NUTRITION BUDDY: ENABLING BEHAVIOR CHANGE THROUGH WEARABLES, RECOMMENDATIONS, AND DIGITAL AVATARS

Dartmouth Computer Science Technical Report TR2016-808

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts

in

Computer Science

by

Daniel Chen

DARTMOUTH COLLEGE

Hanover, New Hampshire

May 31, 2016

Examing Committee:

Tim Tregubov

Xia Zhou

Lorie Loeb
Abstract

In recent years, significant progress has been made in health technology through advancements in mobile technology, wearable computing, and connected devices. These advancements have created lightweight or even automated experiences for users in vital aspects of health such as exercise and sleep tracking. Despite these advancements, one area of health, nutrition, remains stifled by cumbersome user interaction. We present Nutrition Buddy, a novel approach to nutrition tracking which utilizes a simplified data model that allows for lightweight interactions, effective wearable technology, and relevant recommendations provided by a digital nutritionist “buddy” to facilitate behavior change.
Preface

I would like to thank first and foremost my committee, Tim Tregubov, Xia Zhou, and Lorie Loeb for not only serving as advisors for this project but also throughout my time at Dartmouth. Additionally, I would like to thank my friends and family for supporting me, especially those that served as beta testers. A special acknowledgment goes to my sister, Amy Chen, who supplied multiple rounds of artwork for not only Nutrition Buddy but the prototype Pebble watch application Food Baby.
Contents

Abstract ................................................................. ii
Preface ................................................................. iii

1 What is Nutrition Buddy? ............................................ 1
  1.1 The Problem with Current NTAs .............................. 1
    1.1.1 Heavy User Interaction .................................. 2
    1.1.2 Insufficient and Inaccurate Data ..................... 3
    1.1.3 Lack of Motivators ........................................ 3
  1.2 Nutrition Buddy’s Solution .................................... 3
    1.2.1 Simplified Data Model ................................... 4
    1.2.2 Recommendations .......................................... 5
    1.2.3 Wearable Computing ....................................... 5
    1.2.4 The Tamagotchi Effect .................................... 6

2 Digital Avatars for Emotional Attachment ...................... 7
  2.1 Previous Applications for Behavior Change ............... 8
    2.1.1 Greenlite Project ........................................ 8
    2.1.2 Plant Nanny ................................................. 8
    2.1.3 Food Baby Prototype ...................................... 9
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Starting Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Implementation</td>
<td>34</td>
</tr>
<tr>
<td>7.1</td>
<td>Top Level View</td>
<td>34</td>
</tr>
<tr>
<td>7.2</td>
<td>Cloud Service</td>
<td>35</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Technologies</td>
<td>35</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Data Model</td>
<td>36</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Resources</td>
<td>38</td>
</tr>
<tr>
<td>7.3</td>
<td>iOS Application</td>
<td>39</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Frameworks</td>
<td>40</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Storing Application State and Data</td>
<td>41</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Buddy Animation System</td>
<td>42</td>
</tr>
<tr>
<td>8</td>
<td>Results</td>
<td>46</td>
</tr>
<tr>
<td>8.1</td>
<td>Methodology: Beta Testing and Survey</td>
<td>47</td>
</tr>
<tr>
<td>8.2</td>
<td>Overall Performance</td>
<td>48</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Comparison to other NTAs</td>
<td>49</td>
</tr>
<tr>
<td>8.3</td>
<td>Recommendations Evaluation</td>
<td>51</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Historical Weighted vs. Naive Algorithm</td>
<td>52</td>
</tr>
<tr>
<td>8.4</td>
<td>Buddy Evaluation</td>
<td>53</td>
</tr>
<tr>
<td>8.5</td>
<td>Qualitative Feedback</td>
<td>54</td>
</tr>
<tr>
<td>9</td>
<td>Future Work</td>
<td>55</td>
</tr>
<tr>
<td>9.1</td>
<td>Units</td>
<td>55</td>
</tr>
<tr>
<td>9.2</td>
<td>Buddy Improvements</td>
<td>56</td>
</tr>
<tr>
<td>9.3</td>
<td>Launch and Further Testing</td>
<td>57</td>
</tr>
<tr>
<td>10</td>
<td>Conclusion</td>
<td>58</td>
</tr>
</tbody>
</table>
Chapter 1

What is Nutrition Buddy?

This chapter seeks to provide an overview of Nutrition Buddy by first examining the state and problems of mobile nutrition tracking applications (NTAs) today and then taking a broad look at the paradigms of user experience that drive and motivate Nutrition Buddy.

1.1  The Problem with Current NTAs

Nutrition-tracking applications (NTAs) today lack the ease-of-use and lightweight interactions that characterize modern mobile applications. Whereas much has been done in the way of tracking other health metrics, such as exercise, sleep, heart rate, weight, etc., through automated or lightweight methods, nutrition tracking is a problem that is left with ineffective solutions.

We have identified three primary characteristics of NTAs which inhibit adoption and create a bad user experience, they are:

- Heavy User Interaction
1.1 The Problem with Current NTAs

- Insufficient and Inaccurate Data
- Lack of Motivators

The combination of these three traits have created such a bad user experience that only very committed users (those on a strict diet or trying to reach some sort of nutritional goal) will use NTAs on a regular basis, as they have to overcome these three factors. That leaves the remaining majority of the population without an effective option to log, view, and improve their nutritional intake.

The criticisms of NTAs in this section are derived from looking at “MyFitnessPal” [11] and “Lose It!” [5], two of the most popular NTAs for iOS. The next couple sections will go over each of the characteristics of current NTAs in detail.

1.1.1 Heavy User Interaction

Significant progress has been made in health technology in recent years through a combination of advancements in mobile technology, wearable computing, and connected devices (Internet of Things). In industry, companies such as Apple (HealthKit, accelerometer advancements, GPS, Apple Watch) and Fitbit (exercise tracking, connected scales) are paving the way for health tracking which is either completely passive or require very little user interaction.

Despite these advances in other areas of health, nutrition has yet to make such a leap. One unique difficulty in this area is that, with the state of technology today, nutrition tracking is a problem which requires significantly more user involvement. The most popular NTAs today require users to input exact quantities and types of foods that they consume – a cumbersome process that does not lend itself well to automation and wearable technology.
1.2 Nutrition Buddy’s Solution

1.1.2 Insufficient and Inaccurate Data

Another factor that makes the nutrition tracking problem intractable is the massive varieties of foods, cuisines and options available. Although many apps expedite the food logging process through storing common store-bought or chain-restaurant foods in massive, crowd-sourced databases, user interaction becomes difficult as soon as users need to input uncommon foods in uncommon quantities (such as those from a restaurant, of the user’s own creation, or those originating from a country with few users). Also, there is ambiguities as nutritional value varies greatly among the same foods with different ingredients or styles of preparation.

1.1.3 Lack of Motivators

NTAs do not effectively motivate users to revisit the application and foster long-term interaction. They provide tools for analytics such as calorie counts over time, nutrients consumed, and streaks. Aside from these forms of feedback, these apps usually do not attempt to engage the user with more than the occasional notification or digital trophy.

1.2 Nutrition Buddy’s Solution

In the previous section, three main criticisms of NTAs were discussed. The goal of Nutrition Buddy is to produce an NTA which make compromises in some areas to address the problems of incumbent NTAs.

Our solution, Nutrition Buddy, is an NTA for iOS and Apple Watch that employs lightweight user interactions, a simplified data model, wearable technology, and a
1.2 Nutrition Buddy’s Solution

digital avatar to provide an engaging user experience. This experience would provide motivation beyond the individual user’s personal health goals to keep them on track. The lightweight interactions will allow for engagement of a larger audience than current NTAs, and recommendations will provide a pervasive experience not previously seen in nutrition tracking.

1.2.1 Simplified Data Model

Nutrition Buddy uses a simplified data model based off food groups rather than individual foods and nutrients. By doing this, we are able to solve the problem of insufficient and inaccurate data that is characteristic of current NTAs in a manner which is consistent with government recommendations [1]. Additionally, the usage of a simplified data model allows for quicker user interactions – most notably a compelling wearable interface through the Apple Watch. Even without the Apple Watch, the interaction is still significantly lighter as users only have to select from a handful of food groups rather than concern themselves with exact foods and quantities.

The focus on food groups rather than individual nutrients or food items is backed by government recommendations on healthy eating. Although outdated, users may know the USDA Food Pyramid and Food Plate [14], which simplify nutrition recommendations into food groups (we note that they are outdated based on the allocation of food groups, not the focus on food groups itself). More recently, the “Dietary Guidelines for Americans 2015-2020” [1] also implement a food-group based view on nutrition which Nutrition Buddy uses for its default diet plans.
1.2 Nutrition Buddy’s Solution

1.2.2 Recommendations

The use of a simplified data model allows us to deliver recommendations, something that is not typically found in incumbent NTAs. Since users define a diet goal and log their eating within that framework, a recommendation system can be used to show a user where he or she needs to focus to meet his or her goals.

With recommendations as part of the system, Nutrition Buddy provides an experience which goes beyond simple logging and comes closer to the pervasive experience provided by health applications such as exercise tracking, where notifications and recommendations are delivered throughout the course of a day based on activity. Additionally, it evolves nutrition technology from a diet journal to a digital nutritionist.

In a later section, we discuss two different recommendation algorithms: Naive and Historically Weighted, which differ in their consideration of a user’s long-term eating habits.

1.2.3 Wearable Computing

In addition to effective recommendations, our simplified reporting interaction allows for usage of wearable computing. Combined with recommendations, wearable computing provides advantages by allowing for a lightweight logging experience through the Watch App and a pervasive recommendation experience through watchface indicators (Complications) and Glances.

Since the data has been simplified into five food group categories, the logging process is suited for wearables as everything fits on a single screen. Using the Nutrition Buddy Watch App, users are able to log meals with simple taps, analogous to how one might navigate the watch interface to begin tracking for a workout.
1.2 Nutrition Buddy’s Solution

In addition to the Watch App, Nutrition Buddy supports Apple Watch Complications and Glances. Complications are small visual elements that appear on watch-faces, and Glances are information summaries which users can find through a simple swipe up on the watch screen [2]. Nutrition Buddy uses complications as one of the avenues for providing recommendations for what users should eat everytime they check their watch. Nutrition Buddy uses glances to provide both recommendations and a summary of their current eating for the day.

1.2.4 The Tamagotchi Effect

In addition to the use of recommendations and wearable computing to address the “Lack of Motivators” problem of current NTAs, Nutrition Buddy uses a digital avatar to emotionally engage users to adopt healthier eating habits. The “buddy,” a digital agent similar to a digital pet which the user develops a connection to, is both a visual manifestation of the user’s dietary performance and a digital nutritionist. The buddy’s animations range from sad to happy based on the user’s nutrition, and the buddy’s recommendations steer the user along the correct path.

The basis of the usage of this avatar is the Tamagotchi effect [9], the attachment of users to digital agents. As described in the next chapter, this effect is a powerful tool to incite behavior change in a variety of applications, and Nutrition Buddy provides a new application in this field.
Chapter 2

Digital Avatars for Emotional Attachment

The usage of a digital avatar in Nutrition Buddy is based off of many examples of popular digital pet games which have successfully engaged users using the Tamagotchi effect, which is the development of emotional attachment with software agents and/or robots [9]. This effect most notably has applications in addictive mobile games (which may or may not benefit the user), but also has been used for behavior change both in industry and in academia.

One reason that gamification and using the Tamagotchi effect is especially relevant is that benefits from healthy nutrition habits take place on a long time scale. Social psychology research has shown that humans tend to be emotionally disconnected from their future selves, which helps explain poor long-term behavior [3]. By using the Tamagotchi effect with a digital pet that responds immediately to the user’s dietary choices, Nutrition Buddy provides short-term motivation for healthy eating habits where there previously was very little.
2.1 Previous Applications for Behavior Change

While many would think of the Tamagotchi effect for its usage in games such as “Tamagotchi” or the recent “Neko Atsume: Kitty Collector” (a popular mobile game which involves feeding and taking pictures of cats [6]), the Tamagotchi effect is used outside of games for a variety of different kinds of behavior change, including social, educational, health, environmental, and even gardening [9][13][7][10][8]. This section surveys some of the previous work done in this area.

2.1.1 Greenlite Project

The Greenlite project used an animated polar bear in various states of happiness or distress to reflect the energy consumption of a building. Using this application of digital avatar animation the researchers were able to reduce energy consumption and encourage people to adopt more energy efficient habits [10].

2.1.2 Plant Nanny

Plant Nanny is a popular mobile application in which users take care of a plant by logging their intake of water, which in turn waters the plant. The application won Apple’s App Store Best of 2013 [7].
2.1 Previous Applications for Behavior Change

2.1.3 Food Baby Prototype

“Food Baby” is a Pebble Smartwatch application (no companion mobile application) created by the same author of this paper. While it does address the issues of current NTAs and is in many ways similar to Nutrition Buddy, it is far from the final product and should be considered a prototype. Using just a few button clicks, users log their food-intake and the buddy (an anthropomorphic food pyramid) responds by getting happier as the user has a more balanced diet. The application uses a naive algorithm to provide recommendations.

The application has had traction in the modest Pebble user base with nearly 800 installs and 37 hearts [4]. It also addresses the problems of current NTAs: Firstly, the interaction is extremely lightweight in comparison to other NTAs. This is done through the smartwatch platform and a reduction in the number of actions to log food. Secondly, the simplified data model facilitates the lightweight interactions. Instead of trying to cover millions of food items, it reduces nutrition down to 6 food groups (including water), so users only have to navigate to couple of options to find and log their food. Thirdly, it uses the Tamagotchi effect. The pet’s animated responses to the user’s dietary habits provides a quick evaluation of the user’s diet and encourages the user to come back.
2.2 Nutrition Buddy’s Contributions

Problems with Food Baby

**Limited Platform** As this prototype has no corresponding mobile application and uses a relatively unpopular smartwatch platform, the potential user base for this application is very small.

**Limited Dietary Choices** The prototype only supports one relatively old dietary plan, which is not appropriate for all users.

**Limited Scope** The prototype only provides a single-day view of the user’s nutrition, resetting at the end of each day. Users can neither see nor be rewarded for good nutrition habits over time, hurting engagement.

2.2 Nutrition Buddy’s Contributions

Nutrition Buddy follows a long line of applications which use digital avatars to promote behavior change. In addition to being a novel application of the technique to nutrition, a field which tends towards heavy interactions, Nutrition Buddy stands out as a multi-class recommendation problem. Whereas others applications of the digital avatar provide a single recommendation (e.g. conserve energy, or drink or water), Nutrition Buddy is in a unique place as the buddy is both simultaneously a guide and a creature to be cared for.
Chapter 3

Recommendations

Recommendations lie at the core of the Nutrition Buddy as it reflects the nutritionist advice experience in a manner that has not occurred in previous NTAs. Recommendations are provided in terms of which food groups to consume based on the user’s actions so far.

The digital buddy’s recommendations must be timely and appropriate for the user’s customized diet. Although judged subjectively, the quality of these recommendations drive the user’s diet and their perceived usefulness of the application.

The recommendations provided by Nutrition Buddy are unique from those provided by human nutritionists and static information (such as government nutritional guidelines) as Nutrition Buddy not only has the context of an individual user at hand when providing recommendations, but also has the advantage of being able to provide continuous recommendations throughout the day. As such, Nutrition Buddy innovates on previous NTAs as well as other nutrition services. Since it has the benefit of long-term context but also the pervasiveness of an application, we require an algorithm that balances short term considerations with long term trends.
3.1 Naive Algorithm

In this chapter we present two algorithms for determining what food group to eat next: the Naive and Historically Weighted algorithms. These differ in how they account for previous data. We discuss the modifications used to provide meal and snack recommendations, as well as negative recommendations, which are recommendations for what to eat less of in the future and are provided after a user has satisfied all categories.

3.1 Naive Algorithm

The naive algorithm for food group recommendation is a natural extension of the simple data model and diet plan. Simply, it is what a user would recommend based on a single day’s reported data. The algorithm picks the food group which has the lowest percentage completed for the current day. If all the groups are satisfied, then no recommendation is given (the application uses the negative recommendation, if there is one). It is given by this equation:

\[
\text{recommendation} = \begin{cases} 
\text{none} & \text{if all groups satisfied} \\
\arg \min_{\text{group}} \frac{\text{group.eaten}}{\text{group.goal}} & \text{otherwise}
\end{cases}
\]

3.1.1 Problems with the Naive Algorithm

Although the recommendation is accurate and would result in the user meeting their goals if he or she were to follow it exactly, it does not provide anything the user cannot obviously see for him or herself. As such, it is as uninteresting as the recommendation “eat more” or “eat less” recommendation provided by current NTAs.

In addition to this, since the algorithm only looks at a single day, this algorithm
3.2 Historically Weighted Algorithm

does not reflect long term behavior. We would like the algorithm to better reflect long term behavior, similar to how a nutritionist might recommend food, while also maintaining the characteristic that it picks a group that makes sense for the current day.

3.2 Historically Weighted Algorithm

In contrast to the naive algorithm, we would like to use an algorithm for food group recommendations which is appropriate for the user’s current situation but takes into consideration long term trends in user behavior.

To further illustrate the algorithm that we desire, take for example a user who consistently overeats protein and rarely eats fruit in the long term. Say that so far today, the lowest satisfied category is protein at 5% and he or she has completed 20% fruit consumption. Which of the following would we prefer?

- Recommend protein, since he or she has eaten less of it today. This is analogous to what the naive algorithm would recommend.

- Recommend fruit, since he or she tends to overeat protein and undereat fruit in the long term. This is analogous to what a human nutritionist would recommend at an appointment for this user.

Of course, neither is definitively correct, since the recommendation is subjectively evaluated. However, it is clear that there is a tradeoff between considering how the user behaves in the long term and how they are behaving in the current day. Our method addresses this.
3.3 Meal and Snack Recommendations

The resulting algorithm uses historical eating data as weights when evaluating which group to recommend based on completion percentages of the current day.

\[
\text{recommendation} = \begin{cases} 
\text{none} & \text{if all groups satisfied} \\
\arg\min_{\text{group}} \frac{\text{group.eaten}}{\text{group.goal}} \times \text{group.weight} & \text{otherwise}
\end{cases}
\]

The weights are calculated based on the last \( n \) eating days logged (not including current day). Say \( D_i \) is the \( i \)-th day before. The group weight is the quantity of that food group eaten over the past \( n \) days plus some small number \( \epsilon \) over the sum of the goals for that food group over the past \( n \) days.

\[
\text{group.weight} = \frac{\epsilon + \sum_{i=1}^{n} D_i[\text{group}.eaten]}{\sum_{i=1}^{n} D_i[\text{group}.goal]}
\]

Note that when implementing the group weight, we add a small \( \epsilon \) to the numerator so that the weight does not take on the value 0. Otherwise, unexpected behavior occurs where a single food group with weight 0 is recommended until it is full (which makes no sense if multiple food groups have not been eaten in the past \( n \) days).

3.3 Meal and Snack Recommendations

In addition to single food group recommendations, the application provides meal and snack recommendations.

To do this, we initialize a dummy state (holds the number eaten and number satisfied for each group) to the user’s state. We then repeatedly run the recommendation
algorithm on the dummy state, update the dummy state as if the user ate that food group, until we reach some desired quantity $n$. For meals, this is $1/3$ of the sum of the group goals.

The result is that we end up with the next $n$ items that the user should eat, assuming that the user follows the recommendations.

### 3.4 Negative Recommendations

If all of the groups are satisfied, we have no recommendation for what to eat next. Nutrition Buddy then produces a negative recommendation, which suggests what to eat less of in the future. The algorithm picks the overeaten group which has the highest weight (as produced in the Historical Weighted Algorithm), as this is the food that is overeaten the most in the long term. The justification for this is that the user is not able to actually act on the negative recommendation, but should keep it in mind for the future, so a daily overindulgence in a single group does not influence the recommendation until it becomes a habit.
Chapter 4

Buddy Animations

The animation of the digital buddy drives the Tamagotchi effect and is essential for user engagement. To animate the buddy, we use a scoring algorithm to determine the user’s performance and translate that score to an animation. The diet evaluation score must be reflected accurately in buddy’s animation to effectively reward or guilt the user. In this chapter, we show both the scoring algorithm we use and the corresponding animations.
4.1 Scoring User Performance

The algorithm used to score the user’s dietary performance is essential for providing meaningful feedback through the digital buddy. We consider both the problems of undereating, overeating, and balance between groups.

Our score ranges from 0 to 1. The score gives 1 only for full performance (all groups satisfied exactly) and 0 if nothing has been eaten or if there is overeating. The score is calculated using the formula:

\[
\text{score} = \sum_{\text{group}} \begin{cases} 
\frac{\text{group.eaten}}{\text{group.goal}} & \text{if } \text{group.eaten} \leq \text{group.goal} \\
1 - 2\left(\frac{\text{group.eaten}}{\text{group.goal}} - 1\right) & \text{if } \text{group.eaten} > \text{group.goal} \text{ and } \text{group.goal} > 4 \\
1 - (\frac{\text{group.eaten}}{\text{group.goal}} - 1) & \text{if } \text{group.eaten} > \text{group.goal} \text{ and } \text{group.goal} \leq 4
\end{cases}
\]

For cleanliness, the formula does not show that the score is minimized at 0, so there cannot be a negative score.

In the first case inside the summation, we simply take the proportion satisfied as the score for that group.

The second two cases handle overeating. Note that these group scores do not minimize at 0, so overeating in a group decrements the score gained by eating well in other groups. One case is if the group goal is greater than 4. This is specific to the implementation of the algorithm: because the food groups are specified in two different units (oz-eq and cups-eq), we double the punishment for the group which is measured in smaller units.
4.2 Appearance

Sad

Used when the score is below 0.25. In this state, the buddy paces slowly from left to right. The position that the buddy moves to is random.

Content

Used when the score is below 0.5. The buddy moves from a random position on one half of the screen to another, and at a faster pace than the sad animation.

Happy

Used when the score is below 0.75. The buddy moves quickly from one of the screen to the other.
4.2 Appearance

Ecstatic

Used when the score is between 0.75 and 1.0. The animation consists of the buddy jumping a small distance to either the right or left of its current position. To start, it consistently jumps in a direction that brings it towards the center (because the transition from the previous state would leave the buddy at one edge of the screen). After it has reached the center, it has a 50/50 chance of going either left or right when it jumps.
Chapter 5

iOS User Interface

In the following sections we give an overview of all of the views in the application, their functionality, where they fit in the user experience and some explanation of the choices made to reach the goals for Nutrition Buddy.

The main application consists of 5 tabs: Home, which allows users to log food, view their current day, get a simple recommendation, and view the buddy; Recommendation, which presents a recommended meal or snack; Trends, which consists of charts which show long term eating behavior; Last 30 Days, which lists the previous 30 days eating; and More, which consists of food group information, an FAQ, and various settings.

Two additional modal views that lie outside of the tabs are described as well: a food group information view and a diet plan selection.

The views for Nutrition buddy are designed for iPhone portrait use. Although the screenshots presented are taken on an iPhone 6, the views resize and adapt to other iPhone sizes.
5.1 Diet Plan Selection Modal View

The Diet Plan change view is the view the user sees when they first log in to the application. Using the segmented control and slider, the view allows the user to select among three different eating patterns and 8 different calorie counts (every 200 from 1600-3200) which are provided by the USDA [1]. This selection forms the basis for the daily goals that the user aims to achieve.

The three eating patterns (called “styles” in the application) are U.S.-Style, Mediterranean-Style and Vegetarian-Style. U.S. Style is based off the “proportions of foods Americans typically consume, but in nutrient-dense forms and in appropriate amounts.” The Mediterranean-Style varies from this by containing more fruits and protein with less dairy. The Vegetarian-Style is based off of a survey of vegetarians conducted by the National Health and Nutrition Examination Survey.

After the initial login, the user can change the diet plan by opening this view from the “More” tab.

Figure 5.1: Diet Plan Modal View
5.2 Home Tab

After the user has logged in for the first time, the Home Tab is displayed whenever the app is launched. This tab is the primary view of the application: the user uses this view to log foods, view the buddy status, get a simple recommendation, and see the status of the current eating day.

The title displayed in the navigation bar is the current date being logged. Days begin and end at 4am, so between midnight and 4am the day displayed is the previous day, and the day is displayed as “Month XX (yesterday)”.

Below that is the recommendation, which is displayed in a speech box. This provides a single recommendation based on the weighted recommendation algorithm. The buddy is animated in the white space between that and a progress bar fills up based on the user’s performance score.

The remaining space consists of 5 cells, one for each food group. On the left of each of these cells is a circle which indicates the progress so far in that group based on the user’s eating for the current day and the diet plan they selected. The name
5.2 Home Tab

of the group is displayed, along with the units it is measured in and a suggested food and quantity for that group. The suggested food and quantities are extracted from the USDA guidelines [1]. Tapping on this brings up the food group information view for the group.

Finally, to the left of each of these cells are decrement and increment buttons, which the user uses to change the counts of the food group consumed for the current day. Changing these quantities changes the circular indicator for the group, the score progress view beneath the buddy, the simple recommendation, and also the buddy’s animation (if the score has changed to a new threshold).
5.3 Meal Recommendation Tab

Although the user receives a simple recommendation for a single food group in the home view, users looking to plan out a meal or large snack can use the chart and suggestions given in the recommendation tab. Using this chart, a user can plan out a meal to best meet their goals.

The algorithm for these recommendations is discussed in detail in the Recommendations chapter. It uses the single recommendation algorithm based on the user’s current eating day, assumes the user follows the recommendation, and iteratively retrieves the next recommendation until the desired quantity of food is reached. This desired quantity is $\frac{1}{3}$ of the total sum of food group goals for the meal, and 3 for a snack. Tapping on a slice of the chart opens up the food group information view for the selected group.

Figure 5.3: Meal Recommendation Tab
5.4 Trends Tab

The trends tab displays two charts that allow users to see their long term eating habits. The charts can be viewed on either a weekly or a monthly time scale. The top chart shows the average consumptions for each food group (colored bars) next to the goal for each group (grey bars). The bottom line chart shows the change in consumption for each group.

Although the recommendation system takes long term eating tendencies into consideration, this view allows the user to take a long-term look at eating to see areas of improvement for themselves.

Figure 5.4: Trends Tab
5.5 Last 30 Days Tab

This view shows the eating behavior for the last 30 days. It is implemented in a Table View where each row corresponds to a single day that the user logged. It uses a Pie Chart to show the relative portions of each food group and displays the counts for each group for that day.

As opposed to the Trends view, which displays aggregate information, the purpose of this view is to allow the user to look back to individual days.

Figure 5.5: Last 30 Days Tab
5.6 More Tab

The More tab consists of food group information, an FAQ, and various settings. From here, the user can view their current diet plan and access the diet plan change view. Additionally, they can open the food group information modal views, and the FAQ sections, which consist of:

Source of Nutritional Info

Provides reference and a link to the “2015-2020 Dietary Guidelines for Americans” [1].

Cup-eq vs. Oz-eq

Explains how the units vary from their unit-eq versions due to nutrient density and where to find examples of cups-eq and oz-eq.

Diet Plan Differences

Explains the difference between the various diet plan eating styles (US, Mediterranean, Vegetarian).
5.6 More Tab

Calorie Count Selection

Explains the factors that ideal calorie counts depends on and links to an online calorie calculator.

Recommendations

Explains how recommendations are generated and where to find them.

The responses to the FAQ appear in alert views.
5.7 Food Group Information Modal View

The Food Group Information views show information about food groups based on the “2015-2020 Dietary Guidelines for Americans” [1]. This view can be accessed from the Home Tab, Recommendations Tab, and More Tab.

Each information view contains example quantities of the food group, examples of healthy options within the food group, key nutrients provided by the food group, and considerations when eating from the food group.

The information provided are condensed from their original appearance in the “2015-2020 Dietary Guidelines for Americans.”

Figure 5.7: Food Group Information View
Chapter 6

Apple Watch User Interface

Nutrition Buddy is built to not require an Apple Watch, and, as such, the Apple Watch does not provide any functionality that is not present on the iOS application. It does, however, provide the basic, most important functionality in a form that allows lightweight interactions: food logging, eating day review, and single food group recommendations.

Although the Nutrition Buddy iOS application is already built to have lightweight interactions, the Apple Watch component takes this further by allowing for even quicker interactions.

The Nutrition Buddy Apple Watch experience consists of three different components: the Watch App, the Watch Glance, and various Watch Complications. These are three components of the Apple Watch experience that Apple allows developers to modify. Each component varies in the amount of time required to access and the amount of information and interaction that is allowed. This chapter provides a detailed explanation of each component, in order of most information and most amount of time required to access (the Watch App) to least (Watch Complications).
6.1 Watch Application

The Nutrition Buddy Watch App provides a table which users can use to log eating and view eating for the current day.

The application consists of a single table view with rows for each food group. Each row has a randomized emoji from the food group, the quantity eaten for the current day, the name of the group, and a “+”, indicating that users can tap to increment the count.

Using this interface, users can log their food in a manner of seconds without having to take out their phones from their pockets. The main bottleneck with the application is loading the application, as it requires loading some data from the phone. This is a well-known issue with the Apple Watch platform [12].

This application, as with all Apple Watch applications, is launched from the Home Screen. It takes one press to open the home screen, and an additional tap to open the application.
6.2 Watch Glance

Watch Glances are Apple Watch views which allow display of “timely and contextually relevant moments” [2] from applications. They are not meant to be interactive, rather, they are a display of the status of the application or user. They are accessed quickly by swiping up on the screen.

The Nutrition Buddy Watch Glance displays the quantities for each food group and provides a single food group recommendation, similar to the speech box from the buddy in the iOS application.

The glance serves two main purposes for the user. First, it is a quicker way of seeing the day’s current eating if the user has no intention of logging food (otherwise he or she should use the watch app). Second, if the user does not have the Watch Complications set up, it is the quickest way for the user to get a single food group recommendation.
6.3 Watch Complications

Watch Complications are small indicators for various applications that users can use to customize the watchfaces which are displayed by default on the Apple Watch. It is the quickest medium for interaction as accessing the complication is as simple as looking at the watch. It also speeds up application launch, as the user can click on the complication to open the corresponding Watch App, bypassing the home screen.

Nutrition Buddy uses Watch Complications to provide single food group recommendations for users. It supports three different kinds: Utilitarian Small, Modular Small and Utilitarian Large.

Utilitarian Small and Modular Small show a single emoji corresponding for the food group that is recommended. Since Modular Small allows for two lines of text, the word “EAT” appears above the emoji. Utilitarian Large, a Complication that spans the entire width of the screen, displays the full text of the recommendation, e.g. “EAT MORE VEGETABLES.”
Chapter 7

Implementation

7.1 Top Level View

There are three main components that make up Nutrition Buddy: the iOS application, Apple Watch Application, and backend services and datastore. Note that data flow between these applications is relatively slow compared to within each component. Data flow between Apple Watch and iOS Application is done via Bluetooth, between iOS Application and web backend via a REST API, and any interaction on the Apple Watch that requires contacting the web backend requires both Bluetooth and the REST API (this is a limitation of the platform, not a design decision).
7.2 Cloud Service

The cloud service stores all of the variable data. This includes eating days, users, and user preferences (iOS uses no local storage besides storing a user token for authentication). It consists of two main portions, a Postgres database and Django web application which serves as an interface between that database and the iOS application.

7.2.1 Technologies

The technologies used to build the backend are:

PostgreSQL is relational database used to store all the data for Nutrition Buddy.

Django is a web application framework that provides many services, most importantly an object-relational-model on top of the database, a web application server and a JSON API that the Nutrition Buddy iOS application uses.

Django Rest Framework is a toolkit build on top of Django that provides serialization and authentication defaults, among other things.

Heroku is a service which hosts the web server that runs the backend.

Google Domains provides a custom domain name.

CloudFlare allows the web application to be requested via SSL.
7.2 Cloud Service

7.2.2 Data Model

In this section we overview the various data structures stored in the Nutrition Buddy backend. These structures correspond to Django model classes, which each correspond to Postgres database tables.

User

We use the default implementation of Django’s user class, which consists of: username, email, first name, last name, and password (which is appropriately hashed in the database).

Token

The token is used for mobile authentication. It is sent in the header of the secure request from the mobile application and is checked when user and eating data is requested. We use Django Rest Framework’s default token implementation, which has a randomized key and a one-to-one relationship to the User model.

Diet Plan

Diet plans store the goals for each food group for a specific user. Diet plans form a many-to-one relationship to users, but in practice we only use the most recently created diet plan in the Nutrition Buddy iOS application, and the previous diet plans are simply retained as eating days (next section) form a many-to-one relationship to them as well. The fields on the diet plans are:

plan_type ‘US’, ‘ME’, or ‘VE’ depending on the eating pattern selected.
7.2 Cloud Service

`plan_type_calorie_goal` Integer for the calorie count selected.

`vegetables_goal` Integer specifying the ideal daily quantity of vegetables.

`fruits_goal` Integer specifying the ideal daily quantity of fruits.

`grains_goal` Integer specifying the ideal daily quantity of grains.

`dairy_goal` Integer specifying the ideal daily quantity of dairy.

`protein_goal` Integer specifying the ideal daily quantity of protein.

Eating Day

Eating days store the data for a single day of eating for a particular user. In addition to having a many-to-one relationship to users, they also have a many-to-one relationship for the diet plan which was being used at the time.

`date` the date of the eating plan, which is sent by the user.

`vegetables` Integer specifying the quantity of vegetables consumed that day.

`fruits` Integer specifying the quantity of fruits consumed that day.

`grains` Integer specifying the quantity of grains consumed that day.

`dairy` Integer specifying the quantity of dairy consumed that day.

`protein` Integer specifying the quantity of protein consumed that day.
7.2 Cloud Service

7.2.3 Resources

Since the iOS application never accesses the database directly, our backend provides services that the iOS application requests to retrieve and make modifications to data. These resources send and receive data using the JSON format.

Signup

Supports POST, at /api/signup/

Converts JSON in POST request to new user, or validation errors if data is invalid.

Login

Supports POST, at /api/login/

Gives token if correct username and password combination

User

Supports GET, POST, at /api/user/

GET: returns data of user, error if not authenticated.

POST: update data of user, error if not authenticated or invalid data

Diet Plan

Supports GET, POST, at /api/diet/

GET: retrieves most recent diet plan for user, error if not authenticated

POST: create a new diet plan for user, error if not authenticated or invalid data
7.3 iOS Application

Eating Day

Supports GET, PUT, at /api/day/[YYYY]/[MM]/[DD]/

GET: retrieves the eating day for the day specified in URL parameters

PUT: creates or updates eating day for day specified in URL parameters

Eating Days

Supports GET, at /api/eating_days/[YYYY]/[MM]/[DD]/[LOOKBACK]

Retrieves the eating days starting at the date which is [LOOKBACK] days behind the date specified in the URL parameters

Recommendation Weights

Supports GET, at /api/recommendation_weights/

Returns the recommendation weights which are used in the Historical Weighted Recommendation algorithm.

7.3 iOS Application

In this section we discuss some components of the iOS application implementation, including the frameworks used, animations, and the management of application state and data. However, we deliberately leave some components of the implementation out. This includes:

- The implementation of the view interfaces, connections between views, custom subviews, as these are generally solved problems with standard implementations, despite the significant amount of time spend developing this component of the
7.3 iOS Application

application. Additionally, the user interface is discussed in detail in a previous chapter.

- The implementation of recommendations, as the algorithm is covered in the chapter on recommendations.

- Buddy happiness scoring, as the algorithm is covered in the section on Buddy Animations.

- Notifications, which occur when the user has not logged at least 50% of their total goal for the day. The implementation simply schedules a recurring daily notification starting from the current day or the next day, depending on the user’s eating.

7.3.1 Frameworks

In addition to the standard iOS development frameworks provided by Apple, we used:

**Carthage** a dependency manager for Cocoa

**Alamofire** a package that assists with HTTP networking

**Locksmith** a library for working with the iOS keychain (we use for tokens)

**Charts** a charting library used for the Recommendations, Trends, Last 30 Days, and Diet Plan Change views.

**EasyAnimation** a library used to assist with animations for the buddy

**Mixpanel** a mobile analytics platform for tracking usage data
7.3 iOS Application

7.3.2 Storing Application State and Data

Although we do store all of the data for the application in the backend, we use a temporary data store (does not persist between application sessions) to hold the data, reducing load times by reducing the number of requests that are required. We call this the Status, which is a singleton. We make the assumption that the user is accessing the application from a single iOS device and, but not necessarily, an Apple Watch, so the underlying data does not change during usage of the application unless the user deliberately changes it.

Without using this singleton, the diagram for interactions between the data store and the view controllers look as follows:

As you can see from this diagram, every change in view requires loading data from the backend. This becomes costly, especially in the case where the Apple Watch needs to access data, as the data needs to be transferred over both bluetooth and the REST API. However, with the singleton we have a improvement:

---

41
7.3 iOS Application

Using this model, we reduce connections to the backend to the following cases:

- Application load, when the data for the singleton is populated (this does not apply when the app is resumed, only when it is launched)
- When charts and eating day lists are requested (to reduce memory consumption)
- When the user updates data: either changing the count of a food group for the day or changing the diet plan

As an example, consider the usage of the Apple Watch components of the application. Assuming the iOS application has been launched somewhat recently (such that operation system does not terminate the application to free up memory), the load of the Watch App, Glance, and Complications now only require the connection from the application to the phone to be active, and do not require an REST API call.

7.3.3 Buddy Animation System

Nutrition Buddy is built within the standard UIKit-based iOS development system. Animations in this system typically are short-lived, small interactions that occur on a page load, button action, etc. As such, the animation of the buddy is atypical, as it is a long-living animation without many libraries to support it – one would typically see this kind of game-like animation in an application built on SpriteKit, Unity, Cocos2D, or other similar framework, which have completely different models for structuring an application.

As such, the animation of the buddy is a result of stringing multiple short-lived animations end-to-end. Nutrition Buddy uses a system for this built with the following desires in mind:
Modularity

The logic for buddy animation should, as much as possible, lie outside of the view which uses it. This allows for the buddy to appear in multiple places in the application without much hassle, and keeps the contents of a view controller clean. Ideally, we would like added the buddy to be as similar as possible to adding another subview to a view.

Extensibility

The buddy should be built in a manner where animations and states are easily added, removed, and modified.

The features that it needs to support are:

- Various states with different animations and images for those states
- Dynamic state switching (so when happiness score is updated animation follows)
- Proper state transitions, e.g. a buddy in midair should go back to the ground before switching to next animation state

Our resulting animation system requires very little input and interaction from the view controller. It is implemented in a single Buddy class with these public methods:

- **animate** takes in an image view that has been added to the view, and bounds where the buddy can jump around. Should be called when the view appears.

- **stopAnimations** stops the animation of the image view. Should be called when the view disappears, or too many animations on the same image view get created.
7.3 iOS Application

**statusWasUpdated** is called whenever the happiness score of the buddy could change – namely, when the eating is updated. This does not necessarily change the buddy’s animation state.

This simple API makes it so the buddy can be placed in multiple views with ease. So far, these views are the landing page view and the home view.

Internally, the animation is implemented as a series of small animations that compose the continuous animation of the buddy. The buddy keeps track of a class that implements **BuddyAnimator**, an interface we will later describe, that is set when **animate** and **statusWasUpdated** is called. This **BuddyAnimator** handles the actual animation of the image view for a state, whereas **Buddy** controls the high level animation transitions. The implementations of the public method and the private method **nextAnimation** are explained below:

**animate** stores the image view and bounds. Uses the current status to determine the **BuddyAnimator** to use, and calls **nextAnimation**.

**stopAnimations** moves the position of the buddy to the ground and sets a boolean that prevents **nextAnimation** from initializing future animations.

**statusWasUpdated** checks to see if the state has changed from the previous state. If it has, run the finishing animation for the old **BuddyAnimator**, update the **BuddyAnimator**, then run the new **BuddyAnimator**’s animation.

**nextAnimation** runs the animation provided by the current **BuddyAnimator**.

The completion handler for the animation calls **nextAnimation** again.

Using this implementation, **Buddy** manages the high level animation flow. Of key importance in the implementation of **Buddy** is **statusWasUpdated**. The call
7.3 iOS Application

to the finishing animation of the old BuddyAnimator ensures that the buddy is properly reset during state transitions.

BuddyAnimator is a protocol which keeps the animation code extensible and modular between states. The required methods for BuddyAnimator are:

init takes in an image view and bounds.

getNextBuddyAnimation returns a tuple with animation parameters (duration, delay, options), and a function that contains the animation to perform.

finishingAnimation returns the same thing as getNextBuddyAnimation, but is used to reset the buddy state.

Using this simple protocol, we can implement a large variety of animations. Since an implementation of this protocol can store an arbitrary other number of instance variables, there is a lot of flexibility for animations. At the bare minimum, we store the images placed in the image view. In addition, we can add state variables to control animation. For example, we could keep track of a list of the next several positions to move to and put give each of those in a separate animation in different calls to getNextBuddyAnimation. In practice, EcstaticBuddyAnimator uses something similar to this, as the jump animation is split into separate animations that have a state monitored by the animator.
Chapter 8

Results

In this chapter we review collected data during a 20 day beta test period, a survey sent out to beta test users, and statistics gathered from usage data. We use this data to analyze performance towards key goals posed in the early chapters of this thesis, namely, the lightweight interface, comparison to other NTAs, the recommendation algorithms, and the buddy. We also discuss the qualitative feedback we received through these surveys.
8.1 Methodology: Beta Testing and Survey

To evaluate various aspects of the application, we beta tested the application among 32 users (we only count those who logged at least a single eating day) over the course of 20 days. The histogram of the number of eating days logged vs the number of users that logged that eating day is shown below:

![Histogram of Number of Eating Days Logged vs Number of Users](image)

It is clear that there is quite a bit of variance in contact with the application. For the statistics that we present in the results section, we will choose a subset of this group, as the number of eating days logged is a reflection of engagement with the application.

In addition to the beta test, we conducted an online user survey near the end of the beta test, open only to our beta testers. There were 21 responses, answering questions regarding a variety of aspects of the application. 8 of the respondents had used a different NTA in the past and answered addition questions regarding Nutrition Buddy vs other NTAs. The majority were college students affiliated with Dartmouth College. 8 were females and 13 were males.
8.2 Overall Performance

We posed two questions that attempted to gauge overall effectiveness of the application, one asking if Nutrition Buddy influences respondents’ diets positively, and another asking about ease of use.

![Graph showing health eating and ease of logging meals](image)

On the first question, regarding health eating, the survey response shows that respondents generally eat the same or slightly better than they usually do when using Nutrition Buddy. Given that our beta testers were not necessarily people who were looking to improve their nutrition in the first place, this could be interpreted in an even better light.

Responses to the second question indicate that users in general find it very easy to log food on Nutrition Buddy, which achieves one of our main goals, lightweight and easy user interactions.
8.2 Overall Performance

8.2.1 Comparison to other NTAs

In the online user survey, we asked additional questions to users who have used other NTAs. Of 8 respondents who said they used a previous NTA, 4 used “My Fitness Pal,” 2 used “Wholesome,” 1 used “Lose It,” and 1 used “Jawbone Tracker.”

We note that, given that there are only 8 responses for the following questions (as opposed to 21 for the survey as a whole), the results should be viewed with some skepticism, regardless of whether it is positive or negative for Nutrition Buddy.

Two of the questions posed in this section asked about ease of use: the first asked about speed relative to other applications, and the other asked users to say whether it was less or more cumbersome.

![Bar chart showing comparison of ease of use between Nutrition Buddy and other applications.]
8.2 Overall Performance

The results from both of these surveys are consistent with the question posed to everyone, “How easy was it to log meals in Nutrition Buddy?” They show that the testers generally perceived Nutrition Buddy to be easier and quicker than alternative NTAs.

The third question posed asked whether Nutrition Buddy was better or worse than the competing NTAs that users used.

The results were mixed, suggesting that Nutrition Buddy is on par with others that they have used. We also asked for qualitative feedback, which provides some insight – one mentioned that they would have “liked more customizations,” and another said they were not sure if it “fully captured the story.” We note that this question is not necessarily a measurement of achieving our goals, but rather that we do not stray too far from current NTA performance; our solution to the nutrition tracking problem is based off of a tradeoff between detailed information and ease of use, accommodating a different kind of user.
8.3 Recommendations Evaluation

For the beta, we split the users such that half were given the Historical Weighted algorithm for determining recommendations and half were given the naive algorithm. They did not know which algorithm they received. The combination of both groups shows that we have a decent usage of recommendations and they are appropriate.
8.3 Recommendations Evaluation

8.3.1 Historical Weighted vs. Naive Algorithm

Since we split the beta testers into the two groups, we can evaluate the performance of the different recommendation algorithms relative to each other.

We number the questions accordingly:

1. How often did you use the recommendations to inform your eating habits?
2. How appropriate were the recommendations based on your current day’s eating?
3. How appropriate were the recommendations based on your overall eating tendencies?

Below shows a table of the mean response for each question, along with the P Value of the result.

<table>
<thead>
<tr>
<th></th>
<th>Weighted</th>
<th>Naive</th>
<th>P Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># respondents</td>
<td>13</td>
<td>8</td>
<td>N/A</td>
</tr>
<tr>
<td>Question 1</td>
<td>3.38</td>
<td>2.75</td>
<td>0.2795</td>
</tr>
<tr>
<td>Question 2</td>
<td>3.85</td>
<td>4</td>
<td>0.6674</td>
</tr>
<tr>
<td>Question 3</td>
<td>3.62</td>
<td>4.13</td>
<td>0.2578</td>
</tr>
</tbody>
</table>

Based on the table above, users in the survey using the Historical Weighted algorithm were more likely to use the recommendations, but they responded that the recommendations were less appropriate on both an overall and daily timescale. We also observe that this data is far from statistically significant, so we do not gain much insight using this data.

Another analysis we performed was to use the happiness scoring metric for the buddy and see which group performed better among all beta users who have logged at least 3 eating days. 12 Historical Weighted and 8 Naive algorithm users are present in this filter. The average score for Historical Weighted is 0.56 while Naive is 0.44 (with
8.4 Buddy Evaluation

P value of 0.0158), suggesting that those receiving the Historical Weighted algorithm perform better.

To summarize: the results found from our analysis of Historical Weighted vs Naive algorithm users suggest that the Historical Weighted algorithm users come closer to meeting their goals (statistically significant) and may use recommendations more (not statistically significant), but may not find the recommendations appropriate (not statistically significant).

8.4 Buddy Evaluation

Our results from the Buddy evaluation show that the buddy has mixed effects on users. Approximately half felt that the buddy somewhat often informed their decision making. A significant portion of users agreed that the buddy's motion effectively portrays the quality of their eating.

Our results from the Buddy evaluation show that the buddy has mixed effects on users. Approximately half felt that the buddy somewhat often informed their decision making. A significant portion of users agreed that the buddy's motion effectively portrays the quality of their eating.
8.5 Qualitative Feedback

making, while the other half felt that it almost never did.

We found that there are a substantial amount of users (around 30%) who do not believe that the buddy accurately portrays their eating. This suggests that there is room for improvement in the buddy, perhaps improvements that could cause some users to allow the buddy to inform their decision making more. The following section goes over the qualitative feedback, some of it discussing the buddy.

8.5 Qualitative Feedback

At the end of the survey we included an optional feedback text field where respondents could enter in any feedback they like. From reading these responses, three areas of difficulty emerged:

Quantifying Items

Many users complained of having trouble quantifying “cups-eq” and “oz-eq.” They suggested adding more example quantities and having a calculator that finds the quantities from a food (e.g. sandwich).

Depth of Information

Some users suggested that the application did not capture enough information about a users diet, especially regarding unhealthy foods as there is no option to punish based on dessert, alcohol, etc.

Buddy Reaction Some users did not feel that the buddy reacted quickly enough to their eating e.g., since the buddy only changes animation when a threshold is reached, a user could be eating very well but the buddy would not be happy until later in the day.
Chapter 9

Future Work

9.1 Units

The most common piece of feedback regarding the application was that quantifying oz-eq and cups-eq was confusing. Although some work was done during the development of the application to make this less confusing, there is still much that could be done:

- Getting rid of oz-eq and cups-eq altogether, instead relying on some alternative measurement such as servings, which is typically easier to find. The main tradeoff with this approach is that the application would no longer fit with the “2015-2020 Dietary Guidelines for Americans” [1], so some alternative would need to be found.

- Adding onboarding. Currently there is no onboarding after or before the signup flow which explains the purpose and how to use the application. Adding this could help understanding quite a bit.

- Providing more suggestions for quantities within food groups, or including vi-
9.2 Buddy Improvements

Feedback from the user survey suggested that, while there were about half of users who felt that the buddy influenced their decision making, there is still room for improvement. Qualitative feedback discussed making the buddy more responsive to eating well in the short term.

One improvement for the buddy is to make short buddy animations which respond to immediate good or bad eating behavior, which then fall back to a resting animation state based off the overall day. For example, the buddy could jump if the user satisfied food groups they have not reached the goal for yet, or make a displeased face if the user overeats a food group. This short animation could be triggered as the user is logging multiple groups or after a “meal” is logged, which is when the user taps on a bunch of eating groups and then saves it by clicking a button (another potential feature).

Additionally, another improvement is to base the buddy’s animations on not just the current day but long term behavior. For example, the buddy could become wider if the user consistently overeats.

Another improvement is to add the buddy to the Apple Watch application to provide a more pervasive interaction with the buddy.
9.3 Launch and Further Testing

Some of the collected data had small sample sizes due to needing to split the beta testing population. In addition to allowing Nutrition Buddy reach a bigger audience, releasing to the App Store will allow us to collect more metrics on how Nutrition Buddy’s various components perform.
Chapter 10

Conclusion

Motivated by the problems of heavy user interaction, lack of motivators and insufficient and inaccurate data that we saw in current NTAs, we designed Nutrition Buddy, a new NTA which uses simplifies the food consumption data model to provide a more lightweight, engaging experience through recommendations, wearable computing, and a digital avatar. In our user surveys and usage data collected through a 20-day beta testing period, we found that users agreed that Nutrition Buddy was an easy-to-use, lightweight NTA on its own and in comparison to others; we were able to improve on user performance by using a Historical Weighted Recommendation algorithm compared to a naive algorithm; and our algorithm for scoring user performance and converting that to animation was able to further engage users (with promise for improvement). As a final product, we have produced a NTA which provides a simplified, engaging experience for those looking for an alternative to the popular NTAs which require significantly more user input.
Bibliography


[5] FitNow. (2015). Lose It! (6.5.4) [iOS application].


[7] Fourdesire. (2014). Plant Nanny (1.9.2) [iOS application].


