SnackTrack

Thomas Borgia, Kylie Gorman, Brandon Gottlob, Nate Milkosky

Definition of Problem

- There is not a convenient, easy to access way to store information about food items in a house, dorm, or other area of living
- People can forget what food they have, and it may expire and end up being wasted

Why is it a problem?

- Food can easily go to waste or be forgotten
- It can be difficult to keep track of when you will need to buy more food
- It can also be difficult to decide how much food to buy

What solution exists today?

- There are apps that tracks the foods you have eaten
  - Most of these are nutrition apps and log intake of calories and other nutrients
- Example: Calorie Counter by MyNetDiary

How is your solution better?

- SnackTrack will show users what they CAN eat in addition to what they have eaten
- SnackTrack will emphasize efficient use of snacks rather than nutrition
- The information stored in SnackTrack can help the user save money when buying food and stop food from expiring

System Characteristics: Main Features

- Scan food with labels to keep track of food that is ready to eat
- View current catalogued food in a snack list
- A section to input food that does not have a UPC code or is not recognized by the database
- Notify the user when food is expiring as the application is opened
- Allow the user to remove items when they have consumed them
- Saving the food catalogue locally on a users device.

System Characteristics: Additional Features

- Help screen to explain functionality
- Automatic scrolling of a food list
- Click anywhere on screen to cancel keypads
- Remove item by selecting it in the snack list or scanning it
- Specify the number of items the user wants to delete

Implementation - Resources

- Because we wanted to develop for iOS, we were required to use Objective-C as the programming language, and Xcode as our IDE, which let us create the user interface as well as all of the code in the background.
- We also took advantage of some other resources, such as the ZBar SDK which easily allowed us to implement barcode scanning. Another resource was Outpan.com, an online database of items and barcodes that we used to get information about an item from just a barcode. In addition to accessing item information, it permitted the easy addition of items to the database.

Implementation - Components

There are three major components for the SnackTrack:

1. User Interface Code – Implemented in ViewController classes
2. Database Interaction – Implemented in UPCParser class
3. Data structures to store food information – Implemented in FoodList and FoodItem classes

Implementation - User Interface

- Responds to user actions such as button taps and swipe gestures
- Each action has its own method that makes calls to back-end components that manipulate data
- Populates the user interface elements with data from FoodList and FoodItem objects

Implementation - Database Interaction

- All database interaction code is contained in the UPCParser class
- Fetches and parses JSON data from the database to instantiate a new FoodItem object given its UPC code
- Creates HTTP post requests with data formatted in Outpan's Product Markup Language to add new data to the database


```
{
    barcode: "034000004409",
    outpan_url: "http://www.outpan.com/view_product.php?
    barcode=034000004409",
    name: "2 Reese's Peanut Butter Cups",
    description: "",
  - attributes: {
        Manufacturer: "Reese's",
        Ingredients: "Milk Chocolate (Sugar; Cocoa Butter;
        Chocolate; Nonfat Milk; Milk Fat; Lactose; and Soy
        Lecithin and PGPR, Emulsifiers); Peanuts; Sugar;
        Dextrose; Salt; and TBHQ (Preservative)."
    },
  - images: [
        "http://www.outpan.com/photos/ez36j6frgpicr39cz0.jpg"
    ],
  - references: [
        "http://grocery.peapod.com/pd/Reeses/Milk-
        Chocolate-Peanut-Butter-Cups/1-50-oz/034000004409/"
    ]
}
```



```
<name>2 Reese's Peanut Butter Cups</name>
<description></description>
<attribute name="Manufacturer">Reese's</attribute>
<attribute name="Ingredients">Milk Chocolate (Sugar; Cocoa Butter;
Chocolate; Nonfat Milk; Milk Fat; Lactose; and Soy Lecithin and PGPR,
Emulsifiers); Peanuts; Sugar; Dextrose; Salt; and TBHQ (Preservative).
</attribute>
<image filename="ez36j6frgpicr39cz0.jpg" />
<reference url="http://grocery.peapod.com/pd/Reeses/Milk-Chocolate-
Peanut-Butter-Cups/1-50-oz/034000004409/" />
```

Implementation - Data Storage/Manipulation

- FoodItem objects store information about a specific food item, including:
  - Name
  - Quantity
  - Expiration Date
  - Description
  - UPC Code
- The FoodItem class implements methods to compare two FoodItem objects for equality – used in search functions
- The FoodList class stores an array of FoodItems
- FoodList implements search function to find a specific FoodItem object
- FoodList implements special add/remove functions to handle quantity of items – uses search function to increment/decrement quantity of correct item

Evaluation (Based on the Eight Golden Rules)

1. Strive For Consistency

- The application is consistent in respect to colors, fonts, vocabulary and commands.

2. Cater to Universal Usability

- The size of font is reasonable for those with bad vision. The buttons are large enough to press and find on the screen.

3. Offer Informative Feedback

- When a user selects a button, the color changes to inform the user that the button was pressed.

4. Design Dialogs to Yield Closure

- The corresponding buttons guides the user to complete the desired action.

Evaluation (Based on the Eight Golden Rules)

5. Prevent Errors

- To delete a certain number of items,the user is given a number pad to avoid incorrect entries.
- If the number the user inputs is larger than the current quantity, a notice that this cannot be done appears.
- If the user attempts to delete by scanning and the food item is not found, an alert appears that this cannot be done.

6. Permit Easy Reversal of Actions

- If the user accidentally selects the wrong page, there are cancel and back buttons at the top of each page.

7. Support Internal Locus of Control

- The user has complete control over their food inventory through adding, editing and deleting.
- Also we allow the user to add items not found into an online database for future item identification.

8. Reduce Short-Term Memory Load

- The user can simply scan a food item to add or delete to avoid the need for an extensive amount of info about the item from the user.

Evaluation (Based on User Surveys)

- When we started this project, we knew that it would not be useful for everyone.
- We predicted that the application would appeal mostly to those who manage or keep track of the food.
- Based on our experience from the poster presentations, we have reached our target audience.
- We experienced first hand that those who manage their food in college found it useful. Another trend was that more adult female users favored the application than any other group.
- The overall satisfaction of the application based on 34 users was 88%.

Final Thoughts

- Our team plans on future development and the releasing of SnackTrack to the public.

- After hearing constructive feedback from our respondents there are several features that we would like to improve and implement.

- Some of these improvements include:
  1. Generating a shopping list for the user based on the list
  2. Nutrition info to help users keep track of their nutritional health.
  3. Undo ability in case of accidental deletion
  4. Make form labels more visible to the user
  5. Improve help screen
  6. Change home page

Questions?