with matching strings in real time. (Fig. 4)

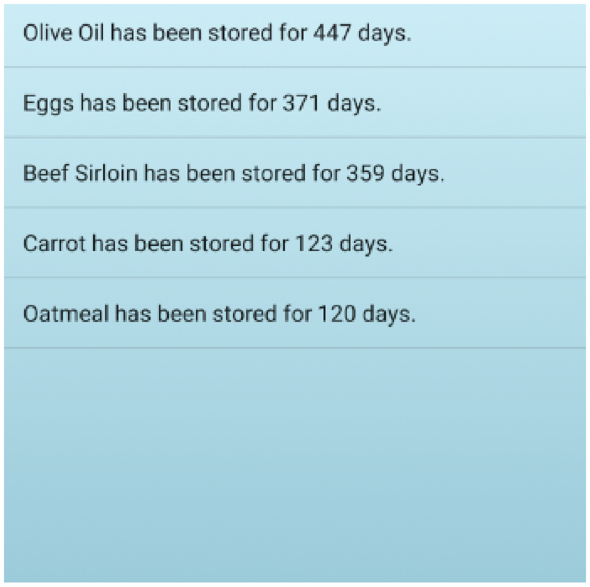


Rice Noodles	, 24 ounces	Purchased: 2019/04/19
Rice	, 50 pounds	Purchased: 2019/01/01

Fig. 4. FridgeBuddy Storage Search. The user can filter their storage for specific food items.

3.4 Notifications

Finally the bell button takes the user to a Notifications screen, which displays the five oldest items stored in the fridge *(Fig. 5)*.



Olive Oil has been stored for 447 days.

Eggs has been stored for 371 days.

Beef Sirloin has been stored for 359 days.

Carrot has been stored for 123 days.

Oatmeal has been stored for 120 days.

Fig. 5. FridgeBuddy Notifications. The app returns the top five oldest food items from the fridge.

4 DEVELOPMENT

In the first stages of FridgeBuddy, we started with a barebone activity with three buttons: Storage, RecipeMe, and Notifications. These buttons led to their respective activities.

4.1 Storage

We started by implementing an addFood functionality, consisting of two editTexts for the user to input the food name and amount, which, along with the current date, would then be saved into the local storage and displayed in a listView. However, we quickly realized that for sake of convenience, not everyone had the leisure to manually input every single food item. Since FridgeBuddy is centered around the idea of convenient tracking, we wanted create a super simple, user friendly experience. As such, we also implemented a barcode scanner using the ZXing scanner library, which accesses a device's front camera and reads the 12 digit UPC (Universal Product Code) code from the target product *(Fig. 6)*. We integrated the scanner into our app but did not have time to implement the actual product data acquisition before the interim presentation.

4.2 RecipeMe

Initially, the recipeMe button generated a recipe from every food item in storage. However, we realized that this actually limited people in their food choices as the API's random recipe generation was not truly random and repeating recipes. To handle this, we changed the recipeMe button to first create an alert dialog to prompt the user to select foods from their storage, before generating the recipe, which gave the user more control over their food choices. In addition, to limit the repetitiveness of the recipes, recipeMe generates 5 different random recipes and then returns a random recipe of the 5.

Fig. 6. Example UPC code. Using our scanner to scan a food item 's barcode, we are able to store it's name and amount.

4.3 Notifications

In the planning of our application, we had believed there to exist an API for food expiration life, which we planned to lookup and store into our database. However, there was none such API, so we had to settle with a simple listview presenting the top 5 oldest foods in storage.

4.4 Midquarter Updates

We finished these three core components and presented them at the interim presentation, along with some mockups we had created for a UI overhaul. In preparing for the Future Work section of our presentation, we mapped out the features the two of us could viably complete by the end of the quarter (in three weeks). We settled on completing the barcode scanner feature as that was the core of our app, as well as revamping the UI to enhance ease of use for the user. Extra features like a search filter and notifications were added as stretch goals, as we did not know how difficult meshing the scanner and food product API would be.

After the interim presentation we set to work on completing the scanner functionality, allowing it to read UPC codes, send a request to the NutritionX API, and finally input the parsed JSONObject to our database. After several iterations in which we switched APIs from Spoonacular to NutritionX, we completed the full scanner functionality with some time to spare for other features.

4.5 Preparing for the Final Presentation

While work progressed on the scanner, we reworked the UI to make it more intuitive for the user and easier on the eye. Changes included moving all the core components such as the food list to the MainActivity, with buttons for Add Food and RecipeMe leading to their respective activities. The initial ListView we used to display items from the database was changed to a RecyclerView that held separate CardViews which encapsulated the individual food items. We did this initially to plan ahead for the inevitable Delete Items function, where we would attach a delete button to each card view. However, we later scrapped this idea and opted for a less finicky option by allowing the user to select items from an alert dialog.

Finally, we wanted users to be able to filter specific items from their entire fridge inventory. We added in a SearchView to do just that, allowing users to filter their inventory by name and update the RecyclerView in real time.

Looking back, the proposal's scope was much too large for a single quarter of work between two people. We never got around to implementing custom recipes or linking our app with Amazon for easy ingredient ordering. These are features that we'll go over in more detail in the Future Work section. However, with all things said and done, we developed a working and effective application that satisfies the core of our proposal to provide users with an easy-to-use digital fridge.

5 CONTRIBUTION

Over the course of FridgeBuddy's development, the two of us worked on every component to some degree. Overall, Victor built up most of the components involving the database and API integration while Peter fleshed out the UI and activity flow from screen to screen.

After Peter enrolled in the class Week 4, the proposal had already been written by Victor and submitted. In the proposal, Victor had detailed the core components of FridgeBuddy so we met up to lay out the basic UI and flow of the app.

In the weeks prior to the interim presentation, we set out to create the basic functionalities of the app and worry about the UI later. Victor set up the MainActivity to redirect users to three separate activities, addFood, recipeMe and Notifications, which he created. At the time, addFood only allowed user to manually input food items and recipeMe provided the user with a random recipe each time. Victor also set up the initial database and related methods in MyDB. Peter modified MyDB to use a singleton design pattern along with several other enhancements to smooth development out. Peter also implemented the barcode scanner inside of the ScanCode activity and created mockups for the logo and new UI moving forward.

To prepare for the interim presentation, we met up to narrow the current content we had for FridgeBuddy down into a few easily digestible slides. We wrote a short script for the 5 minute limit with which we could direct the audience through the overview, initial mocks, demo, and what we planned to have done by the final presentation.

After the interim presentation, Victor pulled in the NutritionX API and linked the barcode scanner to call it upon user query. Now, users could simply scan the barcodes of food items to add them to our database. He also updated Notifications to display the oldest items stored in the database by referencing their purchase date. Peter completely overhauled the UI, moving the fridge inventory over from addFood into the MainActivity and changing the old ListView into the current RecyclerView. He also introduced the FoodListAdapter, which encapsulates each food item from the database inside a CardView, which is then displayed inside the MainActivity's RecyclerView. Finally, Peter introduced the SearchView that allows user to filter specific items within the RecyclerView. We both worked on every activity and class we introduced to the project, but for the most part, the contributions listed above were our main focuses.

For the final presentation, we wrote a short script together and focused the bulk of our time on demonstrating the main features of FridgeBuddy like adding items and querying recipes, while leaving a minute to discuss the smaller features such as search and notifications.

While preparing for our final presentation, both of us sat down to review the course of development and the features included in our final iteration of FridgeBuddy. This timeline we created easily translated to the component and development sections of this report. Both of us wrote about the features we developed for FridgeBuddy in this report and then each proofread it.

6 CHALLENGES

Prior to the interim presentation, there were actually three members of FridgeBuddy. However, one member dropped the class, opening up space for Peter to join the team. However, the other member withdrew shortly before the interim presentation without any prior notice and had not contributed any meaningful work so FridgeBuddy's original development schedule lay in question. The remaining two of us scrapped some planned features due to time constraints and focused on the core elements of the app which we completed. However, the lack of a third contributing member stunted the speed and scope of development.

7 FUTURE WORK

During the development of FridgeBuddy, we had ideas for several features that we ended up scrapping for lack of time and devs.

First and foremost, we wanted users to be able to further customize the recipes based on their fridge inventory. Currently, the RecipeMe component returns a recipe that contains the user's selected ingredients using the Spoonacular API we implemented. We hoped to include more customization by adding tags like “vegetarian” or “high protein” to grab more personalized recipes for the user. This would be definitely be priority for future development.

Another functionality we wanted to include was accurate representation of food expiration dates. We could possibly implement this with web scraping, as there are many websites already containing this information.