# SW Engineering CSC648/848 2019 – Team 102

# Milestone 1 – 2/25/2020

PROJECT – DIGITAL INVENTORY RECOLLECTION TERMINAL

COLLABERATORS

Sanchit Joshi – Team Leader

Anthony Wong – Front End Lead

Zhifan Cai – Back End Lead

Aurpon Gupta – Scrum Master

Matthew Davis – Git Master

Christian Melendez – Server Manager and Developer

Daniel Godfrey – Database Developer

Dmitry Polozov – General Developer

REVISIONS TABLE

|  |  |
| --- | --- |
| Version | Summary |
| 1 | Initial Document Creation |
|  |  |
|  |  |
|  |  |
|  |  |

EXECUTIVE SUMMARY

# Overview

Refrigerated foods and managing the inventory of food items in a refrigerator can be a mundane task when coping with life on the go. One usually tends to forget what they store in the refrigerator and when the food items surpass expiry and are not usable anymore ultimately wasting important time you invested in buying the food items, waiting in long queues at Safeway, Target or any grocery mall nearby.

Additionally, when your refrigerator is stocked up with many food items, it is confusing to decide what to cook for dinner after a long hardworking day at school or work.

Well, we are here to save you from your misery and wasting precious time.

We have an amazing web application that notifies you of the food items in your fridge, what you can prepare for today’s meals with the nutrition facts and figures to keep you healthy and fit to work harder and smarter each day.

# The Problem

* Managing inventory of the food supplies manually is a tedious task as food items are usually bought in bulk for a household and not a single person.
* Also when the food is cooked for the entire household, it is difficult to keep track of the nutritional value that the people have consumed.
* There are various mobile and web applications in the market such as Samsung Whisk, Google Chefling, MyFitnessPal that have the ability to do individual tasks mentioned above and people are reluctant to use them due to manual update of the inventory.

# The Solution

* Our web application uses OCR( Optical Character Recognition) wherein the user can simply scan the receipt and the application will take care of the inventory management of food items.
* In addition to inventory management, the application will also provide expiry dates ,nutrient value information and meal plans to make the intricate process of cooking hassle free and convenient

# Highlights

According to[http://shrinkthatfootprint.com](http://shrinkthatfootprint.com/), USA is one of the major contributors to food waste in the world and

[www.biologicaldiversity.org](http://www.biologicaldiversity.org), says that 43% of the food waste comes from households. Below are the statistics provided from the same sources.

|  |
| --- |
| Image result for graph of household refrigerated food waste in americaImage result for graph of household refrigerated food waste in america |

This is the reason people have started being precautious about managing food inventory and started exploring multiple ways in which they can prevent the issue at hand significantly.

Our application as mentioned before uses Google Vision API to read the receipts from the grocery stores. The initial development phase will support only Safeway receipts and the idea is to broaden the store support for this application. The web application is being developed using MERN Stack( MongoDB, Express, ReactJS,NodeJS)- one of the most efficient frameworks to develop web applications. MongoDB supports NoSQL i.e the data input does not need to have a specific structure and are not limited to a fixed set of attributes.

# Our Target Audience

1. **Early Adapters**

The early adapters for this application are people between the ages of 15 and 25. These are the people that are into fitness, calorie counting and weight watching. The yoga masters, the gym rats and the healthy diet nuts. These users will use the mean plans and calorie counting everyday multiple times a day. Calorie reports and meal options will be the most often used high level functions. Sections for meals and snacks should be top priority. Calorie counts for the meal/snack and the total counts should be easily visible. Non-functional requirements for these group would be an easy way to see if the fridge has enough items for a specific meal, possibly a shade of color or an indicator to show availability. Just like the people in this group, the app should be fast, responsive, and with the least amount of distractions. The sex or race in this group for this application plays no role and does not need to be considered.

1. **Late Adapters**

The late adapters for this application are people between the ages of 25 and 40. These are the people with busy days that need an easy and fast way to keep track of what is inside the refrigerator. Majority of the group will be parents, especially single parents and/or parents with a lot of kids. The calories might not matter as much but should still be an available option. The most used high-level function for the group will be checking the “stock”. A good option for this group will be the ability to group items of their choosing together and to choose which items show up on top of the list. Setting and receiving alerts when a specific item is running low on stock is the most common option asked for during interviews. Having the ability of multiple users add new items to the list, having multiple lists for multiple users, and the ability to edit lists are also the functions desired for this group. Sex and race do not affect this group and does not need to be considered.

1. **Resistant Users**

The resistant users for this application are people over 50 years old. These are people that do not use the electronic devices every day. This is the generation that is used to making an actual paper list and carrying it to the store. Most of these users do not shop online, do not text or Facebook. The only reason these people count calories or watch what they eat is if their doctor told them to do it. This groups children and grandchildren will have to set the application app for them. This group will likely to use only one or two high level functions, maybe current items list and the calories intake for the day. The calories (or other dietary items) limit for the day/month will more than likely have to be set up by someone else and will need to be easily seen with the least amount of actions possible. Sex and race also do not apply to this group for this application.

PERSONAS AND USER STORIES

# John Doe

# John is a 22-year-old. He loves all sports and is a health nut. On a regular day John runs in the morning, weightlifting during the day, and practicing yoga in the evening. Also, John has a full-time job. John has absolutely no time for anything else and he believes in a healthy diet. It is a nuisance for John to keep opening the fridge just to take stock of what he has in there. John gets annoyed at wasting time to look for new meal ideas or even look up calorie count for the old ones. There is just not enough time in the day for john to keep track of all that information.

**User Story:** John wants to have a simple app that will keep track of what he has in the fridge.

**User Story:** John wants possible meal options provided by the application depending on the items available in the fridge.

**User Story:** John wants to see the calorie count for every meal and possible healthier options.

# Rosy Palmer

Rosy is 37-year-old mother of 5 kids. The kids ages range from 2 to 13. Rosy is a stay at home mom that is very involved in the community and her kid’s lives. During normal week there are band practices, soccer practices, dance rehearsals, karate, singing lessons and random outdoor activities. Cooking, cleaning, and picking up after all the kids takes an enormous amount of time. Rosy would like to have a better diet for herself, her husband and all the kids. There is just not enough time with her busy schedule. Rosy ends up getting pizza delivered a couple of times a week because she either does not have the time or she found her fridge half empty or lacking specific ingredients for a specific meal.

**User Story :** Rosy really wants an easy way to track the current items in her fridge that she can access any time anywhere. It would be a big relief to not have to remember what’s in the fridge with all the activities she has going on.

**User Story:** She would love some healthy meal plans and options for her family.

**User Story:** Rosy would love an option for different lists her kids and husband to access and add items they might like, but ultimately Rosy would control.

DATA DEFINITIONS

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

INITIAL LIST OF FUNCTIONAL REQUIREMENTS

# User controlled functional requirements for Product

* Users must be able to freely add/remove items into the fridge without the use of the receipt
* Users must be able to create their own meal plans
* Users must be able to create shopping lists by selecting recipes
* Users must be able to create shopping lists in order to fulfill a meal plan
* Users must be able to ask for reports about their consumption, and must be able view the report from any nutrition category(how many calories, how much protein, how many carbs etc.)
* Users must be able to view the inventory/contents of their fridge at any given time  
   - Users must be able to sort their inventory by nutrition categories.
* User must be able to search for recipes that make use of their current inventory
* User must be able to specify how much of a food item has been consumed

**Autonomous functional requirements for Product**

* Product must be able to enter contents of a receipt into the inventory by means of barcode scan
* Product must be able to enter contents of a receipt into the inventory by means of OCR
* Upon reading a receipt, Product must be able to discern the category of its contents by food/non-food, perishable/non-perishable, type of item(poultry, produce, dairy etc.), and quantities of items.
* Must be able to distinguish the type of item upon entry  
  - Product must keep a record of the date of purchase/entry of every item  
  - Product must keep a record of expiration of items if applicable
* Must be able to retrieve nutritional information about current inventory items by referencing a database of all Safeway grocery items and their nutritional information
* Must be able to retrieve nutritional information of items specified in recipes or meal plans by referencing a database of all Safeway grocery items and their nutritional information
* Product must allow the addition of other Users under control of the main User’s profile

**Other functional requirements for Product**

* Product must generate a database of all Safeway grocery items and their nutritional information

LIST OF NON-FUNCTIONAL REQUIREMENTS

**Performance**

* Application must be optimized for mobile devices
* Application must have an intuitive user interface with regards to adding/removing items, viewing inventory, receiving reports, and managing recipes and meal plans
* Application must prioritize accuracy regarding nutritional information and records of consumption
* Application should be catered to Safeway receipts only

**Expected Load**

* Application must be able to constantly assess inventory while handling user actions such as meal plans and recipes

**Security**

* Only the user should be able to see information about their fridges as well as information about their sub-user

**Storage**

* User information should be stored in a MongoDB database

**Availability**

* User should be able to easily access any of the functions presented

**Usability**

* Application must be easy to use

**Reliability**

* Application should consistently and correctly be able to read receipts and match items with the database

**Scalability**

* Application should be focused on user’s desired meal plans and recipes

**Maintainability**

* Code should be easily organized and documented so that developers may understand functionalities and their implementation.

COMPETITIVE ANALYSIS

|  |  |  |  |
| --- | --- | --- | --- |
| **MYFitnessPal** | **Chefling** | **Whisk** | **Smart Refrigerator** |
| * No specific inventory design. * Food, meals, recipes can be added * manually, or scan barcode. * Can browse restaurants located nearby supported by apply map. * Heavily featured with detailed nutrition, calorie information of each item. | * Each produce item has a predicted expiration date under different storage conditions, items can be sorted by expiration date, added time, alphabetically * When adding new items, there is quick add (with common items under different category already displayed), and barcode. | * Pantry items can be added by manual input or select from given ones. | * Pantry items can be added manually, and scan barcode and receipts. Quick add is also an option. |
| * No shopping list | * Users can manually type, and scan barcode or receipts to add shopping list items. | * An actual shopping the shopping list item online option, needs users to put zip code. * And Online grocery websites are recommended. * Users can pick the desired website and all the listed items will be put into websites’ shopping cart. | * Shopping List can be generated by user input and select existing common ones. |
| * Mostly based on users exercise plan (starting weight, goal weight, activity level). With calculated calories goals. | * Have the option to add guests, but cost money * Options to diet preference (vegan, seafood-free, disliked items etc.) | * No customized user experience section. | * Users can enter diet preference to help the app to select certain recipes to recommend. |

|  |  |  |  |
| --- | --- | --- | --- |
| * It is a fitness app, so it recommends dietary recipes according to users fitness plan. (High protein, low calorie) Mostly from its own database, or articles online. | * Recipes are generated based on current pantry items(not all), a few stored recipes and many just got from other websites. * Can also be imported by users from the web or Create Your Own Recipe | * Recipes can only be generated from manually adding favorite recipes from the web, any app, copied URL, or Create Your Own | * Recipes will be generated according to current * inventory(not all), query articles from other websites and can be created by users. |
| * Plans are not customizable. Have already made plans to choose from.   (membership needed) | * Meal plan is created by adding recipes from the recipe section, or adding it by users. | * No meal plan, but a Collections to organize stored recipes. | * Meal plan will be created by adding existing recipes. |

The overall advantages of the Smart Fridge is that it caters primarily to the user for convenience and usability. It does this by offering large varieties of recipes based off of what the user has available, from both online and customized sources and filtered based off of preferences. Meal plans are built by the user and offers information for planning and tracking available food stocks. Stocking the pantry is as easy as scanning a receipt or using quick and manual adding, and expiration dates can be tracked to see when the user might need to restock. Finally, tracking what needs to be restocked can be done with the shopping list built into the application as well.

SYSTEM REQUIREMENTS

Top of Form

Bottom of Form

We will be using the MERN stack to implement our Fridge App

**MERN Stack:**

MERN Stack is a Javascript Stack that is used for easier and faster deployment of full-stack web applications. MERN Stack comprises of 4 technologies namely: [MongoDB](https://www.geeksforgeeks.org/mongodb-an-introduction/), [Express](https://www.geeksforgeeks.org/introduction-to-express/), [React](https://www.geeksforgeeks.org/react-js-introduction-working/) and [Node.js.](https://www.geeksforgeeks.org/introduction-to-nodejs/) It is designed to make the development process smoother and easier.

Each of these 4 powerful technologies provides an end-to-end framework for the developers to work in and each of these technologies play a big part in the development of web applications.

**Server Host: Google Compute Engine:**

Google Compute Engine (GCE) is the [Infrastructure as a Service](https://en.wikipedia.org/wiki/Infrastructure_as_a_Service) (IaaS) component of [Google Cloud Platform](https://en.wikipedia.org/wiki/Google_Cloud_Platform) which is built on the global infrastructure that runs Google's search engine, [Gmail](https://en.wikipedia.org/wiki/Gmail), [YouTube](https://en.wikipedia.org/wiki/YouTube) and other services. Google Compute Engine enables users to launch [virtual machines](https://en.wikipedia.org/wiki/Virtual_machines) (VMs) on demand. VMs can be launched from the standard images or custom images created by users. GCE users must authenticate based on [OAuth](https://en.wikipedia.org/wiki/OAuth) 2.0 before launching the VMs. Google Compute Engine can be accessed via the Developer Console, [RESTful API](https://en.wikipedia.org/wiki/RESTful_API) or [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface) (CLI).

**Google Vision API**

Google Cloud’s Vision API offers powerful pre-trained machine learning models through REST and RPC APIs. Assign labels to images and quickly classify them into millions of predefined categories. Detect objects and faces, read printed and handwritten text, and build valuable metadata into your image catalog.

**Database: MongoDB**

MongoDB is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [document-oriented database](https://en.wikipedia.org/wiki/Document-oriented_database) program. Classified as a [NoSQL](https://en.wikipedia.org/wiki/NoSQL) database program, MongoDB uses [JSON](https://en.wikipedia.org/wiki/JSON)-like documents with [schema](https://en.wikipedia.org/wiki/Database_schema). MongoDB is developed by [MongoDB Inc.](https://en.wikipedia.org/wiki/MongoDB_Inc.) and licensed under the Server Side Public License (SSPL).

**MongoDB Atlas**

MongoDB Atlas is the global cloud database service for modern applications. Deploy fully managed MongoDB across AWS, Azure, or GCP. Best-in-class automation and proven practices guarantee availability, scalability, and compliance with the most demanding data security and privacy standards. Use MongoDB's robust ecosystem of drivers, integrations, and tools to build faster and spend less time managing your database.

**Web Server: Mongoose**

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

**Server-side Language: NodeJS**

**Node.js** is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), [JavaScript](https://en.wikipedia.org/wiki/JavaScript) runtime environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting)—running scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[[6]](https://en.wikipedia.org/wiki/Node.js#cite_note-6) unifying [web-application](https://en.wikipedia.org/wiki/Web_application) development around a single programming language, rather than different languages for server- and client-side scripts.

**Web Application Framework: Express**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

**Additional Software : ReactJS**

React (also known as React.js or ReactJS) is a [JavaScript library](https://en.wikipedia.org/wiki/JavaScript_library) for building [user interfaces](https://en.wikipedia.org/wiki/User_interfaces). It is maintained by [Facebook](https://en.wikipedia.org/wiki/Facebook) and a community of individual developers and companies.

React can be used as a base in the development of [single-page](https://en.wikipedia.org/wiki/Single-page_application) or mobile applications. However, React is only concerned with rendering data to the [DOM](https://en.wikipedia.org/wiki/Document_Object_Model), and so creating React applications usually requires the use of additional libraries for [state management](https://en.wikipedia.org/wiki/State_management) and routing. [Redux](https://en.wikipedia.org/wiki/Redux_(JavaScript_library))[[9]](https://en.wikipedia.org/wiki/React_(web_framework)#cite_note-9) and React Router[[10]](https://en.wikipedia.org/wiki/React_(web_framework)#cite_note-10) are respective examples of such libraries.

TEAM

**Team Leader**

Sanchit Joshi

**SCRUM Master**

Aurpon Gupta

**Back End Leader**

Zhifan Cai

**Front End Leader**

Anthony Wong

**Git Master**

Matthew Davis

**Database** **Developer**

Daniel Godfrey

**Server Developer**

Christian Melendez

**Implementation Developer**

Dmitry Polozov

CHECKLIST

* Team found a time slot to meet outside of the class

Status- DONE

* Github master chosen

Status- DONE

* Team decided and agreed together on using the listed SW tools and deployment server

Status- DONE

* Team ready and able to use the chosen back and front end frameworks and those who need to learn and working on it, along with study schedule

Status- In Progress

* Team lead ensured that all team members read the final M1 and agree/understand it before submission

Status- DONE