2013

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

DINE HEALTHY MOBILE:

AN IMPLEMENTATION OF AN HTML5 MOBILE WEBAPP

Adam Forsythe

A Capstone Project Submitted to the

University of North Carolina Wilmington in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

Department of Computer Science

Department of Information Systems and Operations Management

University of North Carolina Wilmington

2013

Advisory Committee

Dr. Douglas Kline	Dr. Darwin Dennison
Mr. Royce Nobles	Dr. Ron Vetter, Chair

Abstract

DINE Healthy Mobile: An Implementation of an HTML5 mobile webapp. Forsythe, Adam, 2013. Capstone Paper, University of North Carolina Wilmington.

This paper describes the design and development of a nutrition assessment and diet improvement mobile web application for a company that specializes in nutrition analysis software, called DINE Healthy, LLC. The purpose of this mobile webapp is to allow customers to track their diet and get personalized nutrition analysis while on the go. It will serve as a mobile companion and synchronize with a fully featured cloud-based web application in production. This paper explains the decision to build this product using web technologies that allow for cross platform deployment versus a native solution. Also included is detailed information on the analysis and development process including market analysis, user interface designs, and use case and project management documentation. This paper discusses technical challenges and concludes with explanations on how deliverables were addressed, lessons learned, and future work for this mobile webapp.

Table of Contents

Page

Chapter 1: Introduction

More than half of the leading causes of deaths in the United States are associated with poor food choices and physical inactivity resulting in heart disease, cancer, stroke, diabetes, liver disease, and arteriosclerosis [1]. Poor diet and lack of activity not only cause chronic medical conditions, but also have economic consequences, causing healthcare costs to skyrocket. For instance, the total cost of those diagnosed with diabetes (25.8 million) soared 41% in just a 5-year timeframe, from $174 billion in 2007 to $245 billion in 2012 [2]. Obesity accounted for $147 billion in medical care costs in 2008 and has affected more than a third of adults and 17% of children in 2009-2010 [3, 4]. Millions of Americans suffer chronic diseases, and the good news is that these could be improved or even prevented with diet and regular physical activity. DINE Healthy, LLC can help combat this growing epidemic with software products that provide comprehensive nutrition analysis for those who wish to improve their health. This project implements a mobile web application designed to help customers keep track of their diet and health progress using any popular mobile device.

1.1 Overview

DINE Healthy, LLC is dedicated to building the best diet and health-improvement software products for home-users, nutrition education, nutrition counseling, sports teams, and physical activity programs. With the increasing number of smartphone users and rising demand for mobile applications, DINE Healthy, LLC is looking to round out its product line and tap into the mobile market. The goal of this project is to develop a mobile Web application, DINE Healthy Mobile, which will make it easier for our customers to track their foods and get personalized diet analysis from any popular mobile device.

The intention of DINE Healthy Mobile is to increase profitability and help our customers improve their health by increasing customer satisfaction, interaction, and adherence to the software. This project details the design, implementation, and launch decisions made for DINE Healthy Mobile. It explains how it fits in with the company's current product line and the decision to build a mobile Web application versus a native smartphone application.

1.2 Evolution of DINE Healthy, LLC Nutrition and Fitness Technology

Formally DINE Systems, Inc., DINE Healthy, LLC has a history of innovation in nutrition software for over 20 years. The company released the first micro-based nutrient analysis program in 1982, introduced the first pictured food database on CD-ROM in 1995, and created the patented DINE® Score to help customers understand "how well" or "how poorly" they are eating [5]. This record of innovation continues with launching DINE Healthy Web in 2013, a web-based version of our flagship software product, DINE Healthy Desktop. Figure 1 depicts a timeline for these innovations. Today, the company is seeking to make its mark in the mobile arena with DINE Healthy Mobile, a mobile Web application.

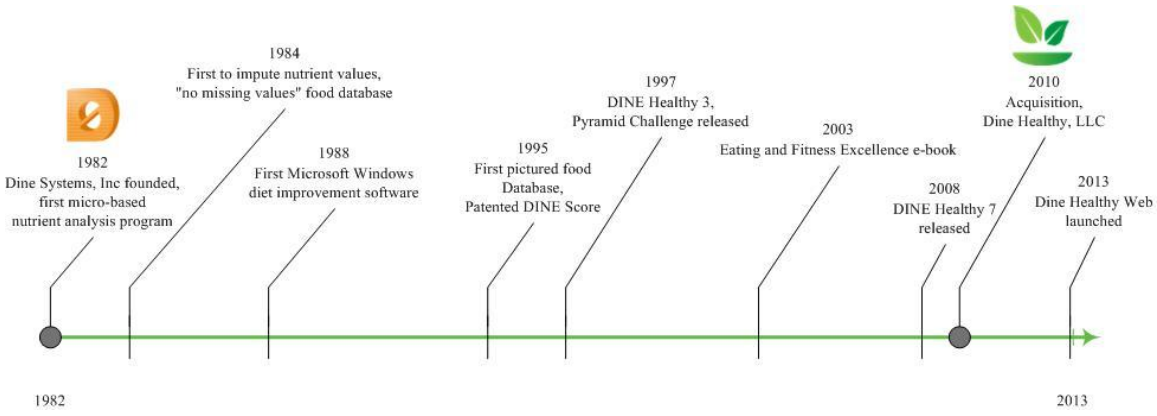


Figure 1: DINE Healthy Timeline

1.3 Purpose of Project

The primary goal of this project is to improve customer satisfaction, interaction, and adherence to the product, thus increasing sales. With the proliferation of smartphones and mobile applications, and new competition entering into the market, it is imperative that DINE Healthy, LLC (DH) continue to build cutting-edge products like DINE Healthy Mobile (DHM) to stay relevant in the industry. This project should also lay the foundation for future software development for the company. A secondary goal of the project is to learn more about mobile technology, mobile application development, and mobile device capabilities and opportunities.

Chapter 2: Market Review and Analysis

2.1 Market Review

At the time of writing this, there are 247 iPhone apps listed on the iTunes App Store and 24,835 Android apps listed on Google Play for the search term "Health and Fitness". Each covers some facet in the realm of health and fitness including, but not limited to, weight loss, grocery list management, counting calories, counting steps, weight training, food tracking, heart rate monitoring, and so on. The purpose of DHM is to allow customers to journal the foods they eat and receive analysis of nutritional content for their daily food intake while on the go. DHM is an extension of DINE Healthy Web (DHW). Any changes made using DHM will be synchronized with DHW and vice versa. A review of three competitors that have a presence in the mobile market and have similar services as DH will help provide perspective.

FitDay

FitDay is software developed by Internet Brands® Health to track foods, log activities, and track progress [6]. The focus of their programs is primarily on individuals seeking to lose weight. FitDay has a "freemium"[1] business model, offering their ad-supported web application free of charge while charging for their full featured, premium software. Customers have the option to use a web application, desktop software, or an iPhone app that is available on the iTunes App Store for $1.99. Since FitDay does not have mobile Web version, users are encouraged to use their iPhone app when requesting the program on a mobile browser. Figure 2 depicts the food journal view for the FitDay iPhone app. Tabs located at the foot of the view include Home, Foods, Activity, Weight, and Calendar. The Home view is for creating and updating Profile information like

[1] http://en.wikipedia.org/wiki/Freemium

height, weight, date of birth, et al. Foods view is the interface to log foods. It contains a food search bar, a summary of Energy, Fat, Carbohydrates, and Protein, serving amounts and calories for each food logged, and the ability to change which day to view and log foods. The Food View interface is presented after searching for a particular food. It also has a summary of the food and offers the ability to view Nutrition Facts for that searched food. The app has reporting and graphs for total day compositions, total nutrition breakdowns, weight tracking, and calories burned. It has the ability to create custom foods and log activities. Overall, the application is an easy to use food and activity entry system and synchronizes with the web application. However, the performance of the app is slow and seems to lag when refreshing between views. It does not provide the ability to update a food on a food record. Reporting is done by accessing the "More" button on the header. It seems more appropriately placed on the footer in the tab controller instead.

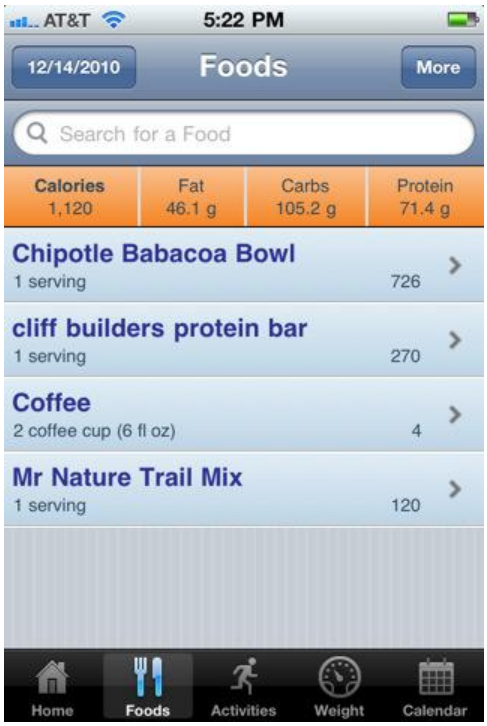


Figure 2: FitDay Foods View

MyFoodDiary

The company's tag line is "Calorie counting made easy". My Food Diary focuses on applications for the more novice-level individuals seeking assistance in weight loss via counting calories [7]. The company advertises that customers can us desktop, iPhone, and mobile versions of their software. However, this is misleading because the iPhone version is not actually a native app, but a mobile Web application designed specifically for the iPhone and is not available on the iTunes Apps Store. Another "turn-off" is that credit card information is required to sign up for a free trial and is automatically charged after 7 days if not cancelled. The highlighted features for the food journaling application include 80,000 foods in their food database, ability to track 15 nutrients, and ability to build a recipe. Figure 3 illustrates the MyFoodDiary "iPhone" web application. The default screen titled "MyFoodDiary", has options for Food, Exercise, Body, and Reports. On the Food view, there is button named "Menu", but it only navigates back to the MyFoodDiary view. Adding a food to the journal is a bit clunky, in my opinion. Once you search and select the food choice from the search results, a "Next" button is positioned in close proximity to the Safari footer options (Bookmarks, Window, and Navigation). Instead of requesting the next view, I tapped Bookmarks accidently. After successfully requesting the Add Foods view, the "Add To My Food Diary" is located in the same awkward position just above the Safari footer. The application is just as responsive as the FitDay native iPhone application, but does not have the native look and feel. For example, the Safari URL and search bar is static, leaving less room for the application to breathe. This less than ideal functionality is not depicted in figure 3, but occurs when using the system. This could potentially confuse customers when attempting to search for foods. The customer would expect a list of foods returned, but would

actually search for web pages via the device's default search engine. Overall, the mobile web application performs well and has a good look and feel. However, improving the placement and naming conventions of the user interface components would help the user experience.

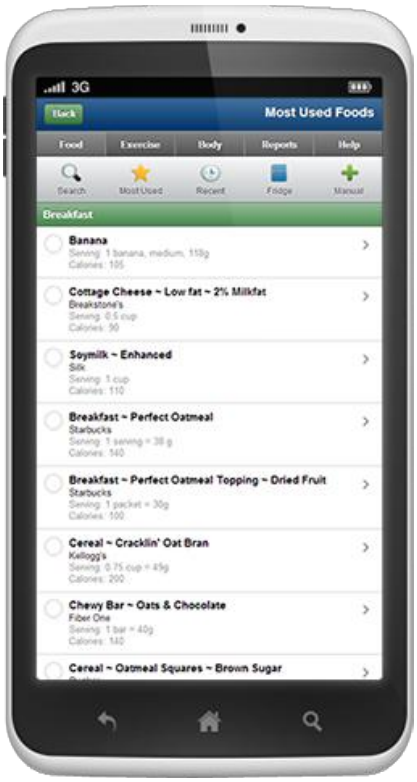


Figure 3: MyFoodDiary Most Used Foods View

MyFitnessPal

MyFitnessPal (MyFitnessPal, LLC) has the largest footprint in the mobile market supporting all popular mobile devices including iPhone, iPad, Android, Blackberry, and Windows Phone 7. The free mobile apps track food and exercise and synchronize with their web application [8]. MyFitnessPal boasts of having the largest nutritional database and the easiest calorie counter on the web. Figure 4 shows the home view for the MyFitnessPal iPhone app. The iPhone app takes advantage of social media, allowing you to sign in with Facebook credentials and quickly connect and add new friends [13]. It

provides an overview of calories consumed versus energy expended and summary breakdowns of major macronutrients (Fat, Carbohydrates, and Protein) and common micronutrients (Vitamins A and C, Calcium, Iron). The app is very responsive and the workflow makes good sense. For example, to add to the Diary view, an "Add" button is located on the upper right of the header. Tapping "Add" brings you to the Add Entry view, which allows for several choices before adding to the Diary. There, you can select during which meal (breakfast, lunch, dinner, and snacks) a food was eaten, change the diary date, choose between adding cardiovascular and strength activities, add water, or enter notes to your Diary. The Diary makes no partition between food logs and activity logs, but instead, combines all events onto one day, in one place. MyFitnessPal mobile app is feature-packed and is, in my opinion, the benchmark for mobile applications designed for tracking diet and exercise.

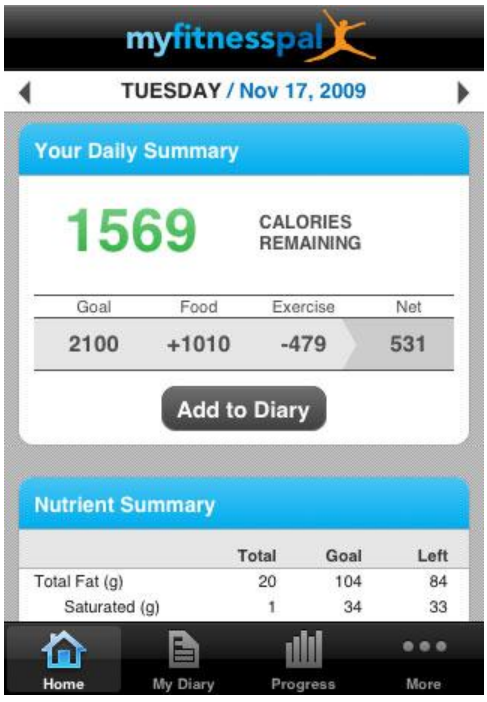


Figure 4: MyFitnessPal Summary View

Summary

Apps designed to log foods, exercise, and track progress possess similar features and user interfaces. Generally, the apps will have a food or activity search in order to search and add a food or activity to a journal. They provide some nutrition content for a food or energy expended for an activity. It can show progress metrics such as weight loss over a period of time, a summary of nutritional composition for daily intake or a date range.

Each of the competitors mentioned have focused on the individual who is seeking assistance with weight loss. DHM will contain these types of features. However, DHM will not only assist the individual user in managing weight, a customer can choose a diet that is best suited for their individual needs. For example, a customer can use the DASH diet to help manage hypertension (high blood pressure) or the renal (kidney) diet to help manage CKD (chronic kidney disease). DHM also will contain the patented DINE score when using the DINE diet that is based on United States Department of Agriculture (USDA) Health Eating guidelines. DHM will also help bridge the connection between healthcare professions (physicians, nutritionist, dietitians, and personal trainers) and educators and their clients. Clients can go about their day tracking their foods and activities while the professionals will be able to council and monitor progress of their clients. Out of the three competitors reviewed, MyFitnessPal was the only app that has taken advantage of a device's hardware features. They offer a barcode scanning feature designed to make it easier to add a food by taking a picture of the food's barcode. Here is a list of common features that these type of apps offer:

- Searchable food and activity database
- Food and activity tracking / logging

- Add, edit, and delete foods or activities from a daily record
- View nutrition content for a food
- Nutrition summary for given day or data range
- Ability to set a weight goal
- View graphs and breakdowns for nutrition content
- Ability to add custom recipes and foods

2.2 Evolving to web-based technology – Release of DINE Healthy Web

Since the acquisition in 2010, DH has been positioning itself to begin building products that will run on the Web and mobile platforms. Some notable changes include the following:

- Updating the DINE Healthy Desktop (DH7) to the latest .NET framework
- Overhauling the corporate website and the desktop software's activation system from PHP to .NET/C#
- Replacing a third-party shopping cart with a custom online shopping cart for added flexibility

These were just a handful of changes made that would allow DH to focus development efforts toward Web and mobile. In two years, the stage was set and DH was ready to start building its first web-based product, DHW, a web-based version of the DH7 software. DHW was released in 2013 and incorporates many of the same features as the desktop version including the foods databank, built-in diets (DINE, DASH, Renal), and similar diet analysis. DHW currently supports the current and prior major release of Chrome, Safari, Internet Explorer, and Firefox on a rolling basis. The product employs web services consumed by jQuery AJAX methods to drive this ASP .NET Web

application. The front-end relies heavily on custom scripted jQuery[2] (a JavaScript library) and JavaScript. All of the dynamic content (reporting graphs, grids, and layout) is custom-built using jQuery/Ajax/JSON. This allows DHW to be very responsive since there is minimal "round-trip", full post backs to the server. Now that DHW is in live production, DH is ready to mobilize.

2.3 DINE Healthy Mobile Product-Market Fit

The DH software target market is not only for the home-user, but also for healthcare professionals (physicians, nutritionists, dietitians, and personal trainers) and educators who manage one or many clients. The DH7 software facilitates the professional-client(s) relationships by allowing client profiles be created and stored locally. Client profiles have personal information such as height, weight, age, and activity level, all of which tell the software what decisions to make for diet recommendations. DH believes that a better way to handle this communication between professionals and their clients is by bridging the communication between the desktop software and web application, DHW. With this approach, professionals would use the desktop software and their clients would use DHW and eventually DHM. The professional could then view a "dashboard" from within the desktop software to see how well their clients are meeting their individual health goals. The professional would have the convenience within the software to communicate to the client, eliminating the need for unnecessary scheduling and face-to-face communication. The professional could use the technology to do things such as prescribe and recommend better food choices, view diet analysis reports, and eventually see how well their entire client-base is performing as a whole. A mobile application would allow customers more convenience when tracking their health. With a

[2] http://jquery.com/

mobile application, the customer does not have to be bound to computer or laptop, so it will also provide better adherence to the program. The mobile application should be an extension of DHW, allowing the clients to keep track of what they eat and how they are progressing. Figure 5 represents how healthcare professional, their clients, and home user could interact with the DH product line.

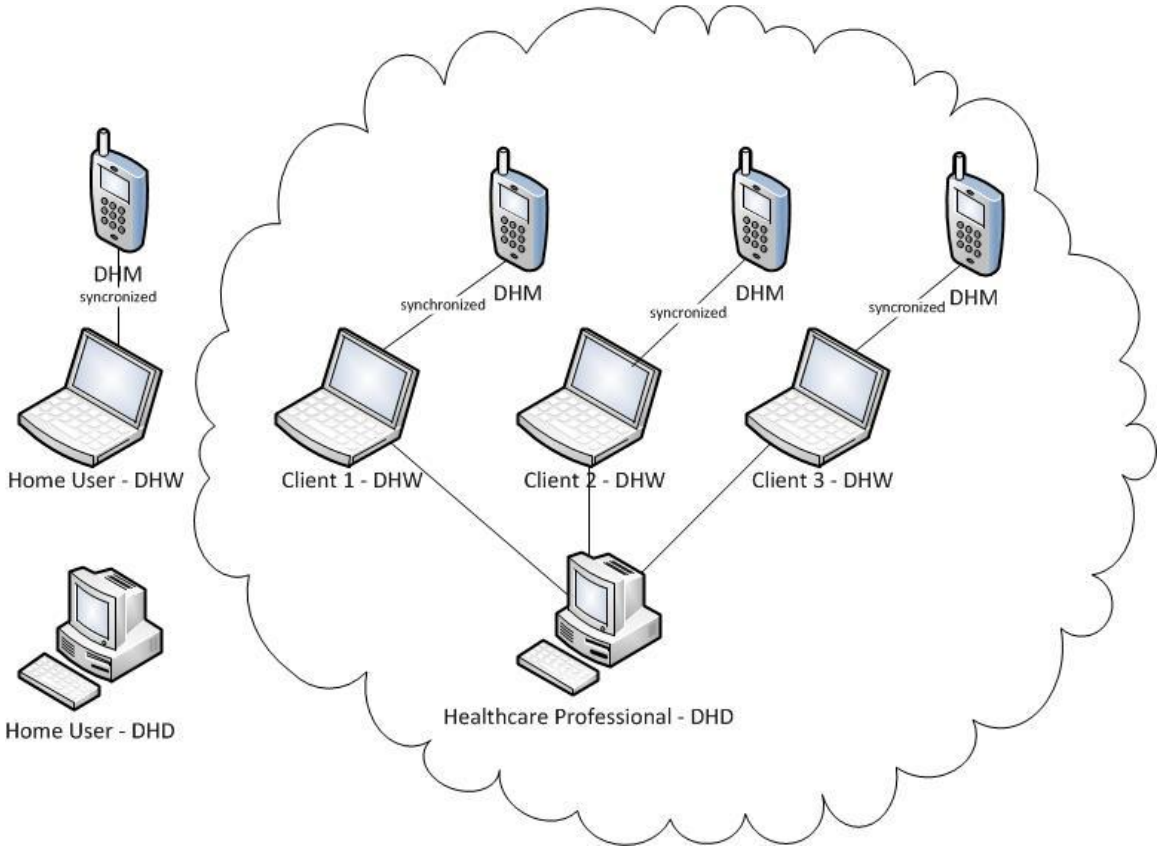


Figure 5: DH Future Product Architecture

2.4 Initial Mobile Strategy – iPhone

At first, building an iPhone application seemed the best mobile strategy for DH. Primarily because we (the owners) are biased and own and are familiar with the devices, we needed access to the device's hardware features such as the camera, and the marketing exposure that comes with having an app on the Apple App Store. I began researching this platform and our objective was to build a prototype of the iPhone mobile

application for DH, which would eventually evolve into the production app. The learning curve was steep for the native environment of the iPhone since it was all new territory for DH. Coming from a J2EE background and recently learning the .NET/C# framework to build DHW, the syntax of the iPhone's native language, Objective-C built upon the C language foundation, is relatively similar to C# and easy to follow. The trickiest concepts were the implementation of its pointers and garbage collection features. Regarding development environments, I have experience developing applications using popular Integrated Development Environments (IDE) such as IBM's Rational Application Developer and Microsoft Visual Studio. The Xcode development environment was foreign at first, especially since I am more familiar with developing on a Windows PC versus a MAC. As far as the user interface, designing an interface that fits all of the product's features on a device window that is 320pts x 480pts (iPhone 4 and earlier) was challenging[3]. For the prototype's user interface, I wanted the app to have a tab bar interface (UITabController) which would be located at the bottom of the device window with a navigation interface (UINavigationController) to help manage content. This interface system would allow users to better navigate views and content of the app [9, 10]. Establishing this interface system and learning how to pass data between views allowed for swifter development of the prototype. I managed to build a working prototype of the mobile app within a few months timeframe. I built mock data for user, foods, and nutrition data in order to work with the app and perform actions like searching foods, adding, deleting, and updating foods on a food record. Having a working prototype was very exciting. We were under the impression that all the was required for us to launch our app was to just clean up the prototype, connect it to actual data, and list the

[3] http://www.idev101.com/code/User_Interface/sizes.html

app on the App Store. Wrong. We still had to come up with a way for the app to run without a wireless connection. This translates to rewriting and duplicating our business logic for the iPhone in Objective-C so it can communicate with the iPhone local database (SQLite[4]). This also meant that we would then need a mechanism to synchronize this local data with live data. Moreover, we needed an installation strategy to determine how we would load up the food databank and other vital elements for the app to run when a customer downloads the app from the App Store. We certainly would not want our app to take too long to download and use. There was still the issue of securing and supporting the app. Code duplication, synchronizing data, installation and security concerns were an iPhone app deal-breaker. These concerns lead DH to rethink its mobile strategy.

The time and efforts spent researching the iPhone app solution was not a total loss. DH gained an enormous amount of knowledge about the obstacles faced when developing a mobile app. Having experience in both the native and web application environments, DH can now make an informed decision on whether to pursue a native app or mobile Web application. Figure 6 depicts the original mock up I designed for the food record view and the resulting prototype view.

[4] http://www.sqlite.org/

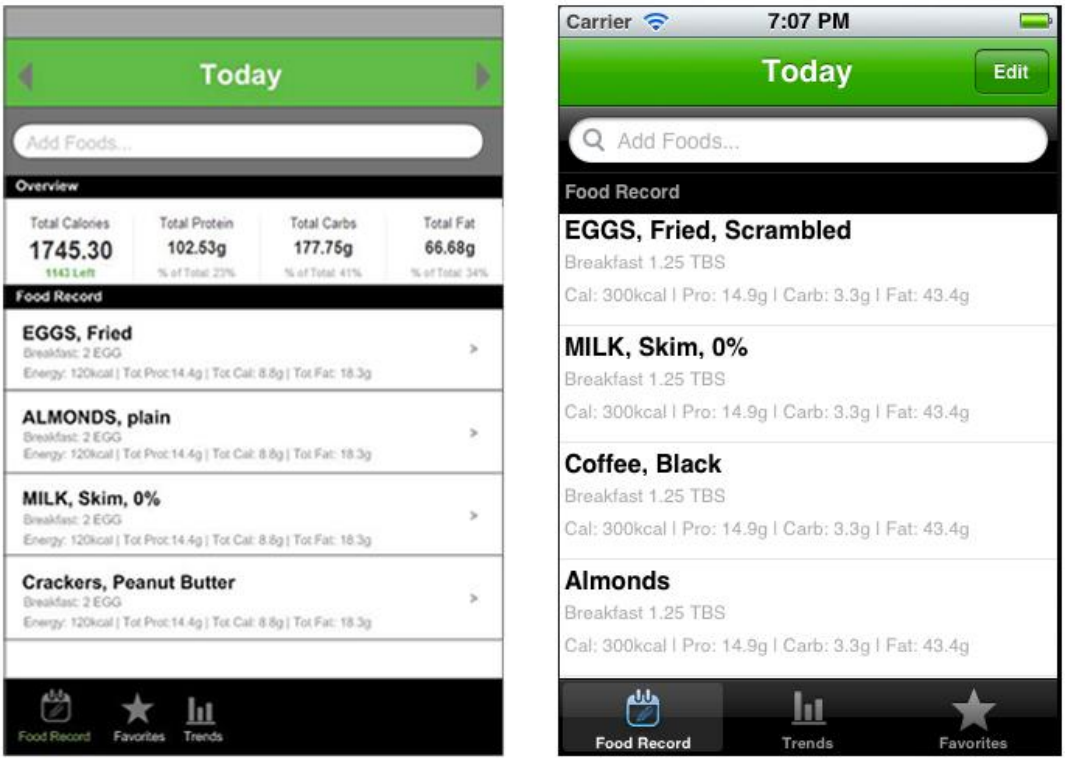


Figure 6: iPhone Prototype User Interface

2.5 Why HTML5?

DHM will be an HTML5[5] mobile application using jQuery Mobile, .NET MVC 4 framework[6] and Windows Communication Foundation[7] (WCF) services. The decision was not easy to move forward with an HTML5-based application versus going native. While it is certainly desirable to have an app on the Apple and Android App Stores, a mobile Web application must come first in order to retain an edge in business. DH considered many factors for this mobile strategy. One factor is that it would be significantly more expensive to develop, support, and maintain a code-base for each native platform. The complexity level and code duplication of going native was also an

[5] http://www.w3.org/TR/html5/
[6] http://www.asp.net/mvc
[7] http://msdn.microsoft.com/en-us/library/dd456779.aspx

obstacle since DH requires rapid development to keep up with competitors in this fast moving mobile world. Given DH's limited resources, a more sensible approach is to focus development on the code-base one time and have the ability to run the app on many device platforms. This would ensure that DH could serve as many customers as possible no matter what device they use. DH has already gained expertise in the HTML5, jQuery/JavaScript, .NET framework, and WCF technologies with the development and launch of DHW. The backend business services for DHW are well established and the front-end scripts are available to DHM for reference and reuse. Therefore, developing the HTML5 mobile application is the logical first step for DH and then in the future, target specific devices such as the iPhone or Android.

2.5.1 Survey of Technologies

Sencha Touch

Sencha is an HTML5 mobile application framework that enables development of cross-platform apps that run on all touch-based devices [11]. The platform offers native-like fluid animations, adaptive layouts, and smooth scrolling. A strong feature, in my opinion, of Sencha, is that it offers native packaging to "wrap" the web application into a native shell and deploy to the Apple or Android App Store giving access to native device features such as the camera.

jQuery Mobile

The jQuery Foundation is a member supported non-profit trade association for web developers whose mission is to advance the web through JavaScript [12]. JQuery Mobile is one of the foundation's open source projects [12]. It is an HTML5-based user interface system, built upon jQuery and jQuery UI (a set of user interface interactions, effects, widgets, and themes) foundation and has broad support for modern smartphones,

tablets, desktop, and e-reader platforms [12]. jQuery Mobile is frequently used in conjunction with PhoneGap/Cordova[8], allowing it to be packaged and run as a native application.

2.5.2 jQuery Mobile, A Step Towards Native

I decided to use the jQuery Mobile framework to build the mobile friendly web application. The main reason for using this framework is that I have experience in client-side programming with the jQuery JavaScript framework. Aside from my familiarity, the framework is open source, has a very good community support system, and a broad support of the vast majority of modern desktop, mobile, tablets, and e-reader platforms.

2.5.3 Concerns and Risks

Table 1 assesses some concerns and risks and their impact before project start. The risks I deemed that were most likely to occur were technical obstacles and underestimating project scope. I did indeed encounter a large learning curve of learning the .NET MVC framework and two JavaScript frameworks when first starting the project. I followed the mitigation strategy for this risk, which was to seek help on forums and study resources. I also encountered some performance issues related to the technology shortcomings risk. I think that the some of these performance lags were due to all work that jQuery Mobile does behind the scenes to make HTML content mobile and admittedly, my inexperience with these new technologies. I was able to speed up the performance of the application by minimizing JavaScript code and CSS. I also combined custom CSS styles into one external style sheet. Another performance tuning measure was to investigate and remove any bindings with Knockout[25] that were not necessary.

[8] http://phonegap.com/

For instance, the nutrition facts for a food does not change and it is not interactive and only for purposes of display. I removed the Knockout bindings for this case and other similar features, which improved performance.

Table 1: Risk Mitigation

1. Risk event description
2. Likelihood of risk event occurring (0.0 – 1.0)
3. Impact on project (0.0 – 1.0)
4. Risk factor is the product of likelihood and impact
5. Risk mitigation strategy

Risk	Likelihood(a)	Impact(b)	Risk factor (a * b)	Mitigation Strategy
Lack of skill-set; Technical obstacles/issues	.6	.7	.42	Study resources, white-papers, best practices; Seek help from forums; Notify committee members if issues could potentially effect project schedule;
Business operations require attention (i.e.) customer support or software bugs in existing products	.5	.6	.30	Devise strategy to devolve duties to ensure project success; work weekends/longer hours;
Technology shortcomings (ex: performance, unable to access native hardware, etc…)	.4	.9	.36	Research difficulties to transfer to existing code-base to new technology platform; Notify committee members with findings;
Customer(s) has feature request(s)	.3	.4	.12	Consider impact on scope to incorporate feature; Determine if feature can and should be added; Notify committee members if feature is desirable and should be considered; Adjust project schedule;
Project underestimated	.5	.8	.40	Adjust project schedule; Consider decreasing feature scope; Keep committee members aware of issues;
More time needed for final paper revisions or scheduling conflicts	.5	.7	.35	Ensure the schedule has slack time; Ensure that committee members have adequate time to review; Give adequate advanced notice;

Chapter 3: Methodology

3.1 Use Case Diagram

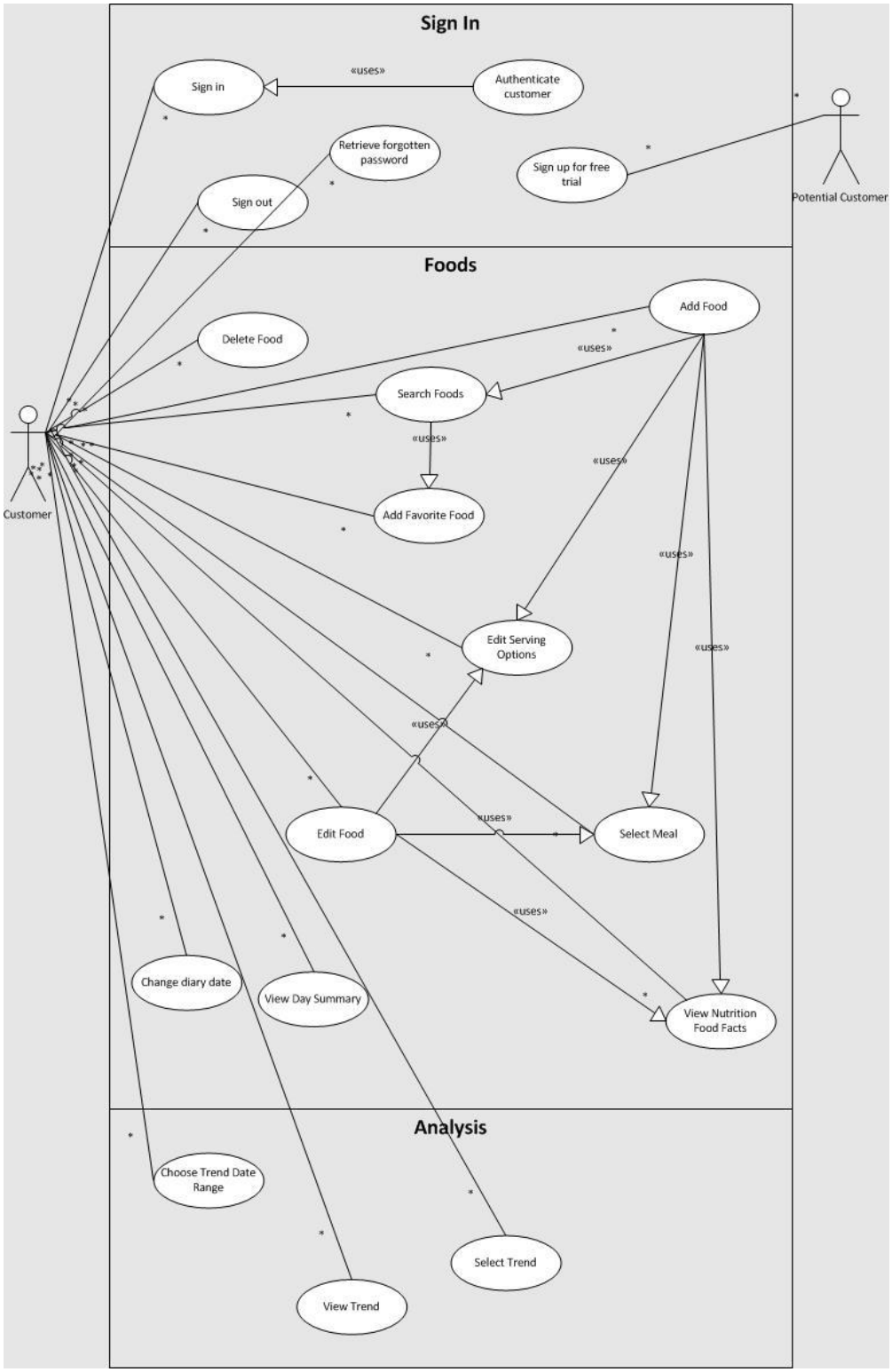


Figure 7: Customer Use Cases

3.2 User Interface Design

The customer's experience with our mobile application is without a doubt very important to us (DH). The app must be intuitive, easy-to-use, and responsive especially since our new customer focus is on older individuals with kidney disease. There are three main views for this application, the view to sign into the system, the view that contains the food diary and its associated screens like editing diary date and overview, and finally the view that contains the dietary analysis. Lucky for me, I have already designed and developed the user interface for DHW and gained experience in mobile UI with development the iOS prototype.

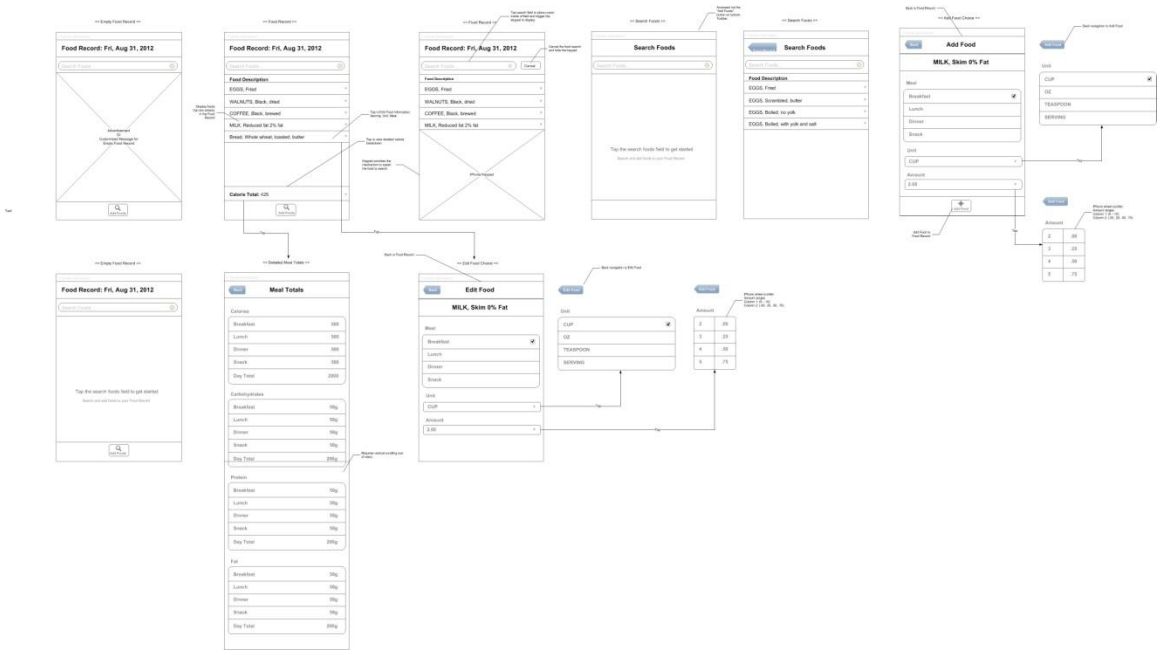


Figure 8: Early stage wire frames

3.2.1 Sign In View

The *Sign in* view is responsible for securely authenticating customer credentials (email address and password). Because this is mobile web application, I was concerned with requiring the customer to sign in each time and every time they wish to use the app.

I was also concerned with storing customer information like an email and password on the device for security sake. I was able to avoid the annoyance to our customers using Forms Authentication in ASP.NET [14]. With Forms Authentication, I was able to encrypt the customer's identity (email address and customer id) in an authentication token when they select the "Keep me signed in" option. As long as the customer does not "Sign out" of the system, any subsequent application requests will automatically retrieve and decrypt the token and authenticate the customer.

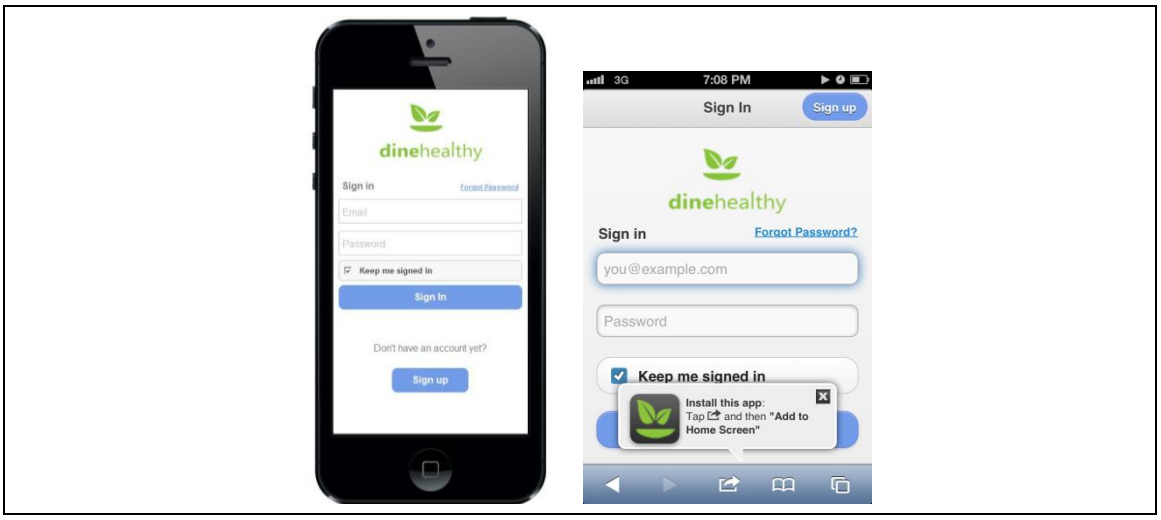


Figure 9: Sign in screen mock and actual interface

From the sign in view, I added a feature, not in the scope of this project that invites the user to download the application to the device's home screen or application menu. This invitation is on a floating div captured in the Figure 9, "Install this app". It gives the user three opportunities to add the app to the home screen. For supporting platforms, the option to add a shortcut is always available from the bookmarks menu on the device. This allows the user to open the application from an icon on their home

screen instead of typing a URL. This concept provides a more native-like feel when using mobile web applications.

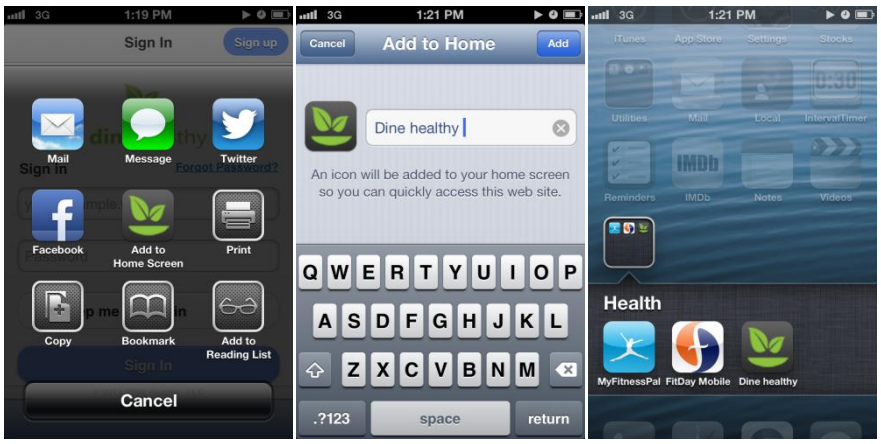


Figure 10: The iPhone 4 allows the user to add an icon to the home screen

On iOS only, it is possible to make the webapp even more native-like by forcing it take up the full screen of the device [15]. This option will remove the Safari controls including the address and toolbar.

3.2.2 Foods View

I believe it is safe to assume that customers will spend the majority of their time with the webapp on the food diary. I decided to model the process of adding a food after the MyFitnessPal application. The process of adding a food requires searching for the food and adding the food choice to the food diary. There are three options to modify for the food to add, the serving size, the serving unit (Ounces, Kilograms, Grams, etc.), and which meal food was eaten (Breakfast, Lunch, Dinner, or Snack). On DHW and DHD, these three options available are on the Add Food views. An issue our users have encountered is that they commonly forget to modify the meal selection and the food diary looks as though everything they ate was for breakfast (it is the default selection). Then

the user will have to edit and modify the meal for each incorrect food. I decided to tweak the process by placing the ability to select a meal before you even search for a food to add. This way you are less likely to forget this selection and you do not have to spend the extra time editing each food on the food diary that has an incorrect meal association. Some other added benefits to this are that there are fewer options to contend with on the Add Food view, it behaves more like a series of steps, and it saves some valuable real estate. I also modeled parts of our Food Search after MyFitnessPal search interface.

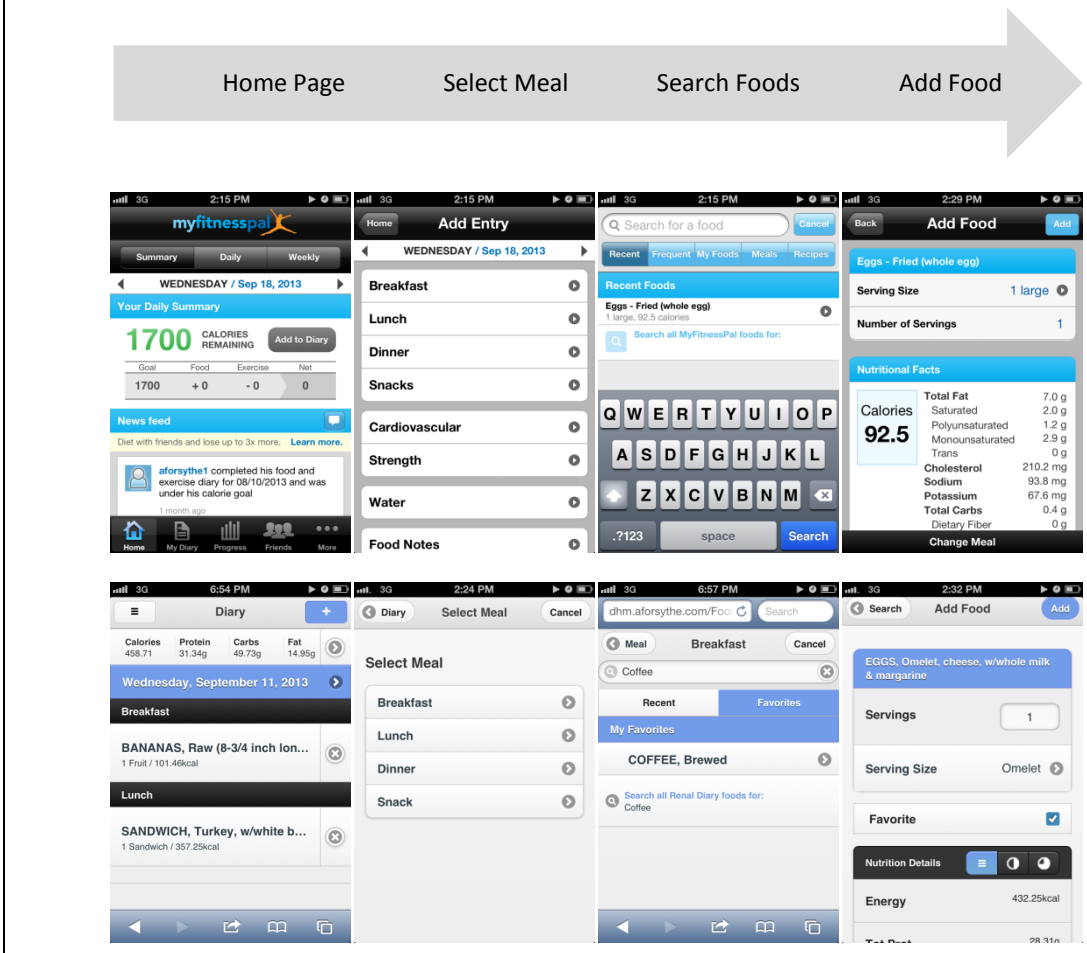


Figure 11: Side by side for the 'Add Food' process with MyFitnessPal and DHM

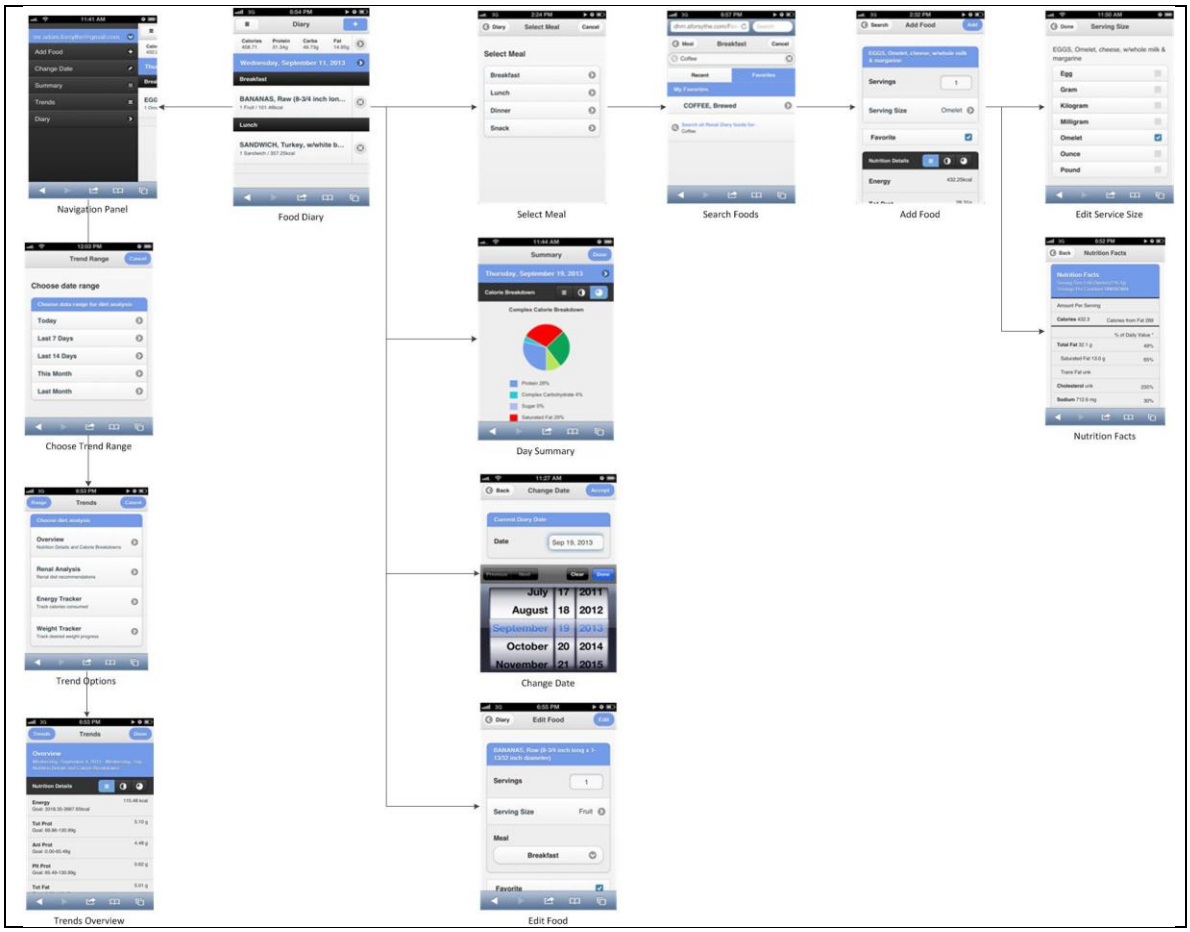


Figure 12: DHM view schematic

3.2.3 Trends View

Trends lets customers know how well or how poorly they are eating over time. For the scope of this project, only one trend was required. I implemented four trends; an overview of nutrition breakdowns, the renal analysis that details how well an individual ate within the dietary recommendations of the Renal (Kidney) diet, an Energy Tracker, and Weight Tracker.

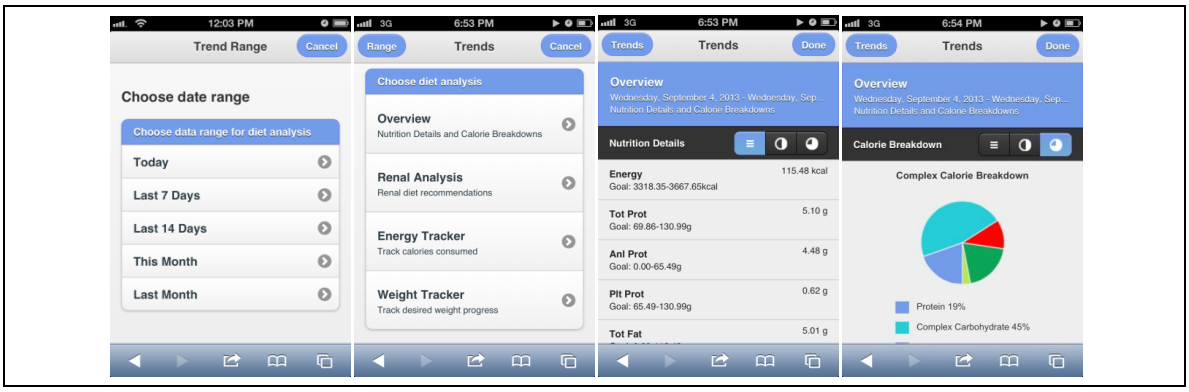


Figure 13: Trends screen views

3.3 List of Technologies

jQuery Mobile

jQuery Mobile is a unified, HTML5-based user interface system for popular mobile device platforms, built on rock-solid jQuery and jQuery UI foundation. Its lightweight code is built with progressive enhancement, and has a flexible, easily themeable design [12].

jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers [17].

jQuery UI (User Interface)

jQuery UI is a set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript library [18].

Ajax (Asynchronous JavaScript and XML)

Ajax is a technique for creating fast and dynamic web pages by allowing them to be updated asynchronously by exchanging data with a server behind the scenes. Ajax allows parts of web pages to be updated without reloading the entire page [19].

JSON (JavaScript Object Notation)

JSON is a data-interchange format that closely resembles parts of the JavaScript syntax. It is useful for any kind of JavaScript-based application. Ajax is used to transport the JSON payloads.

HTML5(Hypertext Markup Language)

HTML5 is the latest evolution of HTML standards. It has a larger set of technologies that allows more diverse and powerful web sites and applications [22].

DOM (Document Object Model)

The DOM is an API for manipulating HTML documents. It provides a structural tree representation of the document, enabling modification of content and visual presentation by using a scripting language such as JavaScript [23].

ASP.NET MVC (Model-View-Controller)

ASP.NET MVC is a framework for building web applications that applies the general Model View Controller pattern to the ASP.NET framework [24].

3.4 Software engineering paradigm for development

I used the Agile Software Development process, a development process that is iterative and incremental approach. The Manifesto for Agile Software Development is as follows:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
Individuals and interactions *over processes and tools*
Working software *over comprehensive documentation*
Customer collaboration *over contract negotiation*
Responding to change *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more. [26]

The goal of this project was to develop a minimally viable product for our customers with interactions we know are necessary. The application will have some given functionality like secure authentication, the ability to add, edit, delete food choices, and view diet analysis trends. Once the product is in a beta, the customers will have the opportunity to give us feedback and make the product their own. An iterative software development process is optimum for our small team. Being small allows us to be nimble in our product development allowing us to turn-on-a-dime and not be "bogged down" with a bunch of red tape you usually find in corporate environments. I broke down the projects tasks in digestible modules that make sense in the users workflow; Login, Foods (Add, Edit, and Delete), and Trends. This broke down the tasks in to about two to three week iterations. Each cycle of this project was functional, unit tested, and reported to this capstone project's committee. In addition, the committee was given access to the application in a test environment.

3.5 Test Plan

The test plan used for the project was .NET MVC unit tests, developer testing using Visual Studio Debug tools, manual test cases, and beta release. I quickly learned that the MVC unit testing for the business logic would not be as important as testing new source code on the front end. The backend for the application was for the most part, is intact and already tested. I did however build a few quick and simple MVC unit tests for the Controller responsible for the authentication and login. This work gave me some experience with this testing approach, but the majority of testing was on the JavaScript source code with developer/unit testing. I used mostly Mozilla Error Console, Firebug, and Google Chrome developer tools to test the custom scripts on the front end. I ran short

on time for using a JavaScript unit test framework. Regarding the beta release, this is on hold until we launch the main web application.

In lieu of creating manual test cases in Microsoft Word or using some similar test case template, I decided to use Team Foundation Server's Test Plan features. This is a great tool to build test cases and easily associate them to project under source control. There is a litany of testing features including the ability to build incremental manual steps to test an application, record pass/fail results, assign cases to team members, and associate with automated testing. Below are test cases I wrote for the "Sign In" use case and some functionality available (See figure 14).

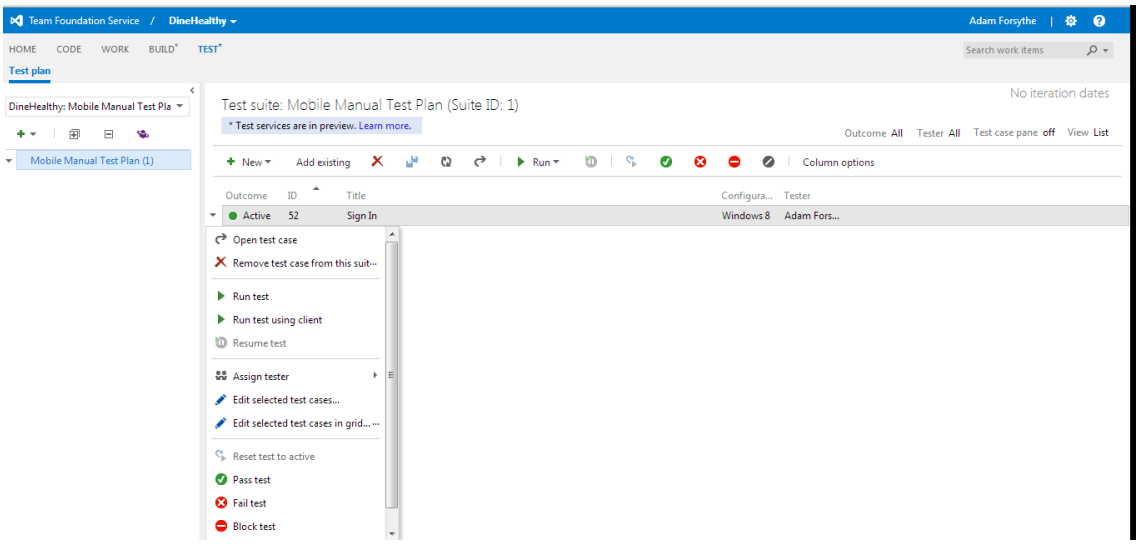


Figure 14: Microsoft Team Foundation Server, test view and features a test plan

Chapter 4: Implementation, Decisions, and Discussion

4.1 A New Product Focus on Kidney disease

The first major decision that I will mention is the change in market direction for Dine Healthy Web and this mobile friendly companion product. We decided to focus our attention on individuals who are in various stages of kidney disease and are currently in the process of revising the Dine Healthy Web product into a product called RenalDiary.com. Only cosmetic changes were necessary for the scope of this mobile project, (i.e.) a logo change and calling more attention to specific nutrients pertinent to individuals with kidney disease. However, once this product is on the market, more changes and customizations will take place to better accommodate these individuals and professionals that serve them.

4.2 Architectural Overview

Beginning with the presentation layer, this MVC 4 solution has three MVC Controllers (Login, Logout, and Foods). Login, of course, is responsible for authenticating the customer. All three controllers use a base controller which instantiates a reference to the service layer's services. This inherited base class also handles storing and retrieving the authentication cookie. The only page refresh or post back occurs when the customer logs into the application and when they log out. This is so I can clean up the forms authentication cookie stored on the client. Once authenticated, the Foods controller renders its View and all of the application's calls to the services that return JSON payloads. This project is mostly a single page application (SPA) with one view that contains several partial views that load as needed with the exception being the Login and Logout Views. Because the application's services are not yet public, I use the controller's

to funnel the calls with wrapper methods. The controller actions allow me to easily return JSON.

The backend layer is in a separate solution and contains the data context using Entity Framework[9], the domain models, and services. This project and the web application reference this backend solution. The models (Data-Transfer-Objects) have already been pared down to the essential data required for the consumer application. I used client side scripting in some cases to customize and format model properties or add new properties. As far as the presentation layer is concerned, the heavy lifting is done with JavaScript and the controller's main purpose is to provide the application with the data.

4.3 Learning a New JavaScript Framework - Knockout Overview

jQuery Mobile was used to build the user interface for this project. I decided to use the Knockout JavaScript framework to handle declarative bindings and automatic user interface refreshing. In DINE Healthy Web, I wrote custom JavaScript bindings, which associated DOM (Document Object Model) elements with model data and custom functions to refresh the interface. Knockout simplifies this process and allows for better separation of code using the MVVM (Model-View-View Model) user interface design pattern [25]. This pattern expands the MVC (Model-View-Control) paradigm to separate the JavaScript from the view (i.e.), there is no scripting in the HTML. MVVM is a design pattern is split into three parts, the Model, the View, and the View Model. The Model is where the business logic resides; mainly server side programming that is independent of the user interface (UI). The View Model is pure JavaScript business logic, which is a pure-scripting representation of the View. The View Model has no knowledge of HTML

[9] http://msdn.microsoft.com/en-us/data/ef.aspx

and is not concerned with trivial actions such styling elements. The View Model helps to keep the clutter down in your JavaScript and allows you to stay focused on the more sophisticated behaviors of the application. The View is the visible part of the application. It represents the state of the View Model. The View displays information from the View Model and when the user makes changes such as changing a selection on dropdown form element, the View Model is updated. The View is the HTML document that has the Knockout (KO) declarative bindings to the View Model (See Figure 14)..

```
// declarative bindings

// this is the view model
var viewModel = {
    customerName = “John”
};
ko.applyBindings(viewModel); // this activates Knockout

// the is the view
The customer’s name is: <input type=”text” data-bind=”value: customerName”/>.

// this would be the same as writing
The customer’s name is: <input type=”text” value=”John”/>.
```

Figure 15: Knockout.js declarative 2-way bindings

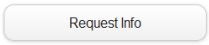
In this example, when the View is displayed, a text field with the name “John” will be shown. “John” is the default value of *customerName* variable. If the user decides to change the value of the text field, the *viewModel* object will automatically update itself [25]. This automatic, underlying two-way declarative binding helps you to avoid rolling your own handcrafted binding.

4.4 jQuery Mobile DOM Exploitations

It is important to understand what jQuery Mobile does under the covers to make it mobile friendly, especially when working with dynamic UI content. The best way I can describe this mobilization process of jQuery Mobile is to demonstrate it using Firebug, a development toolset that allows you edit, debug, and monitor CSS, HTML, and

JavaScript live on any web page. The figure below depicts a simple button and the behind the scenes UI component DOM manipulation of jQuery Mobile.

View: what the user sees



HTML

```
<a data-role="button" href="mailto:barnhillk@uncw.edu?&subject=MSCSIS: Please send me more information!">Request Info</a>
```

jQuery Mobile manipulation

```
<a class="ui-btn ui-shadow ui-btn-corner-all ui-btn-up-c" href="mailto:barnhillk@uncw.edu?&subject=MSCSIS: Please send me more
information!" data-role="button" data-corners="true" data-shadow="true" data-iconshadow="true" data-wrapperels="span" d
ata-theme="c">
  <span class="ui-btn-inner">
      <span class="ui-btn-text">Request Info</span>
    </span>
</a>
```

Figure 16: The display, HTML, and DOM for how jQuery Mobile handles mobilizing webapps

jQuery Mobile adds wrapper elements and CSS classes to style the button and make the button mobile friendly. A simple one-line button declaration was transformed into three elements, a wrapping anchor element with two span elements with specific jQuery Mobile CSS classes. Similar transformations occur on other UI components including lists, toolbars, form components, and other formatting components. HTML5 data attributes are used to notify the jQuery Mobile framework to transform UI components to mobile.

4.5 Applying JavaScript Design Patterns

Another concept that I explored in this project was how to better organize using JavaScript design patterns. I already had some basics concepts under my belt like using external JavaScript files, minimizing code for performance, and the use of JavaScript object literal notation. I also always run my JavaScript code through an online tool called

JSHint[10] to help spot potential errors and use best practices. The design pattern that made the most sense to me with regards to this application was the module pattern. This pattern emulates the concept of classes where you are able to include both public/private methods and variables inside a single object and keep particular parts out of the global scope [16]. JavaScript lends itself to some bad habits and bad design practices if you are not watchful or aware [16]. For instance, declaring variables in the global context pollutes the global namespace, using inline JavaScript in the view is inflexible, passing strings instead of functions to setTimeout. The module pattern helps to keep the code clean and organized with similar encapsulation found in other languages like Java and C#. The figure below represents this pattern.

```
var myNamespace = (function () {

        var myPrivateVar = 0;

        var myPrivateMethod = function (someText) {
                console.log(someText);
        };

        return {
                myPublicVar: "foo",
                myPublicFunction: function (bar) {
                        myPrivateVar++;
                        myPrivateMethod(bar);
                }
        };
})();

// From Learning JavaScript Design Patterns, O'REILLY, Addy Osmani
```

Figure 17: Pseudo-code for JavaScript Module Pattern

4.6 Specific Issues Encountered

Fortunately, most of the issues that I encountered happened early on in the development process. I was able to hit my stride about middle-way through this project. I think that these issues came about while juggling three new technologies at the same time.

[10] http://www.jshint.com

4.6.1 Making jQuery Mobile and Knockout Play Nice

One of the biggest challenges I faced in this project was making the jQuery Mobile and Knockout frameworks work together. This application required so much dynamic content, a good portion of my project time was spent on making the view content actually refresh and reflect actual changes. Before I started this project, I had no experience with .NET MVC, jQuery Mobile, and the Knockout framework. Therefore, I had a sizeable learning curve along with this issue of dynamic content refresh. The issue boils down to what I touched on in a previous section about jQuery Mobile's DOM manipulation. What happens is that you have server side data coming in from an Ajax request and you notify the UI that you now want to display this new data, but jQuery Mobile does not reflect the new changes. This issue causes no errors or warning messages, it just does not work. It all comes down to the timing of when, in the DOM loading timeframe, jQuery Mobile transforms the underlying HTML. jQuery Mobile must be notified to refresh its transformations after the dynamic content is sent. To resolve this issue, I had to follow these steps:

1. From the KO view model, make a request server-side data for dynamic content (For example, retrieve a list of foods)
2. Consume the returned data into the KO view model, which should automatically do the two-way binding where the view should already reflect the new data coming in, but does not with jQuery Mobile
3. Since jQuery Mobile does not reflect the new changes, you must notify jQuery Mobile that there are changes that are waiting in the DOM, which is accomplished

by calling a refresh the UI component. With KO, use custom bindings to notify that changes are ready to consume.

With KO, there is the concept of custom bindings where you can roll your own bindings. This is the best approach from anecdotal Internet searches for answers to the issue. Using KO's custom bindings allows you create a listener for whenever something changes with a data binding and refresh the jQuery Mobile UI content. One thing to note, this issue is not really a problem so much with KO; it is a nuisance anytime you are dynamically refreshing UI content with jQuery Mobile even when building custom declarative bindings.

Another challenge I faced with the two frameworks was the organization of my views and view models. Regarding the views of the project, it has only three HTML pages. One page for login that handles the authentication, one page for logout, and one page which is a multipage document that contains all of the views for searching, foods CRUD, and analysis. jQuery Mobile supports multipage documents, a page architecture where you can nest multiple pages each with a unique identifier. The figure below illustrates the organization pattern that I decided on.

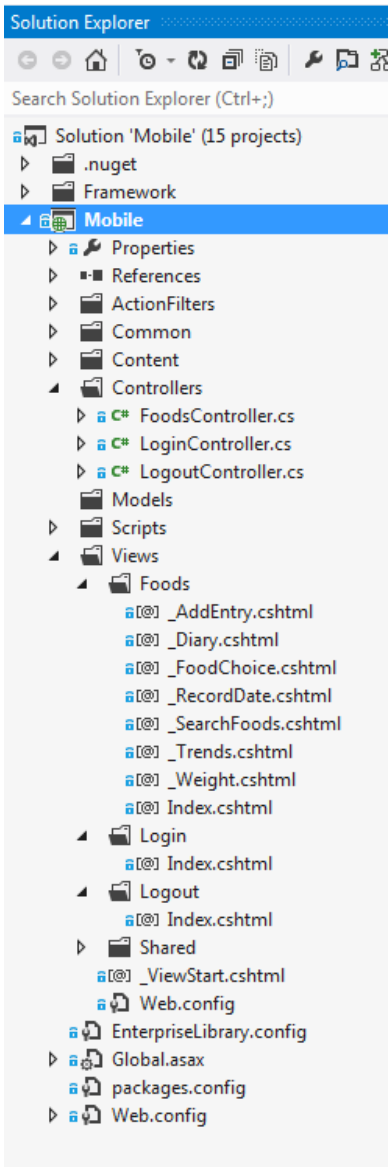


Figure 18: Screen capture of project organization from within Visual Studio

Microsoft's .NET Razor View Engine[11] allows you to organize your Views. There are three Controllers Foods, Login, and Logout with three corresponding Views. I used Razor's Partial Views to further organize the code into smaller, separate pieces. For instance, the Foods View contains seven Partial Views warehousing the different jQuery Mobile pages. Moreover, many of the Partial Views included multipage documents. For

[11] http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine

instance *_Diary.chtml* included jQuery Mobile page for the food diary and a page for an overview analysis of the day. This setup created some complexity with setting up my KO view models. It took a few passes to establish a solution for the multipage document. My first attempt was to create one view model for each view in the application, but this resulted in a number of issues and just plain odd behavior with the KO bindings. The reasons for these issues and behaviors was due to the fact that KO requires all of the view models observables, the variables that KO keeps track of, be instantiated before the view renders the bindings. Using a jQuery Mobile's multipage architecture resulted in these binding errors because all of the pages are loaded into the DOM whether the user sees them or not. This arrangement exposed KO observables in the view before my view model for the unseen pages were declared, thus throwing JavaScript errors. To give a better example, the first view the user sees displays a list of foods eaten for the day, the view model for this particular view is declared and instantiated. However, all of the views are actually downloaded completely, but not all of my view models are declared. Having a view model for each view was close; the solution was to use one view model that hosted the nested pages view models. Having a 'master' view model and an initializing constructor in each of the view models allowed me better stage when declarations are happening. The pseudo-code below illustrates this concept.

```
var masterViewModel = {
        foodsVM: null,
        searchFoodsVM: null,
        trendsVM: null,
};

// javascript class objects
function FoodsViewModel(){};
function SearchFoodsViewModel(){};
function TrendsViewModel(){};

// on init
var init = function() {
viewModelWrapper.foodsVM = new FoodsViewModel();
viewModelWrapper.searchFoodsVM = new SearchFoodsViewModel();
```

```
viewModelWrapper.trendsVM = new TrendsViewModel();
};
```

Figure 19: Pseudo-code for handling multipage view model declarations using Knockout

4.6.2 Sometimes Native is Just Better

Another matter of contention was deciding on the best way to accept user input for the number of servings on a given food choice. The problem is that using a text field creates to usability issues. For example, the user could tap on the wrong side of the default serving amount and have difficulties clearing the value and the default keyboard varies from device to device. With HTML5, you can specify a number type, which helps render the number keypad for a device, but iPhone displays the following keypad in figure below and Android only displays the "ABC" keyboard.

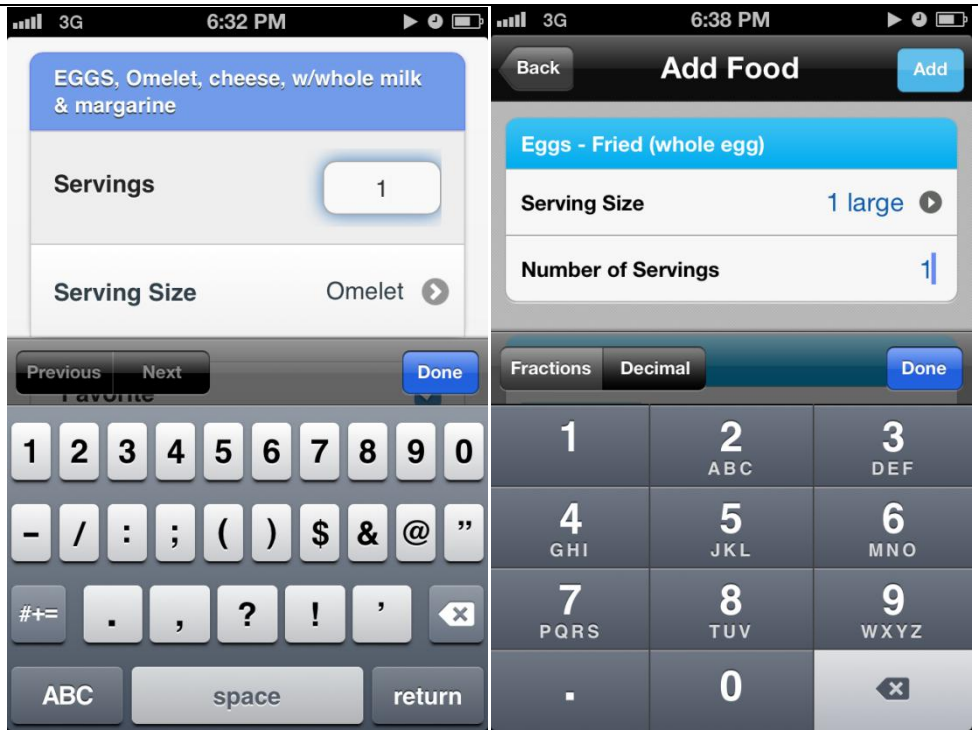


Figure 20: Side by side of mobile webapp and native app. Mobile webapps are stuck with less than optimal keypad

You can also specify a regular expression that guards against dirty input and formats the number how you want. In this case, I want a number formatted with two

decimal places (xx.xx) and render the iPhone keypad that the native app uses in Figure 20. Unfortunately, this is not supported for mobile webapps. Mobile webapps are stuck with the multi-character/number keypad. I also attempted to auto-focus and select all when the page is rendered, but this resulted in the iPhone's tooltip that gives you the option to "Copy", "Select", or "Select All". I thought that this would be confusing so I removed the script. A number picker was not optimal either; our application can accept large servings like 10,000.00 mg of Fish Oil. Scrolling a number picker would not be acceptable and would take forever (too long) to get to those large serving amounts. I stayed with the number-picker input text field and used regular expression to establish the pattern. I also added some padding to the right side of the field to give the user a better chance to tap on the right side of the serving to help in the case that the user wants to delete the default serving and add their own. I guess I will just have to keep my eye out when this is supported on iPhone.

Chapter 5: Conclusion

5.1 Deliverables

Below are the original deliverables of the project along with a short description of how these objectives were met.

Company information and background

I provided the history and innovations of our company before co-founding DINE Healthy, LLC. I also provided peek into future ventures.

Why an HTML5 mobile application was chosen over native application

There were three primary reasons for this decision. One it was cost effective for us to build a mobile friendly application versus a native application. Second, we wanted to avoid having to repeat source code for each popular device (iPhone and Android). Third, we could only support one device at this time with our small development team.

Previous works in web and mobile development

I designed an iPhone prototype that was shelved to pursue the mobile web application. This was my introduction into mobile development.

Software tools, and source control

The main tools I used to create this application were Visual Studio 2012[12], .NET/C#/MVC, SQL Server[13], and Microsoft Azure[14]. The source control for the first half of the project was Project Locker[15] and then we were a Microsoft Bizspark[16] account that provided us with Team Foundation Server (TFS)[17].

Market and competition analysis

---

[12] http://www.microsoft.com/visualstudio/eng/visual-studio-2013
[13] http://www.microsoft.com/en-us/sqlserver/default.aspx
[14] http://www.windowsazure.com/en-us/
[15] http://projectlocker.com/
[16] http://www.microsoft.com/bizspark/
[17] http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx

I mentioned three competitors that offered similar services to ours. Now that our focus has changed, FitDay would be the most direct competitor because they are offering a product that allows you connect with professionals via online messaging. There focus is on individuals seeking to lose weight, whereas our new product geared toward individuals in various stages of kidney disease and the kidney-care providers/professionals that serve them.

Technologies surveyed and technologies used to build the application

I surveyed various mobile frameworks and decided on using jQuery Mobile for reasons already stated. I learned a new JavaScript framework in the process.

Software engineering paradigm for development

An iterative software development process was used to build this application. Being "Agile" allowed us to change direction and tailor a solution for a new product.

Software design and analysis artifacts such as wire frames, use cases, project management documents including WBS and Gantt charts

I have provided artifacts throughout this paper and in the Appendix.

Testing plan and implementation to release the application to BETA

We have gathered emails for potential customers for our new product. The product is functionally ready for early access, but the release of the web application portion is still being developed. Once released, this mobile companion will be release in beta. I ran unit tests and developed/executed manual test case documentation stated in the Test Plan using Microsoft Team Foundation and Microsoft Test Manager.

All tasks were completed in the Project Work Breakdown Schedule (Appendix A) except the beta release described above. It was required that I only add one dietary trend;

I was able to add four trends (Overview, Renal Analysis, Energy, and Weight tracking). I also added the ability to bookmark the webapp to a device's home screen.

5.2 Lessons Learned

I have a rather bad habit of jumping into a new technology before learning as much as I can before writing code. In regards to this project, I did read and work through examples from "jQuery Mobile Up and Running" by Maximiliano Firtman and watched a number of tutorials from PluralSight[18] on .NET/MVC framework, but I did not take enough time up front to study the Knockout framework. This lead to some of the headaches mentioned in previous sections. Instead of backing up and taking the time to review the framework, I ended up wasting time on trial and error, 'hope this works' coding. It can be time convenient to copy / paste code from a help forum like Stack Overflow[19] and not fully grasp the concept of what you are solving but in the end, this will haunt you.

5.3 What would I do different

As far as technology is concerned, I think that all were the right choice for the product. For the client-side frameworks, jQuery Mobile is easy to use and handles behind the scenes magic to go mobile and Knockout is a real time-saving framework that helps to build clean and organized JavaScript and has the MVVM concept [25]. I'd much rather work in .NET MVC than .NET Web Forms because it is better organized, the page life cycle is clean, and the framework works better for web applications that have dynamic UI content. One thing I would do different is spend more time learning Google Chrome[20]

---

[18] http://www.pluralsight.com/training
[19] http://stackoverflow.com/
[20] https://developers.google.com/chrome-developer-tools/

and Firebug[21] JavaScript debugging tools. These tools allow you to step through your JavaScript and give you a peek inside to what is happening. Usually, I will use the Firefox browser's Error Console to debug scripts because it lets me know instantly that something is wrong with the script and where the problem is. I started to work with those tools in this project but ended up having to use my old standby.

I would also organize the client side scripts by separating certain modules into their own JavaScript files using a file and module loader framework called RequireJS[22] . Currently, all of the application's client side coding is located in one JavaScript file. I organize the scripts this way for optimization reasons. However, having all of the application's source code in one place makes it difficult to manage, organize, and debug. I would much rather create a file for each KO view model or particular module and separate the data service calls into their own classes and files.

5.4 Future Work

There is a lengthy feature list, which will make this minimal viable product even more valuable. One upcoming feature is the ability to create and manage your Profile information (age, gender, ideal caloric level, activity levels, goal weight, and conditions) from the mobile device. Another feature is the ability to add "Recent" foods to your food diary. This functionality would not rely on backend web services, but instead use HTML5 local storage[23]. Local storage can also be used to save the state of where the user is in the application. For instance, if there user closes the application while in middle of adding a food; this is where the application should pick up if the user reopens the application. Eventually, we will need native solutions for this application. One way that

[21] https://getfirebug.com/whatisfirebug
[22] http://requirejs.org/
[23] http://diveintohtml5.info/storage.html

DH could go native is to use PhoneGap, a technology that allows you to build applications using HTML, CSS, and JavaScript. Outside of new features, I would like to invest some time in some other JavaScript frameworks like Backbone.js[24] and Node.js[25].

To conclude, this was an excellent project that allowed me to apply skills attained from the Masters of Science in Computer Science and Information Systems and in my previous software development experience. I was able to explore new terrain, pick up new skills, and work through several complications all while fulfilling a business need for my own business.

[24] http://backbonejs.org/
[25] http://nodejs.org/

References

[1] Retrieved 4 2013, April from Centers for Disease Control and Prevention: http://www.cdc.gov/nchs/data_access/Vitalstatsonline.htm: CDC. http://www.cdc.gov/

[2] Retrieved 4 2013, April from Centers for Disease Control and Prevention: http://www.cdc.gov/diabetes/pubs/pdf/ndfs_2011.pdf: CDC. http://www.cdc.gov/

[3] Retrieved 4 2013, April from Centers for Disease Control and Prevention: http://www.cdc.gov/obesity/adult/causes/index.html: CDC. http://www.cdc.gov/

[4] Cynthia L. Ogden, Ph.D., Margaret D. Carroll, M.S.P.H., Brian K. Kit, M.D., M.P.H., & Katherine M. Flegal, Ph.D. (2012). *Prevalence of Obesity in the United States, 2009–2010*

[5] DINE Healthy, LLC (2013, April 4). DINE Healthy, LLC, Copyright 2013 DINE Healthy, LLC. All rights reserved. Retrieved 4 2013, April from DINE Healthy, LLC: http://www.dinehealthy.com/About.aspx

[6] FitDay. (2013, April 4). FitDay, Copyright 2000-2011 Internet Brands, Inc All rights reserved. Retrieved 4 2013, April from FitDay: http://www.fitday.com/.

[7] MyFoodDiary. (2013, April 4). MyFoodDiary, Copyright 2013 MyFoodDiary, Inc, All rights reserved. Retrieved 4 2013, April from MyFoodDiary: http://www.myfooddiary.com/.

[8] MyFitnessPal. (2013, April 4). MyFitnessPal, Copyright 2005-2012 MyFitnessPal, LLC All rights reserved. Retrieved 4 2013, April from MyFitnessPal: http://www.myfitnesspal.com/.

[9] *UITabBarController Class Reference.* (2013, April 4). Retrieved from http://developer.apple.com/library/ios/#documentation/uikit/reference/UITabBarController_Class/Reference/Reference.html

[10] *UINavigationController Class Reference.* (2013, April 4). Retrieved from http://developer.apple.com/library/ios/#documentation/uikit/reference/UITabBarController_Class/Reference/Reference.html

[11] Sencha. (2013, April 4). Sencha, Copyright 2013 Sencha Inc. All rights reserved. Retrieved 4 2013, April from Sencha: http://www.sencha.com/.

[12] jQuery Mobile. (2013, April 4). jQuery Mobile, Copyright 2013 The jQuery Foundation. All rights reserved. Retrieved 4 2013, April from jQuery Mobile: http://jquerymobile.com/.

[13] Facebook Developers. (2013, September 23). Facebook, Copyright 2013, All rights reserved. Retrieved 9 2013, September from Facebook Developers: https://developers.facebook.com/docs/reference/dialogs/oauth/.

[14] ASP.NET Forms Authentication Overview. (2013, September 23). Microsoft, Copyright 2013, All rights reserved. Retrieved 9 2013, September from Microsoft: http://msdn.microsoft.com/en-us/library/7t6b43z4.aspx.

[15] Firtman, Maximiliano R. *jQuery Mobile: Up and Running*. Sebastopol, CA: O'Reilly Media, 2012. Print.

[16] Osmani, Addy. *Learning JavaScript Design Patterns*. Sebastopol, CA: O'Reilly Media, 2012. Print.

[17] jQuery (2013, September 23). *jQuery Write Less, Do More*, Copyright 2013 The jQuery Foundation. All rights reserved. Retrieved 9 2013, September from jQuery: http://jquery.com/.

[18] jQuery User Interface (2013, September 23). *jQuery User Interface*, Copyright 2013 The jQuery Foundation. All rights reserved. Retrieved 9 2013, September from jQuery UI: http://jqueryui.com/.

[19] WC3 Schools (2013, September 23). *Ajax Introduction* , Copyright 1999-2013 Refsnes Data. All rights reserved. Retrieved 9 2013, September from WC3Schools: http://www.w3schools.com/ajax/ajax_intro.asp.

[20] Mozilla Developer Network (2013, September 23). *JSON*, Copyright 1999-2013 Mozilla Developer Network and individual contributors. All rights reserved. Retrieved 9 2013, September from Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/JSON.

[21] Microsoft Developer Network (2013, September 23). *Building Modern Mobile Web Apps*, Copyright 2013 Microsoft. All rights reserved. Retrieved 9 2013, September from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/hh994907.aspx.

[22] Mozilla Developer Network (2013, September 23). *HTML5*, Copyright 1999-2013 Mozilla Developer Network and individual contributors. All rights reserved. Retrieved 9 2013, September from Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5.

[23] Mozilla Developer Network (2013, September 23). *DOM*, Copyright 1999-2013 Mozilla Developer Network and individual contributors. All rights reserved. Retrieved 9 2013, September from Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/DOM.

[24] Galloway, Jon. Professional ASP.NET MVC 3. Indianapolis, IN: John Wiley & Sons, 2011. Print.

[25] Knockout. (2013, September 24). Knockoutjs.com, Copyright 2013 All rights reserved. Retrieved 24 2013, September from Knockout: http://knockoutjs.com/

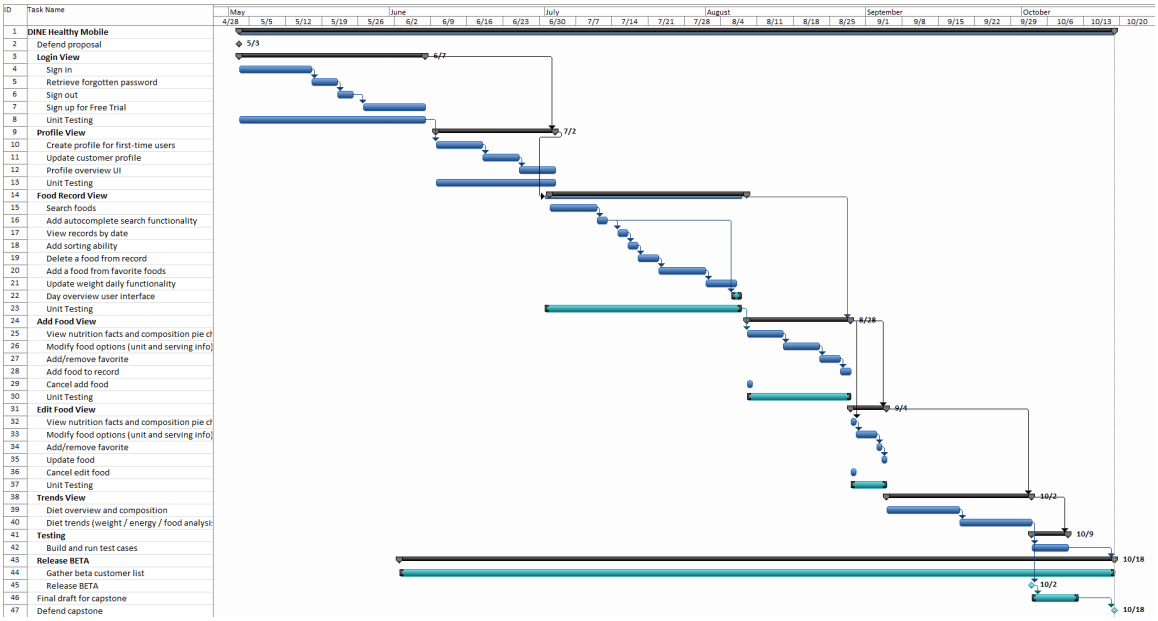
[26] Manifesto for Agile Software Development. (2013, September 24). Copyright 2001 All rights reserved. Retrieved 24 2013, September from Manifesto for Agile Software Development: http://agilemanifesto.org/

Appendices

Appendix A: Project Schedule

Task Name	Duration	Start	Finish	Predecessors
DINE Healthy Mobile	**121 days**	**Fri 5/3/13**	**Fri 10/18/13**	
Defend proposal	0 days	Fri 5/3/13	Fri 5/3/13	
Login View	**26 days**	**Fri 5/3/13**	**Fri 6/7/13**	
Sign in	10 days	Fri 5/3/13	Thu 5/16/13	
Retrieve forgotten password	3 days	Fri 5/17/13	Tue 5/21/13	4
Sign out	3 days	Wed 5/22/13	Fri 5/24/13	5
Sign up for Free Trial	10 days	Mon 5/27/13	Fri 6/7/13	6
Unit Testing	26 days	Fri 5/3/13	Fri 6/7/13	
~~Profile View~~	~~17 days~~	~~Mon 6/10/13~~	~~Tue 7/2/13~~	~~3~~
~~Create profile for first-time users~~	~~7 days~~	~~Mon 6/10/13~~	~~Tue 6/18/13~~	~~8~~
~~Update customer profile~~	~~5 days~~	~~Wed 6/19/13~~	~~Tue 6/25/13~~	~~10~~
~~Profile overview UI~~	~~5 days~~	~~Wed 6/26/13~~	~~Tue 7/2/13~~	~~11~~
~~Unit Testing~~	~~17 days~~	~~Mon 6/10/13~~	~~Tue 7/2/13~~	
Food Record View	**28 days**	**Mon 6/10/13**	**Wed 7/17/13**	**8**
Search foods	7 days	Mon 6/10/13	Tue 6/18/13	
Add autocomplete search functionality	2 days	Wed 6/19/13	Thu 6/20/13	15
View records by date	2 days	Fri 6/21/13	Mon 6/24/13	16
Add sorting ability	2 days	Tue 6/25/13	Wed 6/26/13	17
Delete a food from record	2 days	Thu 6/27/13	Fri 6/28/13	18
Add a food from favorite foods	7 days	Mon 7/1/13	Tue 7/9/13	19
Update weight daily functionality	4 days	Wed 7/10/13	Mon 7/15/13	20
Day overview user interface	2 days	Fri 6/21/13	Mon 6/24/13	16
Unit Testing	28 days	Mon 6/10/13	Wed 7/17/13	8
Add Food View	**15 days**	**Thu 7/18/13**	**Wed 8/7/13**	**23**
View nutrition facts and composition pie chart	5 days	Thu 7/18/13	Wed 7/24/13	23
Modify food options (unit and serving info)	5 days	Thu 7/25/13	Wed 7/31/13	25
Add/remove favorite	2 days	Thu 8/1/13	Fri 8/2/13	26
Add food to record	2 days	Mon 8/5/13	Tue 8/6/13	27
Cancel add food	1 day	Wed 8/7/13	Wed 8/7/13	28
Unit Testing	14 days	Thu 7/18/13	Tue 8/6/13	23
Edit Food View	**6 days**	**Wed 8/7/13**	**Wed 8/14/13**	**30**
View nutrition facts and composition pie chart	1 day	Wed 8/7/13	Wed 8/7/13	30
Modify food options (unit and serving info)	2 days	Thu 8/8/13	Fri 8/9/13	32
Add/remove favorite	1 day	Mon 8/12/13	Mon 8/12/13	33
Update food	1 day	Tue 8/13/13	Tue 8/13/13	34
Cancel edit food	1 day	Wed 8/14/13	Wed 8/14/13	35
Unit Testing	5 days	Wed 8/7/13	Tue 8/13/13	30
Trends View	**10 days**	**Thu 8/15/13**	**Wed 8/28/13**	**31**
Implement on diet analysis	10 days	Thu 8/15/13	Wed 8/28/13	37
~~Diet trends (weight / energy / food analysis)~~	~~10 days~~	~~Thu 8/29/13~~	~~Wed 9/11/13~~	~~39~~
Testing	**5 days**	**Thu 8/29/13**	**Wed 9/4/13**	**39**
Build and run test cases	5 days	Thu 8/29/13	Wed 9/4/13	39
Release BETA	**69 days**	**Mon 6/3/13**	**Thu 9/5/13**	**42**
Gather beta customer list	69 days	Mon 6/3/13	Thu 9/5/13	
Release BETA	1 day	Thu 9/5/13	Thu 9/5/13	42
Final draft for capstone	7 days	Fri 9/6/13	Mon 9/16/13	45
Defend capstone	5 days	Tue 9/17/13	Mon 9/23/13	46

Appendix B: Gantt Chart



Appendix C: Use Cases

<table>
<tr><td>Use Case Name</td><td colspan="2">Sign In</td></tr>
<tr><td>Scenario</td><td colspan="2">User signs into system</td></tr>
<tr><td>Brief Description</td><td colspan="2">A user gains access to the application by signing in with their already established username and password. The user can choose to keep logged in.</td></tr>
<tr><td>Actors</td><td colspan="2">Registered and non-registered users</td></tr>
<tr><td>Related Use Cases</td><td colspan="2">Sign Out</td></tr>
<tr><td>Preconditions</td><td colspan="2">In order to sign in the user must have already established their username/password by signing up for free trial or purchasing subscription</td></tr>
<tr><td>Post conditions</td><td colspan="2">The customer is authenticated and is forwarded to the food diary view</td></tr>
<tr><td rowspan="2">Flow of Events</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Customer goes to sign in view

3. Customer enters valid credentials using the sign in form</td><td>2. The system determines if customer has auto-login by locating and decrypting cookie

2.1. If customer is auto-login, the system will auto-sign in and display the food diary view

4. The system will determine if the user credentials are valid, profile has been created, and subscription is not expired

4.1 The system will create a new authentication cookie that is either persistent or non-persistent

4.2 Display food diary view</td></tr>
<tr><td>Exception Conditions</td><td colspan="2">1. If customer enters invalid credentials
a. Error notifications are given for unsuccessful login
2. If customer’s subscription or free trial is expired
a. Notify customer that there subscription has expired and provide a link to renew their subscription
3. If customer has forgotten their password
a. A link is provided to Retrieve their password on corporate webpage
4. If customer does not have an account
a. A link is provided for customer to create an account and profile on the main web application
5. If customer’s persistent cookie is invalid or has been deleted
a. Redirect customer to sign in page
6. If customer has a valid subscription, but does not have a Profile set up
a. Redirect customer to an error page and notify customer on what to do</td></tr>
</table>

Use Case Name	Sign Out	
Scenario	User signs out of system	
Brief Description	A user signs out of the application; the system cleans up login data	
Actors	Registered customers	
Related Use Cases	Sign Out	
Preconditions	In order to sign out, you must first be signed into the system	
Post conditions	The sign out view is displayed with a message of successful sign out; authentication data is removed	
Flow of Events	**Actor**	**System**
	1. User chooses to sign out of application	2. The system performs authentication clean ups (Forms Authentication SignOut, removal of cookie data) 2.1. Display sign out view with message of successful sign out
Exception Conditions	Users closes application without Sign Out – auto-login is still available	

Use Case Name	Sign Up	
Scenario	User signs up for free trial	
Brief Description	For user's that have not participated in our free, they can sign up for free trial from the desktop application	
Actors	Non-Registered customers	
Related Use Cases	Sign In	
Preconditions	User must be on sign in screen	
Post conditions	The user is directed to the main desktop application on sign up screen	
Flow of Events	**Actor**	**System**
	1. A user is not subscribed and clicks the "Sign Up" button	2. The system opens the main app on the sign up screen
Exception Conditions	User has already free-trialed, the main desktop app will handle	

Use Case Name	Retrieve Forgotten Password	
Scenario	A subscribed customer forgets their password	
Brief Description	A subscribed customer has forgotten their password, they can retrieve from the main corporate website; a button is provided on the mobile webapp	
Actors	Subscribed customers	
Related Use Cases	Sign In	
Preconditions	User must be on sign in screen	
Post conditions	The user is directed to the forgot password form on the main website	
Flow of Events	**Actor**	**System**
	1. A user forgets password, click "Forgot Password"	2. System forwards user to forgot password form – this system handles the rest
Exception Conditions	n/a	

Use Case Name	Search Foods	
Scenario	User searches for a food	
Brief Description	A user searches for a food choice to log to food diary	
Actors	Registered customers	
Related Use Cases	Add Food, Add Favorite Food	
Preconditions	The search food functional is available only when attempting to add a food; The view is after a meal selection is chosen	
Post conditions	The food searched is chosen and the add food view is displayed	
Flow of Events	**Actor**	**System**
	1. The user types a food description into the search bar	2. The system will perform auto-search after the third character is typed 2.1 Display list of foods that match the search criteria
Exception Conditions	1. If no foods are found a. A message "No foods were found for <search term>"	

	2. Cancel Search a. Clear search data, display food diary

Use Case Name	Change Diary Date	
Scenario	User changes the food diary date	
Brief Description	A user can change the diary date to view foods and nutrition information for that day	
Actors	Registered customers	
Related Use Cases		
Preconditions	The user must be on either the food diary view or the overview view	
Post conditions	Once the date is accepted, display the previous view (food diary, overview)	
Flow of Events	**Actor**	**System**
	1. User taps the food diary date 3. User changes the date and accepts by clicking "Accept"	2. Display change date view 2.2 Auto-selected date input field to render date picker keypad 4. Fetch diary information for date selected 4.1 Display previous view with information for date selected
Exception Conditions	1. If customer enters an invalid date into field a. Provide error message 2. If customer changes date, but then cancels a. Return customer to previous view with previous date selected	

<table>
<tr><td>Use Case Name</td><td colspan="2">Add Food</td></tr>
<tr><td>Scenario</td><td colspan="2">User adds a food to their diary</td></tr>
<tr><td>Brief Description</td><td colspan="2">A user add the foods eaten for a given day to their food diary</td></tr>
<tr><td>Actors</td><td colspan="2">Registered customers</td></tr>
<tr><td>Related Use Cases</td><td colspan="2">Add Favorite, Edit Food, Delete Food</td></tr>
<tr><td>Preconditions</td><td colspan="2">The user must be on the food diary view</td></tr>
<tr><td>Post conditions</td><td colspan="2">The newly added food choice will be display on the food diary list</td></tr>
<tr><td rowspan="6">Flow of Events</td><td>Actor</td><td>System</td></tr>
<tr><td>1. User taps the add button on the food diary header</td><td>2. Display select meal view</td></tr>
<tr><td>3. User selects the meal for the food choice</td><td>4. Store selection and display Search Foods</td></tr>
<tr><td>5. User selects a searched food from the search foods list</td><td>6. Fetch information for food choice and pre-populate default servings</td></tr>
<tr><td>7. User changes serving options to desired values</td><td>8. On change, the system will update food information for the food and dynamically refresh the view</td></tr>
<tr><td>9. User selects “Add” to add the food to the food diary</td><td>10. The system adds the food to the diary
10.1 Display food diary with new food</td></tr>
<tr><td>Alternate Flow</td><td colspan="2">1. On the Search Foods View, the customer selects Favorites
2. A list of the customer favorites are provided
3. Customer can search and/or add food from favorites</td></tr>
<tr><td>Exception Conditions</td><td colspan="2">1. If customer cancels Add Food
a. Customer is forwarded back to the Food Record view
2. If customer does not comply with the required fields for the form
a. An error notification is supplied
3. If customer does not have favorite foods
a. Give message to customer that no favorite food exist</td></tr>
</table>

Use Case Name	Delete Food	
Scenario	User deletes a food from their diary	
Brief Description	Once foods have been added to a food diary, the user can delete the food	
Actors	Registered customers	
Related Use Cases	Add Food, Edit Food	
Preconditions	The user must be on the food diary view and have foods to delete	
Post conditions	The food diary and nutrition summary will reflect the delete action	
Flow of Events	**Actor**	**System**
	1. On the food diary, the user taps the "x" delete button for a food choice from the list 3. Display the food diary without the food deleted	2. The system will delete the food from the diary and retrieve new values
Exception Conditions		

Use Case Name	View day overview	
Scenario	User views the nutrition summary for a given food diary	
Brief Description	On the food diary view, a user is presented with a high-level summary of the day; A user can click the summary to view the full summary for the day and change the date to view another day's breakdowns	
Actors	Subscribers	
Related Use Cases	View nutrition facts	
Preconditions	The user must be on the food diary date and foods must exist on diary	
Post conditions	The user views the day summary of nutrients and breakdowns	
Flow of Events	**Actor**	**System**
	1. User is on food diary view and has foods logged; the users taps the arrow next to the top-4 summary 3. User taps "Done"	2 Fetch and pre-populate overview 2.1 Display overview 4. Return to food diary 4.1 Clear data
Exception Conditions	1. Customer does not have foods logged a. Notify customer with a message	

Use Case Name	Edit Food	
Scenario	User edits a food to their diary	
Brief Description	Once a food has been added to a food diary, the user can choose to edit serving information and/or the meal it was eaten	
Actors	Registered customers	
Related Use Cases	Add Food, Delete Food	
Preconditions	The user must be on the food diary view and have foods to edit	
Post conditions	The newly edited food choice will be display on the food diary list	
Flow of Events	**Actor**	**System**
	1. User taps the food choice row for the food they wish to edit 3. User changes serving options and meal selection to desired values 5. User selects "Edit"	2. Fetch and pre-populate food date 2.1 Display edit view 4. On serving information changes, the system will update food information for the food 4.1 Refresh UI 6. The system makes the food choice change the food to the diary 6.1 Display food diary with new food
Exception Conditions	2. Customer cancels update food a. Customer is forwarded back to Food Record view, no changes 3. Customer fails to enter required form fields b. Warnings provided, detailing problems with form	

Use Case Name	Add Favorite Food	
Scenario	User adds a favorite food to diary	
Brief Description	On the Search Foods screen, if user has favorites, these can be added to their food diary	
Actors	Subscribers	
Related Use Cases	Add Food, Search Foods	
Preconditions	The user must be on the Search Foods view and have favorites	
Post conditions	The newly add favorite food choice will be display on the food diary list	
Flow of Events	**Actor**	**System**
	1. User taps add, selects meal, and taps Favorites on the Search Foods view 3. User can selects a favorite food to "Add" by tapping its row	2. Fetch and pre-populate favorites 2.1 Display list view 4. Fetch default serving information 4.1 Display Add Food 4.2. Proceed with "Add Food" case
Alternate Flow	1. Once the user is on the Search Screen and the Favorites is 'on', they can type the favorite in the search bar, which will automatically search favorite foods using the built-in list view filter search of jQuery mobile 2. If the search filter comes up empty, the user can search food database for description entered	
Exception Conditions	1. Customer cancels a. Customer is forwarded back to Food Record view, no changes 2. Customer does not have favorites a. "No favorites yet"	

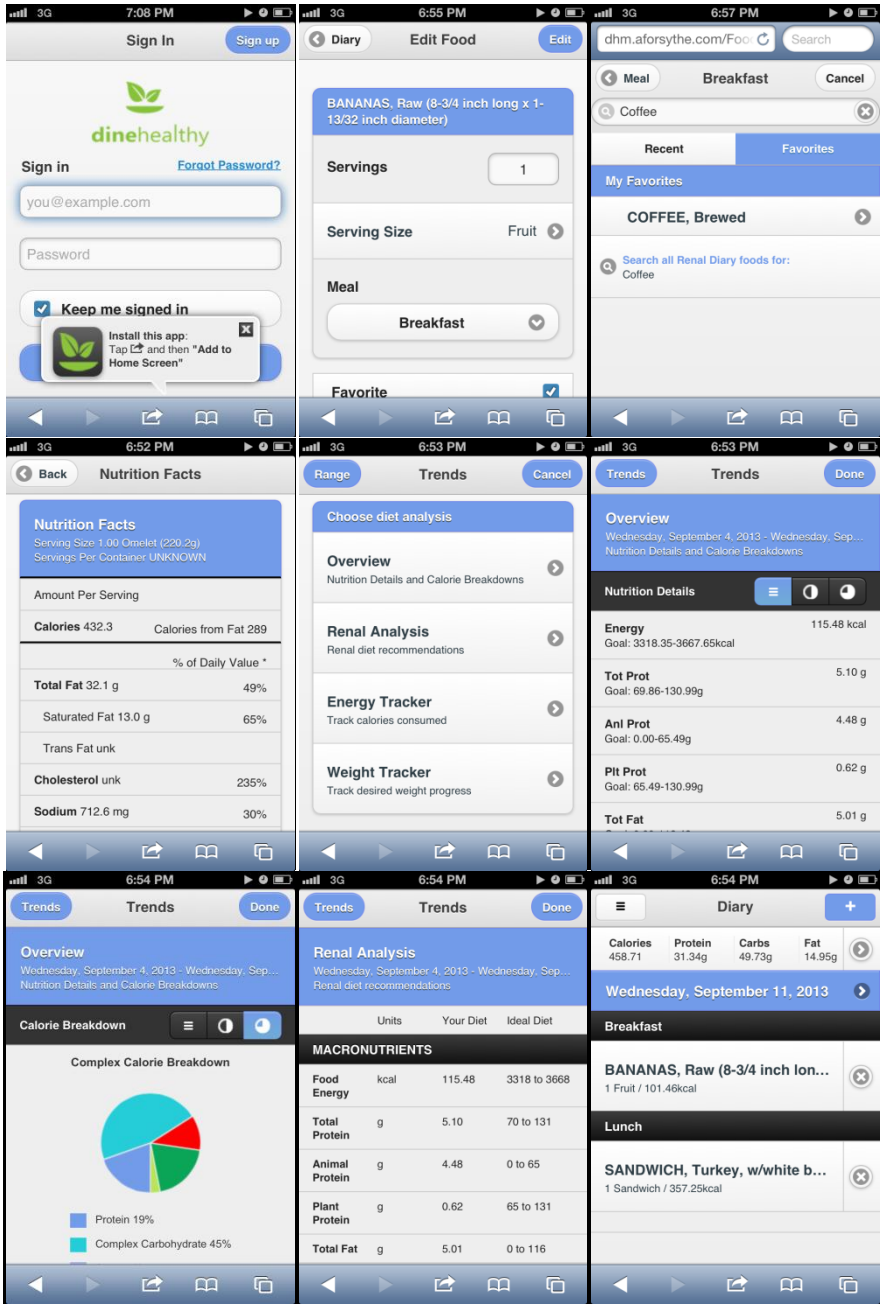
Use Case Name	View nutrition facts	
Scenario	User can view nutrition facts for a given food	
Brief Description	On the edit or add views, the user has the option to view nutrition facts for that food	
Actors	Subscribers	
Related Use Cases	Add Food, Edit Food, View day overview	
Preconditions	The user must be on the add or edit views	
Post conditions	The user directed to previous view after viewing the facts	
Flow of Events	**Actor**	**System**
	1. User taps “View Nutrition Facts” button on the add or edit view 3. User taps “Done”	2. Fetch nutrition facts data and pre-populate UI 2.1 Display facts 4. Display previous view

Use Case Name	Choose diet trend	
Scenario	User can choose which particular diet analysis to view	
Brief Description	A user chooses which diet analysis trend to view	
Actors	Subscribers	
Related Use Cases	View nutrition facts, view day overview	
Preconditions	User is on the choose trends view	
Post conditions	The user is directed to the trend choice	
Flow of Events	**Actor**	**System**
	1. User taps Trends, chooses the date range 3. User taps the trend choice to view 5. User taps “Done”	2. Display trend choices 4. Fetch date for date range and trend chosen 4.1 Display analysis 6. Clear data 6.1 Display food record
Alternate Flow	1. User choose another trend to view by navigating back from the trend view	
Exception Conditions	1. Customer cancels a. Customer is forwarded back to Food Record view 2. Customer does not have data for range a. No data shown, message provided	

Use Case Name	Choose trend date range	
Scenario	User can choose a data range for diet analysis	
Brief Description	A user can choose from a predetermined data range list for trends	
Actors	Subscribers	
Related Use Cases	View nutrition facts, view day overview, Choose diet trend	
Preconditions	User is on the choose trend date view	
Post conditions	The user is directed to the trends list view	
Flow of Events	**Actor**	**System**
	1. User taps Trends 3. User taps the date range choice	2. Display trend date range view 4.Store choice 4.1 Display list of trends
Exception Conditions	1. Customer cancels a. Customer is forwarded back to Food Record view	

Use Case Name	View trend	
Scenario	User can view diet analysis reports for a given date range	
Brief Description	Diet analysis trends allow users to determine how well or how poorly they are eating	
Actors	Subscribers	
Related Use Cases	View nutrition facts, view day overview, choose diet trend, choose trend date range	
Preconditions	User has chosen date range and trend to view	
Post conditions	The user is directed to food diary after view is complete	
Flow of Events	**Actor**	**System**
	1. User taps Trends, chooses the date range, and selects a trend from a list 3. User taps "Done"	 2.Fetch diet analysis data for trend and range 2.1 Display trend 6. Clear data 6.1 Display food record
Alternate Flow	1. User choose another trend to view by navigating back from the trend view	
Exception Conditions	1. Customer cancels a. Customer is forwarded back to Food Record view 2. Customer does not have data for range a. No data shown, message provided	

Appendix D: Screen Shots



Appendix E: JavaScript Source Code

(This appendix has been removed due to potentially confidential / proprietary content.)