Ander’s Kitchen: Take-Out Restaurant Website

University of Kentucky Computer Science

Senior Design Project

Spring 2017

# Index

[**Index**](#_3hmmxstm7gri) **1**

[**Personnel**](#_mltodzpginjf) **3**

[**Course, School, Date**](#_xhdt64b0683c) **3**

[**CS499 Computer Science Senior Design Project**](#_rsex8ym1sh7s) **3**

[**Disclaimer**](#_rymiazucok64) **3**

[**Abstract**](#_zf6cepemhhhf) **4**

[**Introduction**](#_l1uc0s8v2qh2) **4**

[**Project Requirements**](#_hbbnm32kfv3x) **4**

[**Product Planning**](#_upavgbuvn8ve) **5**

[Sizing estimate](#_9o62suupmmw5) 5

[Front-end](#_epq1gwwfysj0) 5

[Welcome Page](#_d6bvcqhibmgg) 5

[Owner interface](#_uibjjigbw15i) 6

[Customer interface](#_8z9ah2d23pm3) 7

[Back-end](#_ozqoo3svbrsg) 7

[MySQL Server](#_ue5x4hz3b492) 7

[API Server](#_uctbrucuzr4m) 8

[Analysis of Pre-/Post-Implementation Estimations](#_viyadicscows) 8

[**Schedule & Milestones**](#_qx65yy3licdr) **9**

[**Platforms, Tools & Languages**](#_ynaa9mevfsdu) **10**

[Environment](#_vtt1kyrjpyw) 10

[Front-end](#_dy183qu6vt4u) 10

[Languages:](#_bk1moa91lk7k) 10

[CSS Development Framework:](#_j4zwwr8sdayt) 10

[Development Environments:](#_n5qxvxorodj3) 10

[Back-end](#_jcbfyzqn4vre) 10

[Languages:](#_c7nemvihgwuw) 10

[Servers:](#_g8wkz4gq1uin) 11

[Development Environments:](#_3z1w83cfz8d5) 11

[**Design**](#_yjs19k2c99pa) **12**

[Module Descriptions and Data Flow](#_nqexqxxtevxa) 12

[Front-end](#_jo0cl87cl439) 12

[Back-end](#_l2q37a52xh1l) 12

[User Screens](#_iyngv7ap1cj) 13

[Menu Interface](#_nl8gh6obwldz) 13

[Cart Display](#_adpcv8s8u41h) 14

[User Scenarios (Use Cases)](#_jdoj9jgjx4mg) 15

[Owner](#_58vmiks77h2i) 15

[Customer](#_ple7nn8p5und) 15

[ER-Diagram and SQL DB Outline](#_eondviwq3llb) 16

[Design Considerations](#_szdb2ffphuy8) 17

[CSS Framework](#_llhgo0i4lci5) 17

[API](#_a6cwkrll3h3x) 17

[Documentation specific to the project](#_y57x8l1kyai9) 17

[Front-End](#_i14we5m9hsr7) 17

[Back-End](#_snmxccaiv2sn) 17

[**Implementation**](#_sdhmj6fh75tb) **18**

[Unit Testing](#_8bc91ex5wfcr) 18

[General Website](#_lqdr5pr1dazy) 18

[Customer](#_34nvojcal4e5) 18

[Owner](#_s2sr9m58qxof) 18

[Integration/Function Testing](#_1sxxuob7x41t) 19

[System Testing](#_v0fva61t1gah) 20

[Browsers](#_8z2w6ite6cv6) 20

[**Future Enhancements & Maintenance**](#_i9u59a7zd2su) **21**

[Future Enhancements](#_85pogcrnff63) 21

[Optional, Unimplemented features](#_12tpgnc1jllr) 21

[Quality of life](#_2xgodcqym6bc) 21

[Maintenance (KNOWN ISSUES)](#_ygxtalsibmu2) 21

[**Conclusions**](#_hvba71yilfss) **22**

[**References**](#_riug5vm22lit) **22**

[Front-End](#_oi3kyq3v71r3) 22

[Back-End](#_ah3br3j0tpbe) 22

[**User’s Manual and Installation Guide**](#_kukyh3b0vsra) **23**

[Front-End](#_bwnhvficb8yq) 23

[Shopping list scheduler](#_t69onb4ydw7b) 23

[Hosting platform](#_6ouzv36umea) 23

[Back-End](#_e0c3obfngtza) 23

# Personnel

* Adam Al-dbhany
* Kalib Lindsey
* Kristen Shearer
* Owen Tanner Wilkerson

# Course, School, Date

# CS499 Computer Science Senior Design Project

*Ander’s Kitchen - Takeout Restaurant Website Concept*

University of Kentucky

April 28, 2017

# Disclaimer

This project has been designed and implemented as a part of the requirements for CS-499 Senior Design Project for Spring 2017 semester. While the authors make every effort to deliver a high quality product, we do not guarantee that our products are free from defects. Our software is provided "as is," and you use the software at your own risk.

We make no warranties as to performance, merchantability, fitness for a particular purpose, or any other warranties whether expressed or implied.

No oral or written communication from or information provided by the authors or the University of Kentucky shall create a warranty.

Under no circumstances shall the authors or the University of Kentucky be liable for direct, indirect, special, incidental, or consequential damages resulting from the use, misuse, or inability to use this software, even if the authors or the University of Kentucky have been advised of the possibility of such damages

# Abstract

This project, given the name Ander’s Kitchen, is a web application designed to run a take-out only restaurant business plan. The goal of this particular business model is to create fresh and healthy family-style and family-sized meals at a low cost. A major component of this project was to set up and operate a rudimentary dot-com business. By having different web applications for different aspects of the business, we hope to simplify the “startup” in “startup businesses”, specifically restaurants. The result of this project is a website which allows easy-to-use features both for the owner in running the business and the customer in accessing it.

# Introduction

The goal of this restaurant is to do one thing very well: to make high-quality, family sized dinners for take-out or delivery at a low cost. At the beginning of each week, a menu will have been published containing the meal available each day. Customers will be able create secure accounts, order meals for each night of the week, and indicate whether they would like to pick up the meal or have it delivered to their homes for an extra fee. The goal is to create a model which is easy and convenient for customers, all at a low price.

For the owner and administration portal, the goal is to cut unnecessary costs for a small business owner. Due to the nature of the business model, it becomes necessary to only buy fresh ingredients that are needed to fill the orders that have been placed. This helps to cut down on the food that would normally go to waste.

This style of web-based restaurant will hopefully be seen as an innovative way to revolutionize the eating experience. The users we targeting with this project are families who struggle getting all the kids out to dinner. We are trying to provide a quality service to parents who want to provide a healthy meal for the entire family but simply do not have the time to prepare it. And most importantly, our software can work for anyone who wants to try a different, more modern type of restaurant with less waste, less inconvenience, fewer lines, and endless options.

# Project Requirements

The following contains the list of required features which were required by the customer

* Responsive website allowing users to register accounts, purchase meals, and sign up for the delivery
* Specialized interfaces for users and owners
* Input fields must be secured against injection attacks
* A back-end that translates the current day's orders into a shopping list
* Back-end must track meal purchases along with ingredient purchases
* A 'Recipe Book' that allows owners to store recipes and their ingredients for future use
* Integration of a payment system

# Product Planning

## Sizing estimate

The following sizing follows the Fibonacci story points model common in agile models (we use the modified format 0.5,1,2,3,5,8,13 and a priority of 1-5 with 1 being top priority in the story)

### Front-end

#### Welcome Page

**Pre-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story points |
| Present weekly meal schedule | 1 | 1 |
| Sign in page | 1 | 1 |
| About us page | 2 | 0.5 |
| Contact us page | 3 | 0.5 |

##### 

**Post-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story points |
| Present weekly meal schedule | 1 | .5 |
| Sign in page | 1 | 1 |
| About us page | 2 | .5 |
| Contact us page | 3 | .5 |

#### 

#### Owner interface

**Pre-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story points |
| Meal Scheduling | 1 | 3 |
| Recipe book | 1 | 5 |
| View Daily orders | 1 | 0.5 |
| Viewing daily shopping list | 1 | 1 |
| Financial history viewing and analysis | 2 | 3 |
| Order history viewing | 3 | 1 |
| Customer profile viewing | 4 | 1 |
| View popular meals | 5 | 0.5 |

##### 

**Post-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story points |
| Meal Scheduling | 1 | 5 |
| Recipe book | 1 | 5 |
| View Daily orders | 1 | .5 |
| Viewing daily shopping list | 1 | 3 |
| Financial history viewing and analysis | 2 | deprecated |
| Order history viewing | 3 | .5 |
| Customer profile viewing | 4 | .5 |
| View popular meals | 5 | deprecated |

#### Customer interface

**Pre-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story points |
| Make/schedule a meal order | 1 | 3 |
| View order history | 2 | 2 |
| View profile | 3 | 2 |

**Post-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story points |
| Make/schedule a meal order | 1 | 1 |
| View order history | 2 | deprecated |
| View profile | 3 | 1 |

### 

### Back-end

#### MySQL Server

**Pre-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story Points |
| Server Instance | 1 | 2 |
| DB Schema | 1 | 1 |

**Post-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story Points |
| Server Instance | 1 | 1/2 |
| DB Schema | 1 | 2 |

#### API Server

**Pre-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story Points |
| Server Instance | 1 | 3 |
| DB Table Models | 1 | 5 |
| Data Views | 2 | 5 |
| API Routes | 2 | 5 |

**Post-implementation Estimation**

|  |  |  |
| --- | --- | --- |
| Feature Name | Story Priority | Story Points |
| Server Instance | 1 | 3 |
| DB Table Models | 1 | 5 |
| Data Views | 2 | deprecated |
| API Routes | 2 | 8 |

## 

## Analysis of Pre-/Post-Implementation Estimations

Most of the components were roughly the same story point value between pre and post implementation estimations. This is said in exception to the Shopping list generation and API routes. The shopping list generation required an independent node client script to be written to automate the querying to and from the database and Spoonacular and construct the shopping list. The scheduling aspect of this became slightly involved. Additionally, API route development took some time as there had to be fairly constant revisions to existing routes while new route development pushed on in parallel.

# Schedule & Milestones

|  |  |  |
| --- | --- | --- |
| Class/Deadline | Fri, Jan 27 | Projects assigned, teams formed |
| Class/Deadline | Fri, Feb 10 | Project web page up and running |
| Milestone | Sat, Feb 11 | Start setting up AWS EC2 instance and host tools |
| Class/Deadline | Fri, Mar 3 | Status update meeting with instructor |
| Milestone | Sat, Mar 4 | Finish setting up AWS EC2 instance and host tools |
| Milestone | Sat, Mar 4 | Start front-end development |
| Class/Deadline | Mon, Mar 6 | Design page due |
| Class/Deadline | Fri, Mar 10 | Midterm presentation |
| Milestone | Fri, Mar 24 | First functioning version of the API pushed to Github |
| Milestone | Sat, Apr 1 | Finished optional HTTPS on API Server |
| Class/Deadline | Mon, Apr 3 | Test plan due |
| Class/Deadline | Fri, Apr 7 | Test plan meeting with instructor |
| Milestone | Sun, Apr 9 | Integration testing can begin |
| Class/Deadline | Fri, Apr 14 | Code review/status meeting with instructor |
| Milestone | Mon, Apr 17 | Finished final version of Database schema |
| Class/Deadline | Fri, Apr 21 | Final Presentation Practice |
| Milestone | Tues, Apr 25 | Final version of API pushed to Github |
| Class/Deadline | Fri, Apr 28 | Final Presentation |
| Milestone | Wed, May 3 | Project delivery |

# Platforms, Tools & Languages

## Environment

## 

## Front-end

#### Languages:

* HTML5
* Sass compiled to CSS3
* JavaScript

**Motivation**: Previous experience with the platforms (would recommend angularjs in future revisions)

#### CSS Development Framework:

* Foundation 6

**Motivation**: Previous positive experience with the platform

#### Development Environments:

* Netbeans with Netbeans Browser Connector
* Cordova Mobile System Emulation
* Selective Steve Sass compiler

**Motivation**: Previous positive experience with the platforms. Netbeans has great emulation

### Back-end

#### Languages:

* Javascript:
* Node.js

**Node Package Modules (NPM)**: Many packages were used and can be find in the *package.json* file. The packages listed below are the main technologies which allowed for the creation of a web server and a connection to the database:

* Express.js
* Sequelize.js
* MySQL

**Motivation**: Taking an API approach opposed to just php files which communicate to the database was to satisfy one of the main requirements: scalability. This allows for future development of a mobile app to easily communicate via HTTP requests. Additionally, the plethora of resources available online, and many companies choosing to opt for a Node.js based API server, this seemed fitting.

#### 

#### Servers:

* MySQL Server
* Express Server

#### Development Environments:

* Sequel Pro
* Postman
* WebStorm

**Motivation**: *Sequel Pro* and *WebStorm* are great applications for monitoring the current data stored in the MySQL database and code writing and formatting of the overall project, respectively.

*Postman* is a great tool for making HTTP(S) requests, allowing you to set any specific headers, crafting a JSON request in the Body, as well as seeing the status and JSON response from the server. This helped tremendously with testing and altering the way response JSONs would look.

# Design

## Module Descriptions and Data Flow

### Front-end

**Modules:**

Welcome Interface

* A welcoming page showing weekly meals, contact information, and directing to sign-in to the platform.

Owner Interface

* Access to all administrative features

Customer Interface

* Access to order placement and user profiles

Driver Interface (deprecated)

* Access to an optimized delivery route through google maps and delivery customer information

## 

### Back-end

**Modules:**

MySQL Server

* Contains all necessary tables and relations of the database, as well as all data for the users, orders, and weekly meal “forecast”

API Server

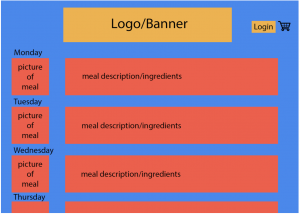
* A strictly formatted Node.js+Express server following an Model-View-Controller(MVC) convention, where the View is nothing more than a status and a JSON response to a HTTP request. This allows for validation and authentication of data before it is stored in the database, as well as the ability to add new features and statistics easily. Additionally, it allows control over what information is being sent back to the client, and its formatting.

## User Screens

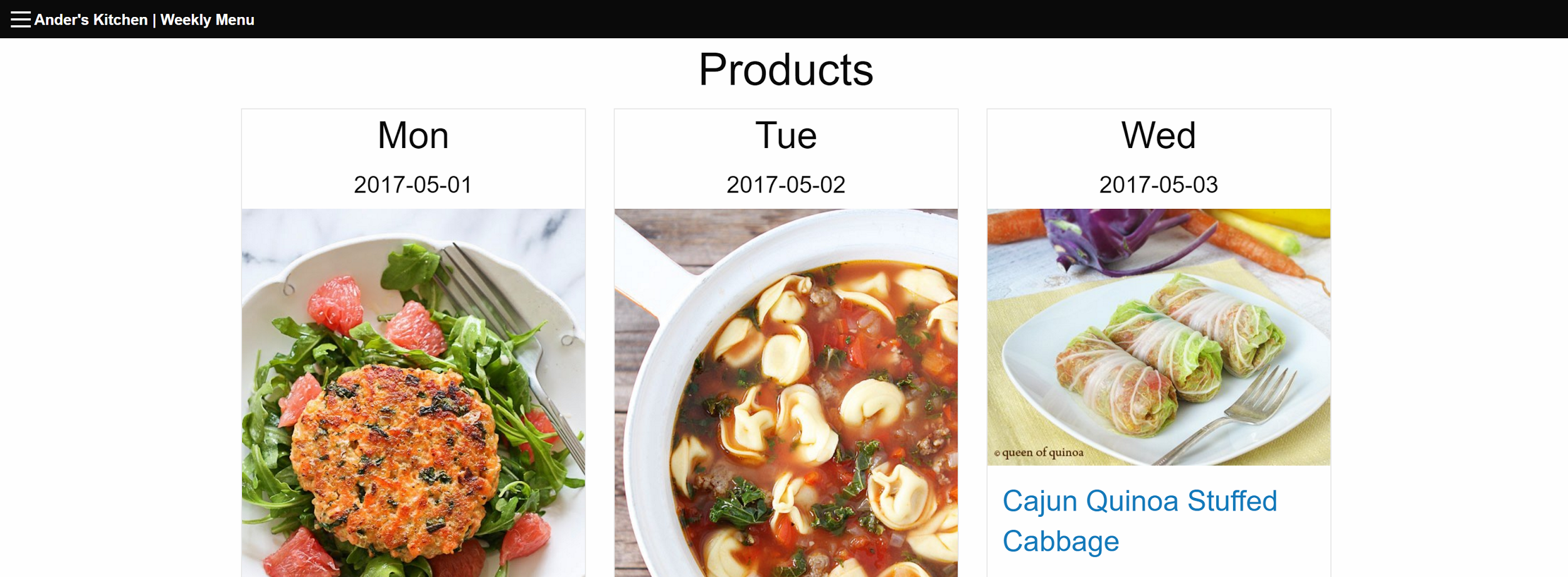
The following includes mock ups of the design direction for the interface pages followed by the final version images. Some additional mock ups are included as well.

### Menu Interface

## 

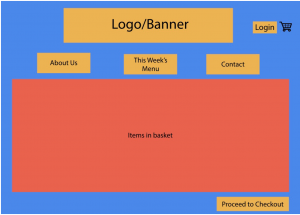


**(Mock Image)**

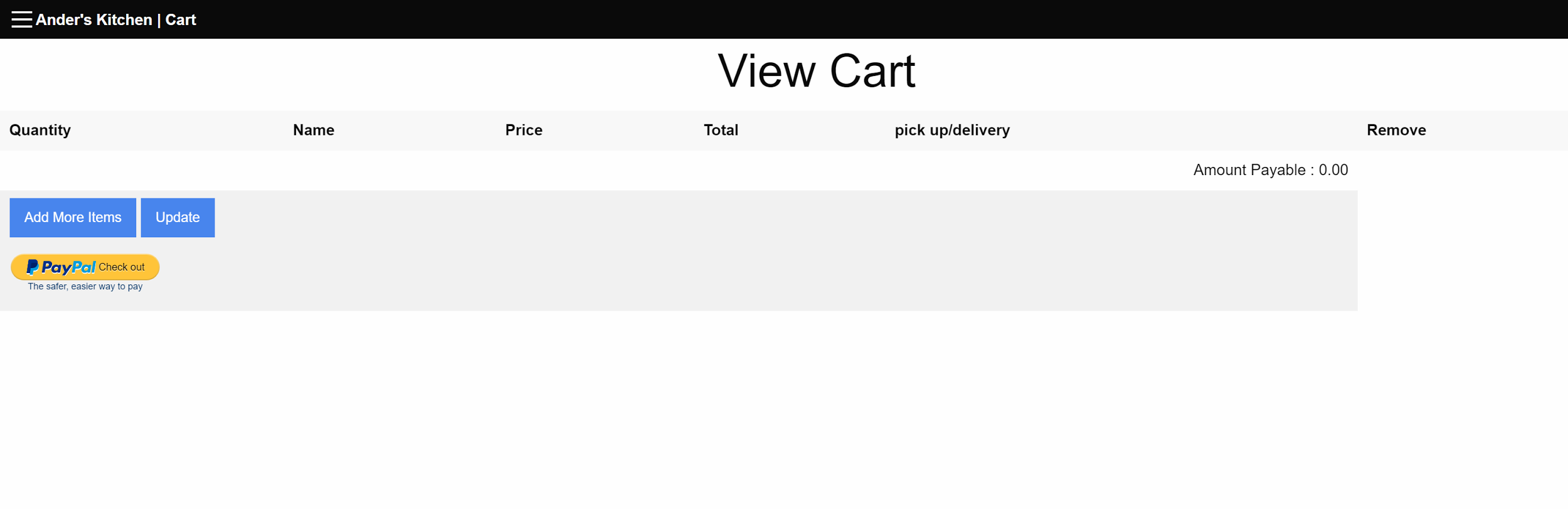


**(Final)**

### Cart Display



**(Mock Image)**

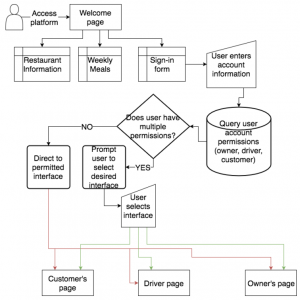


**(Final)**

## 

## User Scenarios (Use Cases)

The user scenarios all follow this preliminary system flow:



### Owner

Once an owner launches into the owner interface, the following features would become available:

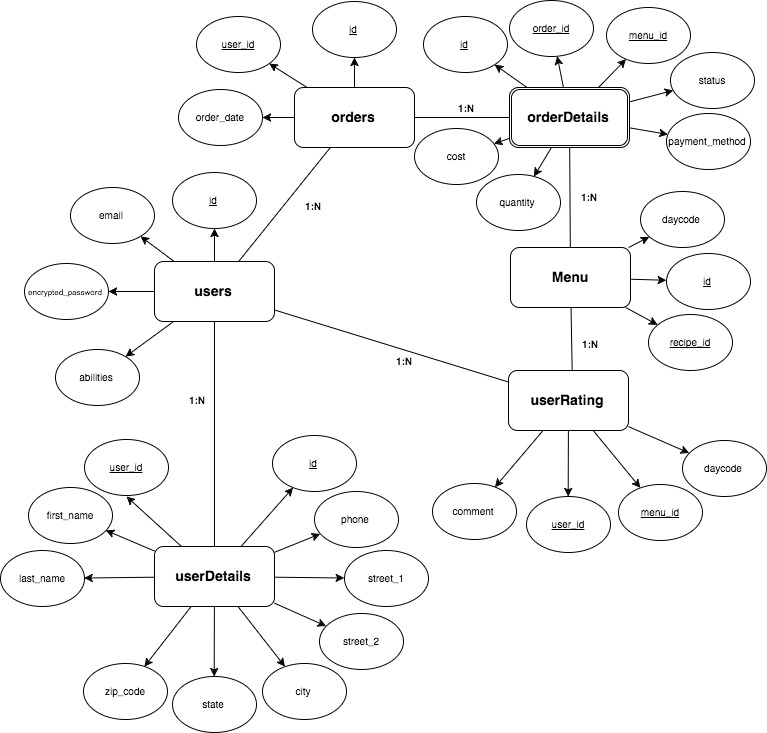
* Viewing daily orders
* Viewing daily shopping list
* Browsing order and financial data
* Browse the recipe book
* Edit recipes in the recipe book
* Schedule weekly meals as far in advance as desired
* View customer profiles, which includes order history, most ordered meals, etc.

### Customer

Once a customer signs into the customer interface, the following features would become available:

* Setup orders for the current week
* View order history
* View profile

## ER-Diagram and SQL DB Outline



## 

## 

## Design Considerations

### CSS Framework

Foundation 6 was chosen because of the front-end developer’s preexisting fluency with the framework. It is a mobile-first framework that provides an environment for rapid, responsive front-end development.

### API

Development of an API was considered over hard coding queries into the website’s front-end purely do to scalability. This will save a tremendous amount of time if we choose to have a mobile application down the road, being as all the data is controlled and distributed via the API server. This could also be extended to allow for a public API, allowing the software to be licensed and distributed.

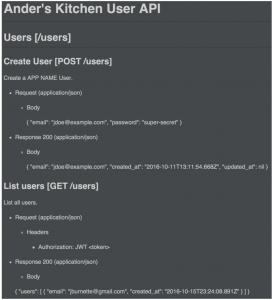
## Documentation specific to the project

### Front-End

N/A

### Back-End

For every MVC instance there is an accompany *md* file which outlines the different GET/PUT/DELETE methods implemented. This includes the initial request, any parameters or headers needed, and finally the response for a successful query. A brief example of the User’s API documentation is shown in the image below:



# Implementation

## Unit Testing

In each of the following cases, we will be providing input where necessary to ensure that the correct result/output is achieved.

PTC will refer to Positive Test Cases, NTC will refer to Negative Test Cases

### General Website

1. Test account creation functionality
   1. Result: <PASSED>
      1. PTC: Create account with all correct/new fields: <PASSED>

* Account Created
  + 1. NTC: Create account with duplicate email address: <PASSED>
* Account Not Created, Error: “Email address already in use!”
  + 1. NTC: Create account without providing password: <PASSED>
* Account Not Created, Error: “Email is required”
  + 1. NTC: Create account without providing password: <PASSED>
* Account Not Created, Error: “Password is required”
  + 1. NTC: Create account without providing first name: <PASSED>
* Account Not Created, Error: “First Name is required”

1. Test login functionality
   1. Result: <PASSED>
      1. PTC: Login with correct information: <PASSED>

* Successful log in
  + 1. NTC: Login with correct email, incorrect password: <PASSED>
* Unsuccessful log in

### Customer

1. Test cart functionality
2. Result: <PASSED>
   1. PTC: Can place meal in cart: <PASSED>
   2. PTC: Can remove meal from cart: <PASSED>
   3. PTC: Can change quantity: <PASSED>
   4. PTC: Make sure total is correct: <PASSED>
3. Test order placement
   1. Result: <PASSED>
      1. PTC: Can use Paypal button to pay for order: <PASSED>

### 

### 

### 

### 

### Owner

1. Meal Scheduling
   * Result: <PASSED>
     1. PTC: Select meal for unscheduled day: <PASSED>

* Meal assigned to day
  + 1. NTC: Select meal for scheduled day: <PASSED>
* Error message, meal already scheduled for given day

1. Recipe Book
   * PTC: Ensure that saved recipes appear in the recipe book
     1. Result: <PASSED> Successful
   * Ensure that recipes are not duplicated
     1. Result: <FAILED> (NOW RESOLVED <PASSED>)
        1. NTC: Attempt to add already added recipe: <FAILED>

(NOW RESOLVED <PASSED>)

* Duplicate added, bug which needs to be addressed

1. Daily Orders
   * PTC: Correct listings of orders to be displayed
     1. Result: <PASSED>
2. Daily Shopping List
   * PTC: Ensure that the correct quantities of ingredients are displayed
     1. Result: <PASSED>
   * PTC: Ensure that correct types of ingredients are displayed
     1. Result: <PASSED>
   * PTC: Ensure that the correct totals of ingredients are displayed
     1. Result: <PASSED>
3. Order History
   * PTC: Admin can see all past orders
     1. Result: <PASSED>
   * PTC: Ensure all order information is displayed
     1. Result: <PASSED>

## 

## Integration/Function Testing

1. Proper page redirection on log in
   * PTC: Check that the login page redirects to the correct page per admin and customer permissions
     1. Result: <PASSED>
2. Page permission checking and lockout
   * Check that customer is locked out of the proper pages if those pages are forced requested by directory (I.e. If a customer or unidentified user accesses the owner portal by requesting <https://www.anderskitchen.com/ownerPortal.php> in their browser, they should be automatically locked out of the page)
     1. Result: <PASSED>
        1. NTC: User attempt Admin Portal: <PASSED>

* Error: “You Really Shouldn't Be Here.. “
  + Make sure those with proper permissions are NOT locked out of the respective pages
    1. Result: <PASSED>
       1. PTC: Admin attempt Admin Portal: <PASSED>
* Access granted

1. Token acknowledgment upon page request (detect a past user on the same machine who didn’t log out after their last session)
   * If a user doesn’t formally sign out, are they recognized automatically upon returning to the site within 30 days of sign in
     1. Result: Was unable to test (30 days…)
   * PTC: The options to sign in is replaced with the option to sign out
     1. Result: <PASSED>
2. Token deletion upon sign out
   * Assure that the sign out function deletes the current session token stored in local storage
     1. Result: <PASSED>
   * The option to sign out is replaced with the option to sign in
     1. Result: <PASSED>

## System Testing

### Browsers

Because the website is not meant to be browser specific, it will be tested on a number of browsers including:

* Chrome
  1. Result: <PASSED>
* Firefox
  1. Result: <PASSED>
* Internet Explorer
  1. Result: <PASSED>
* Safari
  1. Result: <PASSED>

# Future Enhancements & Maintenance

## Future Enhancements

### Optional, Unimplemented features

(Those things we wanted to implement, but ran out of time):

* Route planning and optimization for driver
* Weekly newsletter system
* Email/SMS confirmation messages
* Customer surveys/ratings system
* Financial data tracking and analysis

### Quality of life

**A list of capabilities that the developers recommend to be added before this product goes live:**

Note: these were not requirements for the revision 1.0 developers, but we would have certainly implemented these given enough time.

* user: canceling orders
* admin: searching their saved and used recipes by some criteria

## Maintenance (KNOWN ISSUES)

The following are known bugs that the revision 1.0 team sadly ran out of time to fix as they required a deep dive into the a specific infrastructure.

* Scheduling meals doesn't work on mobile
  + there is an iframe/Foundation 6 Modal issue on the recipe book page was attempting to schedule a meal on mobile
* Unsyncing of mobile and desktop navigation bars on admin screen
  + there is an issue with Foundation that required us to manually sync the mobile and desktop navigation menus; in case someone resizes the window on the desktop. Our workaround works fairly well; yet, on the rare occurrence, the mobile menu will unsync from the desktop menu and reflect that the user is on a page that they are not. NOTE: this issue only happens when resizing desktop windows. everything works fine regarding navigation on mobile platforms.

# Conclusions

To summarize, we have all definitely learned a lot from this project. Most of the team had previous experience with group work but none as all-encompassing as this project. Throughout the semester, we had to come together and learn to agree on communication in order to get anything done. Our schedules were so hectic that meeting more than once a week was nearly impossible, and sometimes even once a week wasn’t a reality. From this we learned that communication is key to the success of any project, as well as the fact that organization is vital. If we were to redo the project from scratch, I believe we would try to start earlier on things we considered “easy”. In the end, the “easy” things ended up causing their own issues or just consumed more time than anticipated.

Through the project, we didn’t receive any changes to the core functionality requirements of our website. However, when we ran into problems with the security and certifications and wanted to focus more effort on that aspect, it was emphasized by Dan that all aspects still deserved equal attention even when we still had a good amount of work to do on the security. Specifically, we wanted to focus more attention on the security at the expense of some of the payment functionality, but Dan made it clear that implementation and security of payment were just as important as securing the website in general.

Overall, I believe we would say Dan is satisfied with the outcome of the project. He was pleased with the ideas we presented in meetings and noted that we had made good progress in our final meeting before the presentation and handoff. Throughout the project lifecycle, we referred heavily to his requirements and specifications to hopefully provide him with the product he originally outlined, and would like to think that we have succeeded in our endeavors this semester.

# References

## Front-End

Spoonacular:<https://market.mashape.com/spoonacular/> and

<https://spoonacular.com/food-api>

Foundation 6:<http://foundation.zurb.com/sites.html>

Unirest (used by spoonacular): <http://unirest.io/php.html>

httpful (php request lib): <http://phphttpclient.com/>

AWS: <https://aws.amazon.com/>

## Back-End

Sequelize.js: <http://docs.sequelizejs.com/en/latest/>

Express.js: <https://expressjs.com/en/guide/routing.html>

# User’s Manual and Installation Guide

## Front-End

Just put the www/ file into wherever your server is hosting its web files. Typically /var/www/html.

## Shopping list scheduler

npm install the following: node-rest-client, unirest, and node-schedule

open the script and enter a valid email and password for an admin

run in background with nohup node shoppingListClient.js &

## Hosting platform

our sever used an installation structure very similar to that found at the following link: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html>

I will include the apache conf file where I implemented the reverse proxy that points to the back end through the front end certificate (in other words, the proxy that allows <https://www.anderskitchen.com>/api/ requests to be directed and secure).

## Back-End

*Set-Up:* Please refer to the *README* doc located at the root of the API repository to find all technologies needed for the server to run, as well as how to start the server once the repo has been cloned.

*Note:* An empty database must first be setup with the same name you provide in the *.env*.

*Usage:* Please refer to the *api-docs* directory located at the root of the repository. It outlines all current HTTP(S) End-Points the API offers, as any headers or JSON body attributes needed, and what the response may look like.