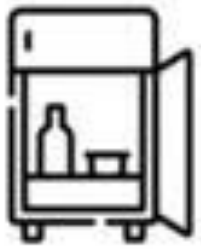
Requirements Report

Team Fridge Filler

11/6/2020



Project Sponsors:

Dr. Richard Rushforth, Sean Ryan

Team Members:

Shangyi Dai, Jonathan Derr,
Travis Flake, Gage Gabaldon, Zhibang Qin

Mentor:

Sambashiva Kethireddy

Overview

This document outlines the requirements for team Fridge Filler. The requirements will be categorized into functional, non-functional, and environmental requirements, and will be discussed in detail. After all the requirements, we will lay out our plan.

Accepted as the baseline requirements for the Fridge Filler Project. Richard Rushforth, Ph.D. - 11/24/2020

Clients: _____

Team Lead: Jonathan Derr Jonathan Derr

Table Of Contents

1. Introduction 3

2. Problem Statement 4

3. Solution Vision 5

4. Problem Requirements 6

Front End 7

Mobile App 7

Admin Portal 13

Back End 17

Database 22

Analytics 22

API (Application Programming Interface) 24

5. Potential Risks 25

6. Project Plan 26

7. Conclusion 27

Introduction

Food banks have always had problems with getting food to families, and in recent times this demand is increasing even more. Even before the pandemic, Saint Mary's Food Bank and other food banks have had trouble making sure that food is not thrown away and wasted, which is caused in part by the uncertainty of what food will come through the food banks, meaning the distribution of food products is not uniform. For example, in the United States alone:

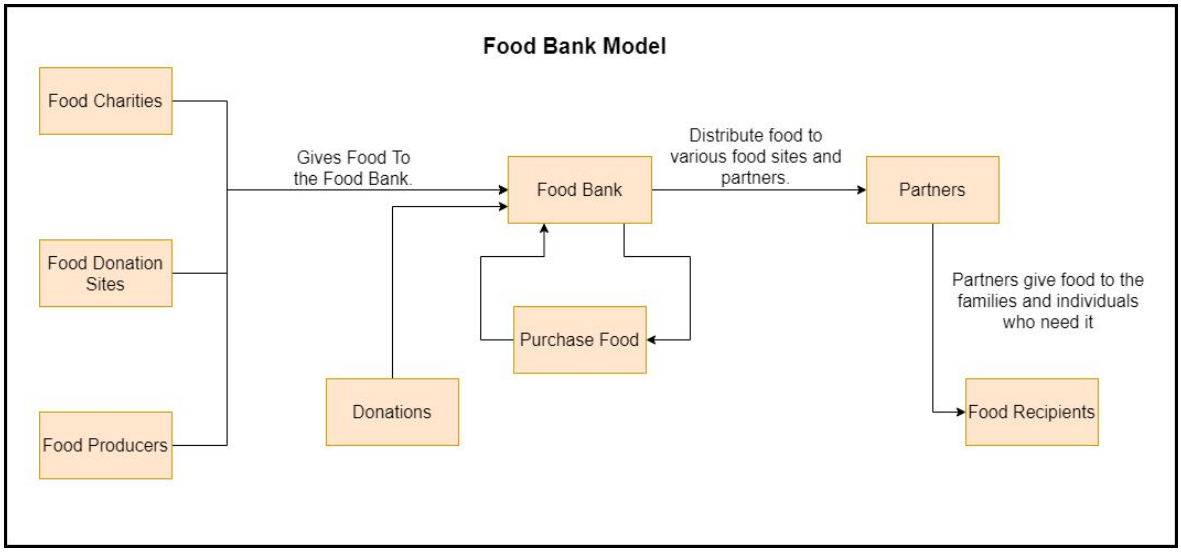
- Every year, 80 billion pounds of food are thrown away in the United States.
- Food banks across America receive different food products in varying quantities from donors throughout the country.
- Foodbank workers may be unfamiliar with products given to the food bank.

All of these factors create an inefficiency that is hard to solve with the changing supplies available to the food banks and with the lack of knowledge to best use the food. This is where our group, Fridge Filler, comes in. With our sponsors, Richard Rushforth and Sean Ryan, we have partnered with Saint Mary's Food Bank (SMFBA) to help solve this problem. They believe that the current interaction between the food banks and the people who use them is inefficient. The way they interact now is that people come into the food bank, receive a box full of whatever the food bank has available at that time, and then the person has to try and figure out what they can make with what they have just received. This is fine for some people, but not all people like to cook or know what they can make with the ingredients given. Along with the uncertainty of what will be at the food bank creates problems with the families and makes everything more chaotic.

Fridge Filler aims to create a mobile application that can serve as a go-between for the food banks and the people and families that use the food banks. That way, families can figure out what food to cook with the ingredients they have through simple, easy-to-follow recipes. The app also can recommend places to pick up missing ingredients and even a way to sync what you have at home with the app. The overall goal of the application is to help families effortlessly plan meals and to help ensure that the food at the distribution sites is used more effectively.

Problem Statement

The general workflow of a food bank is very reactionary. The food bank gets lots of different and unusual food from various food donors throughout the state. They then want to push the food as fast as they can to the food sites. Without this constant supply from the food bank, the food sites will undoubtedly run out of food. This is a good thing with the constant flow of food but comes with various issues for both the food bank and the user.



The food donated is great but causes an issue that the food bank consumers don't know what to do with some of the more eccentric and unusual ingredients. Since the food bank gets what they get, whether it is catfish or red bean paste. This causes an issue where consumers either waste the food given or don't pick up the unusual food at the foodbanks. Saint Mary's Food Bank wants to help the consumers and the people working at the various distribution centers. To better use the ingredients given.

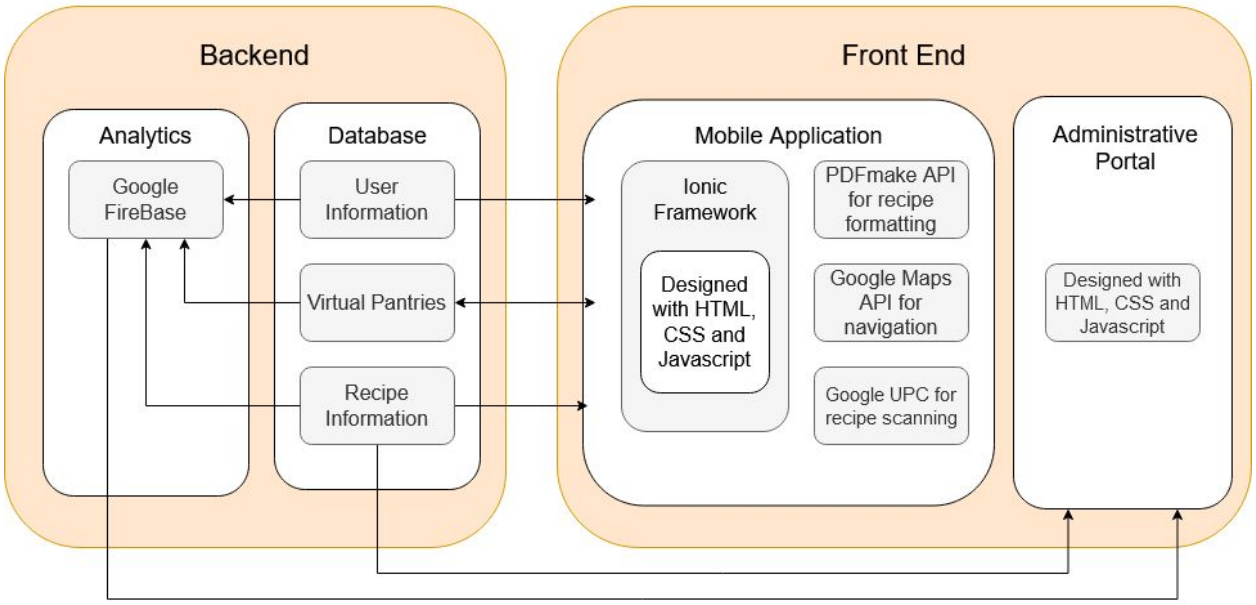
The common problems at Saint Mary's Food Bank are:

- Food is not being used effectively at the food bank.
- Unusual food and unfamiliarity with ingredients.
- Online recipes are varied and require expensive ingredients along with too many complicated steps for those served by the food bank.
- Knowledge about what to do about the food is a mixed bag.
- Foodbank workers often can't help with this, as they often don't have enough time to plan out ways for people to use the food.

Solution Vision

Our solution is a mobile application and website that will help solve the food efficacy problem. This app will primarily provide recipes to the food bank employees and those who are served by the food bank. The app will have some navigation to better help those find a nearby food bank and analytics for the food bank to know what the user is eating. The main features of the app will include:

- A searchable library of recipes that will help consumers to use the food given to them.
- A Virtual Pantry to help keep track of what food the user has and helps recommend recipes.
- Navigation to both food bank sites and food retailers (preferably SNAP and EBT)
- Analytics about the way the users use the recipes and what foods are getting used where.
- Printable recipes for personal or food bank employee use.



The recipes will be provided by SMFB and will be dynamically recommended based on the ingredients of the user's pantry and what they received at the food bank. The recipes will be simple and will provide the user with a way to use some of the more foreign food. With this, the employees of the food bank and the users can also print out these recipes to easily display and use recipes at the food sites or for personal use.
Navigation will also be a big part of the project. The app will help with navigation to food sites and different grocery stores to help fill in missing ingredients for the user.

The last part of the app is the analytics and survey part. This will run in the background and will collect data on what recipes people are using, how much they like these recipes, and what food is getting used, which could help determine how food is distributed to the different food sites in the future. If this works, then SMFB and other food banks across the nation can help use this app to help distribute the food they are given and what food to buy.

Project Requirements

This section will outline all of the requirements that our end product must-have, with the appropriate details included. To keep things organized and readable, the requirements will be separated into the two parts that will be required for our system to work: the Front End, and the Back End. The requirements will then be split up into three categories: functional, non-functional, and environmental. The functional requirements are all the necessary components of the app. The non-functional requirements are all the performance and secondary requirements the app will need. Lastly, the environmental requirements are all the requirements imposed by the project or environment.

Use Cases will be used to provide context for the different components listed below and will come from one of three perspectives.

The first perspective that will be used is Joe, a minimum-wage shift worker that recently lost his job, and relies on food packages from his local distribution center to stay fed. He never learned how to cook for himself, and downloaded this app at the recommendation of some of the volunteers at his food pantry.

The second perspective that will be used is Linda, a volunteer at Joe’s local food pantry, which is partnered with Saint Mary’s Food Bank. She wants to be as helpful for the people that come in as possible but doesn’t have the time to prepare information to put in the food boxes about how to use the ingredients in it. She often gets discouraged when she sees food that she handed out in the garbage because whoever picked it up doesn’t know what to do with it, and wants to find a better way for her food pantry to operate.

The last perspective that will be used is Mike, a part-time system administrator working for Saint Mary’s food bank. Mike’s job is to manage the data used by our app and prepare information that will be viewed by his superiors about data returned by the app.

Front End

The front end will serve as the basis of what the user will see when they open the app or view the website. This will include graphics, data analytics, and app design. This is the part of the app that users will interact with and is vitally important to make the design as simple and accessible as possible.

Mobile App

The mobile app version of this project is the main front of the app. In the mobile app, all user functionality will be present, including Navigation to food distribution centers, a digital pantry system, a recipe browser, and a way to print recipes you have saved.

Functional Requirements

Default App View (Pages)

- **Login Page**: User Login/Registration Options.
  - There will be different user types, notably an employee or worker view will be available to non-consumer users.
  - From here users will be able to login to an existing account with the app or register for a new account, where they will need to provide basic information, specifically a username, email address, and password for the account. This will be authenticated and stored within our database, and the email will specifically be needed in the case of losing account access.
  - If a user does not wish to register for an account, they can proceed as a guest, and still have access to recipe browsing and navigational features. However, they will not be permitted pantry access, as specified in sponsor meetings.

- **Navigation Page**: A Search Bar to enter a location, a map component to give navigation instructions to the nearest food bank/pantry based on location.
  - Searches based on an address, or device's location.
  - A user can input the desired location (a "starting point" if you will), and using the Google Maps API, there will be in-app navigation from the desired starting location to any food banks registered within the system.
  - If a user does not wish for navigational instruction, the default map view will contain a list of approved food distribution sites designated by administrator users from the Saint Mary's Food Bank.
  - **Stretch Goal:** The Navigation page will also allow users to find SNAP and WIC approved retailers. Since not all food bank recipients will be approved for

SNAP and/or WIC, the display of all SNAP and WIC locations will be a toggled option on the integrated map.

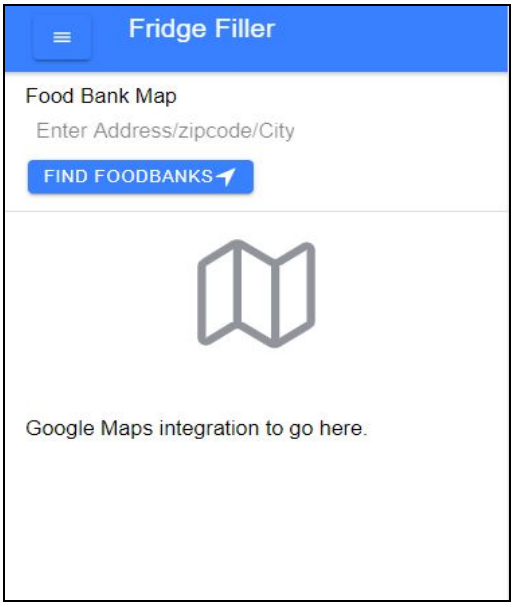


Figure X: Envisioned Maps Page Prototype Design

Use Case: Finding a Food Bank
Let's say our hypothetical user, Joe needs to find the nearest food distribution site. Using this page, he could input his current device coordinates, or use the location of the device to find which food distribution sites are nearest to him.

- **Digital Pantry**
  - Pantry Home: A home page for the pantry, where users can navigate to pages to add ingredients or manage their current pantry.

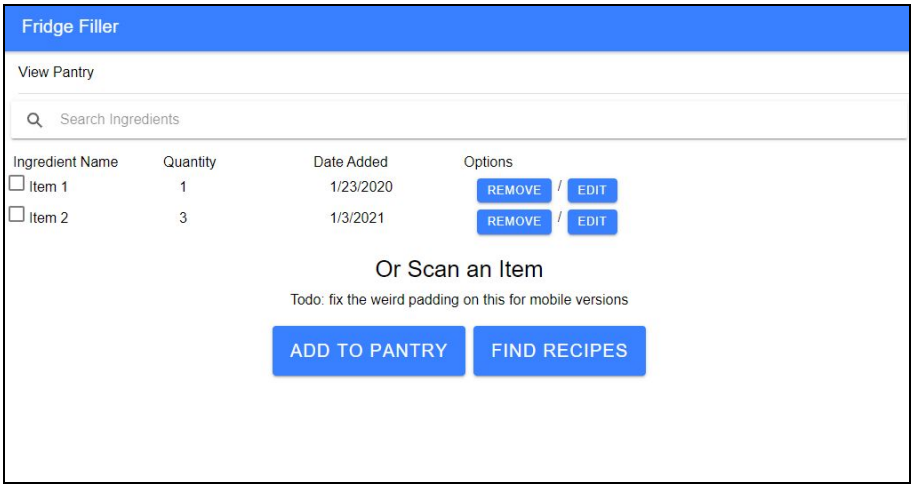


Figure X: Tentative Pantry View Design

- The Pantry Home page will serve as more of a general navigation hub for pantry features. It will contain connections to the digital pantry, where you can browse your ingredients, and search for recipes that use them, as well as a connection to the Add to Pantry page, where users can add ingredients from their Emergency Food Box (EFB) or household to the digital pantry for recipe searching and keeping inventory of what food they have.

- Add to Pantry: A page to add food items, where users input a UPC or item name, as well as quantity. Here users can also elect to flag ingredients received from a food box.
- When adding items to the pantry, the input bar will allow users to type a food item's name, as well as the quantity of the item. If preferred, UPC recognition can also be used for unknown ingredients given that they have a standard UPC format.

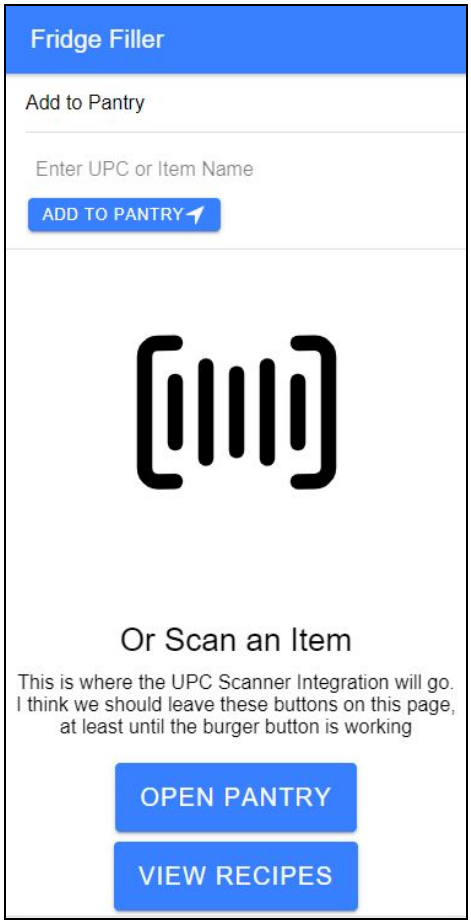


Figure X: Early Add to Pantry prototype page

        - **Note**: UPC scan recognition is listed as an optional feature, alongside item recognition. At a bare minimum, we expect the application to be capable of UPC input for food items.
    - View Pantry: A page to view currently owned ingredients. From here users can select ingredients they own to search recipes that include them. They can also edit the quantity or delete food items from this page if they no longer have or wish to use them.
        - When editing ingredients, users can modify the item name or the quantity of the item.
        - Upon opting to remove an item from the digital pantry, a warning message will appear on the screen, to ensure that a user does not accidentally delete items that they intended to keep within their pantry.
        - On each ingredient within the pantry, there will be a checkbox to select the ingredient. This feature can be used for either deletion of multiple items, or to select ingredients that you own to search the recipe database for. For instance, if a user selects chicken and rice, then press the Search Recipes button, the recipe database will look for recipes that contain both chicken and rice, and display them to you on the recipe browsing page.

Use Case: Adding Your EFB to the Pantry

Let's say Joe has just gotten home from the food distribution center and has been given his Emergency food box. From here, Joe can then enter each food item based on UPC or item name. Joe can also input how much of each item they have received. From there, Joe can use these ingredients to search the recipe database, which is explored more below.

The digital pantry feature will be used primarily by the end-user, and not as much by SMFBA workers or administrators.

- **Recipe Management**
    - Recipe Search: A page to search and view recipes by rating, ingredients, cuisine, difficulty to make, and time to make. From here you can view and "Favorite" recipes you like to print or use from the app.

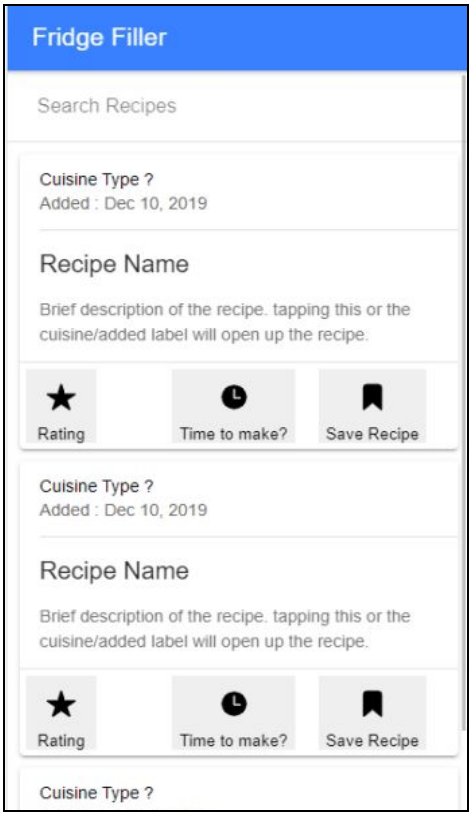


Figure x: the tentative recipe browser page design.

- Recipes can be filtered by cuisine type, time to make, difficulty to make, and ingredients
- “Favorited” recipes will show as options on the recipe printing page.
- Recipes hosted within the app will be converted to plaintext, as images included within a recipe will make it consume more storage if it were to be stored locally, and will also cause it to take longer to load on slower or unstable internet connections. The sponsors wish for the app to be accessible to as many people as possible, and for that, we are trying to maximize the number of devices that can run it by keeping the required system strength and storage low.

- Recipe Print: A print recipe page, where you can queue up recipes in your bookmarked section to print.
  - The recipe print page serves to allow consumers to choose which recipes they would like to print, along with other general printing functionality (where the document will print, recipes per page). As specified in recipe search, the recipes are plaintext, without any images included, to ensure that users will be able to load them and store them on the device without

needing a great internet connection or a large amount of storage available.

Use Case: Finding Recipes at Home

Now that Joe has received his EFB and put the ingredients into the digital pantry, he is still not sure what he can cook using his foods. He can now use his ingredients list to search the recipe database for recipes that include ingredients that Joe has selected. Joe can now look at recipes, and search recipes by name. Recipes can be filtered by time to make, cuisine type, and rating as well to help Joe find a recipe that they will like. After Joe finds recipes that they want to try or like, they can print off the recipe, or save the PDF for offline use later.

Alternate Use Case: Finding Recipes as a Food Bank Worker

Let's say our food bank volunteer, Linda needs to find recipes to provide EFB recipients. She can use the recipe library to find recipes that contain the ingredients that are in the food boxes that are being provided. She can then print the recipes off, and provide them to food bank recipients along with their food boxes.

Alternate app view based on the account type

Other than the default app view, it has been requested that there be a separate view of the app for food bank workers to ensure ease of access to the important features they will need on the job. It has also been requested that some features be gated behind a user registration, and for that, we will be providing a guest view as well.

- **Worker View**
  - The application will have a separate view for workers at food distribution sites. This view will be mainly of the recipe search and printing functions.
  - The print view will not use the pantry in this, it will use ingredients to find recipes.
  - It will use Recipe Print and Recipe Search pages from the main app view.

- **Guest View**
  - Much like the worker’s view, those who do not sign up for an account in the app will be given a restricted version of the application. Guests will not be able to store ingredients in the digital pantry, as it will be stored on an individual, user by user basis.
  - Other functionality of the app will still be fully available, but any features attached to the digital pantry will be unavailable in this view, namely the search of recipes based on the pantry and the complete use of the pantry.

Non-Functional Requirements

Appearance

- The User interface will be aesthetically similar to the Saint Mary's Food Bank website, utilizing a red and white color scheme, with simple shapes and sharp design.
    - This can be modified based on feedback from user testing, as it was also suggested that the orange and green appearance of the AZ Food Bank Network website or the Feeding America website could be referenced.

Accessibility

- The application will use larger fonts, and provide step by step instructions to users to ensure that the visually impaired and tech-illiterate aren't struggling to use the app.
    - The user experience will have an intuitive, simple design such that no new users will feel intimidated by their lack of technological knowledge and experience. All search boxes will have placeholders that tell the user what is meant to be inputted, and any instruction prompts will have simple steps.

Environmental Requirements

Cross-Platform Application

- Our client requires an application that can work on as many systems as possible to facilitate users who only own certain devices. We must develop an Android/IOS/Web version of the application.
    - The framework we have chosen, Ionic, is based on HTML, CSS, and JS, which are the building blocks for most modern web applications. The Ionic framework will allow us to develop an application that functions on IOS, Android, and as a website with simple modifications.

Admin Portal

The application needs an administrator portal to alter the information of sites and recipes provided to the food bank app users. The administrator portal will be where Food Bank officials can change the list of food distribution sites, changing the location, hours of operation, or other rules that can affect consumers visiting these sites. Administrators will also be able to edit or delete recipes that are shown within the app, potentially to add warnings for allergies, or delete recipes if the feedback is showing that the recipe is considered bad by the community.

For example, a manager needs to change the information of a food site, he will open the login page, enter username and password then login. Then he will open the food site management page and choose the edit button behind the site he needs to change and then edit the information.

Functional Requirements

Login Page

System Administrators will log in on this page to manage the database.

- Correct username and password are required, there will be no registration.
- There will be a ‘remember me’ option box to keep the information of the account and automatically fill in the information the next time the admin logs in.
- Other pages will not be able to be opened until the user has logged in successfully.
- After logging in the admin view page will be displayed and admins will be able to manage the database.

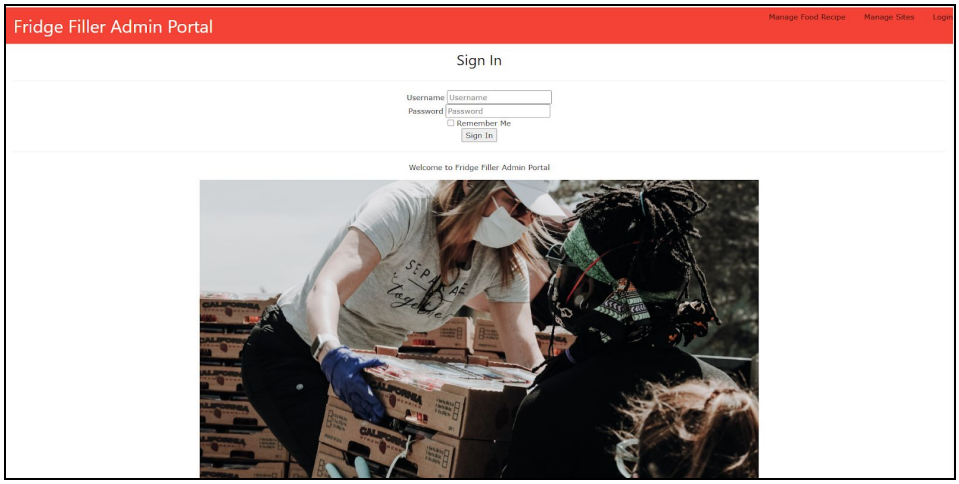


Figure: Login Page

Food Distribution Site Page

This page is used for the management of the food distribution sites.

- On this page, the name, opening time, and address of the sites will be displayed in a tabular form and there will be a bottom for the admin to edit them.
- There will be a search bar for admin to find a certain site.
- Admins can manage the name, address, information(such as opening time) of the sites through the edit bottom in the form, or add new sites through the added bottom.

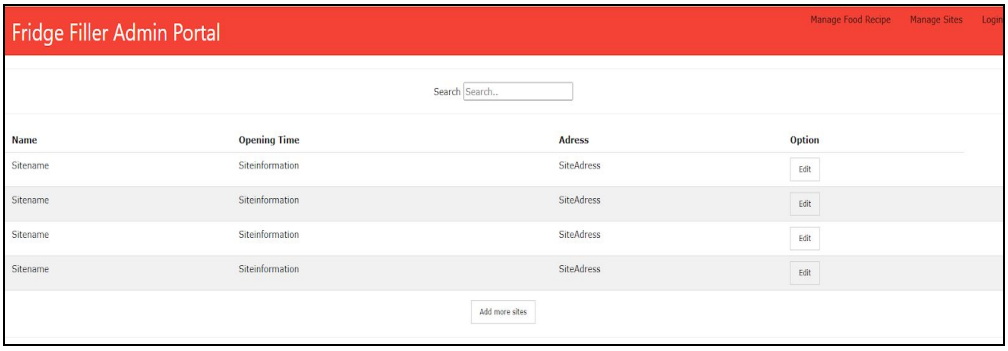


Figure: Sites Management Page

Use Case: Managing Food Distribution Sites

Our admin user, Mike, may eventually receive an updated list of food distribution sites, which is not the exact list of sites already shown within the app. Mike can then input information about the new sites here, as well as remove sites that have been closed or are no longer permanent food distribution sites according to Saint Mary's Food Bank. If a site has received a change in operating hours or other protocol to follow for visitors, Mike can also edit the general description of the site to contain this information.

Recipe Management Page

This page is used for the management of the food recipe database.

- On this page, the information of the recipe will be shown like a card that contains the picture, name, description of the recipe and there will be an edit bottom for admins.
- There will be a search bar for the admin to find a certain recipe.
- Admins can add a new recipe, delete a recipe, or manage the detailed information of the recipe through the edit bottom in the form or add new sites through the added bottom.

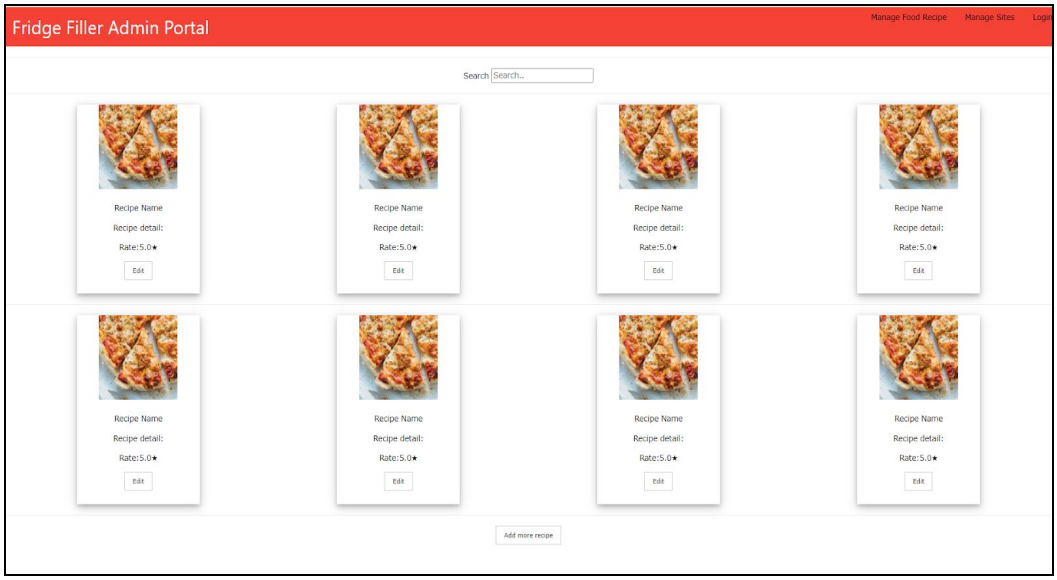


Figure: Recipe Management Page

Use Case: Changing The Recipe Database

The database for healthy recipes may need new recipes one day, and the administrator, Mike, will need to add these. Using the Add New Recipe button, Mike can input a recipe formatted to match the other recipe files, or manually input data that was submitted in an invalid format to create a new recipe listing that meets the standards of the recipes the app shows. If Mike needs to change a recipe, he could search the recipe on the recipe management page and edit the recipe data to include the updated recipe. If a recipe is deemed poor quality or too difficult to expect users to make, the recipe can also be deleted by Mike here, should SMFBA wish to.

Default View

- Login Page: Admins need to enter a username and password to log in.
- Admin view: This is a page with a bottom related to the management page of food distribution sites or food recipes after logged in.
- Recipe Management: Admins can view the recipe list, add, view, edit, or delete recipes.
- Site management: Manage the name, address, information, and so on of the existing dist sites or add new dist sites.

Non-Functional Requirements

Appearance

- The admin portal will have a simple design in the login page and admin view page. On the admin site management page there will be a form to manage the detailed information. On the recipe management page, there will be cards to show the recipe. The color of the admin portal is similar to other parts which are mainly red and white color.

Authority

- Usernames and passwords are needed on the login page.
- Except for the login page, all pages can only be opened after login.

Environmental Requirements

Robustness

- To help keep maintenance tasks to a minimum, we should try to make this admin interface as robust as possible to avoid having to add functionality later.

Back End

The backend will encompass the parts of the app that the users will not see directly. This includes the more functional parts of the project, such as data storage, analytics gathering, and general logic. This end will be the heart of the app and serves as the backbone of all the frontend design.

Database

The database contains all the data we get or build. It is the base of our application and website and will store all of the information required for the service to work properly.

Functional Requirements

Virtual Pantry

- We need a pantry table to store what food the users have. The pantry table should be able to store the following information:

- The name of the item
- The amount/quantity of the item
- Optionally, the expiration date of the item
- Optionally, the UPC (a unique identifier) of the food, if the item was added with the UPC scanner on the frontend
- In addition to this, there will be an ID number (foodID) for each item in this table that corresponds with the user ID the food item belongs to.

The following diagram provides a rough idea of what the table will look like.

foodID (PK)	foodName (SK)	foodNum	foodUPC (SK)	expirationDate

(Key: PK = Primary Key, SK = Secondary Key, FK = Foreign Key)

Recipes

- A recipe table will also be necessary to store all the recipes. The recipe table should be able to store the following information:
  - The name of the recipe
  - The ingredients of the recipe
  - The steps of the recipe
  - The ID of the recipe
  - The number of ratings the recipe has received
  - The rating of the recipe, which will be calculated as such:
    - The recipe rating will be multiplied by the number of ratings the recipe has received to get a total score. The new rating will then be added to that total score, then the number of ratings will be incremented by one. Then, the new rating will be the total score divided by the number of ratings.

The following diagram provides a rough idea of what the table will look like

recipeID (PK)	recipeName (SK)	recipeIng	recipeSteps	recipeRating (SK)	numRatings

(Key: PK = Primary Key, SK = Secondary Key, FK = Foreign Key)

Distribution Sites

- We need a site table to store the location data for all Food Bank distribution sites. The site table will help us support navigation in the app using data from Arizona Tribal Food Distributions. This table should be able to store the following information:
  - The horizontal coordinate and vertical coordinate of the site (Used to locate a point on different size of maps)

- The name of the site
- The address of the site
- The zip code of the site
- The longitude and latitude of the site
- The Phone Number of the site

Simple from the SNAP_Store_Locations.csv

ID (PK)	Name	Addr	City	State	Zip5	County	Longitude	Latitude

(Key: PK = Primary Key, SK = Secondary Key, FK = Foreign Key)

The SNAP retailers

- We need a retailer table to store the GIS data for SNAP(The Supplemental Nutrition Assistance Program) retailers. The retailer table will help us support navigation in the app using data from SNAP Retailer Locator. This table should be able to store the following information:
  - The horizontal coordinate and vertical coordinate of the retailer (Used to locate a point on different size of maps)
  - The name of the retailer
  - The address of the retailer
  - The zip code of the retailer
  - The longitude and latitude of the retailer
  - The Phone Number of the retailer

The following diagram provides a rough idea of what the table will look like.

ID (PK)	Rname	Addr	City	State	Zip5	County	Longitude	Latitude

(Key: PK = Primary Key, SK = Secondary Key, FK = Foreign Key)

Users

- We need a user table to store all the information about users and administrators. The user table should be able to contain this information:
  - The name of the user (User name)
  - The phone number of the user(Optional)
  - The email address of the user(Optional)
  - The password of the user
  - The ID of the user

The user table sample:

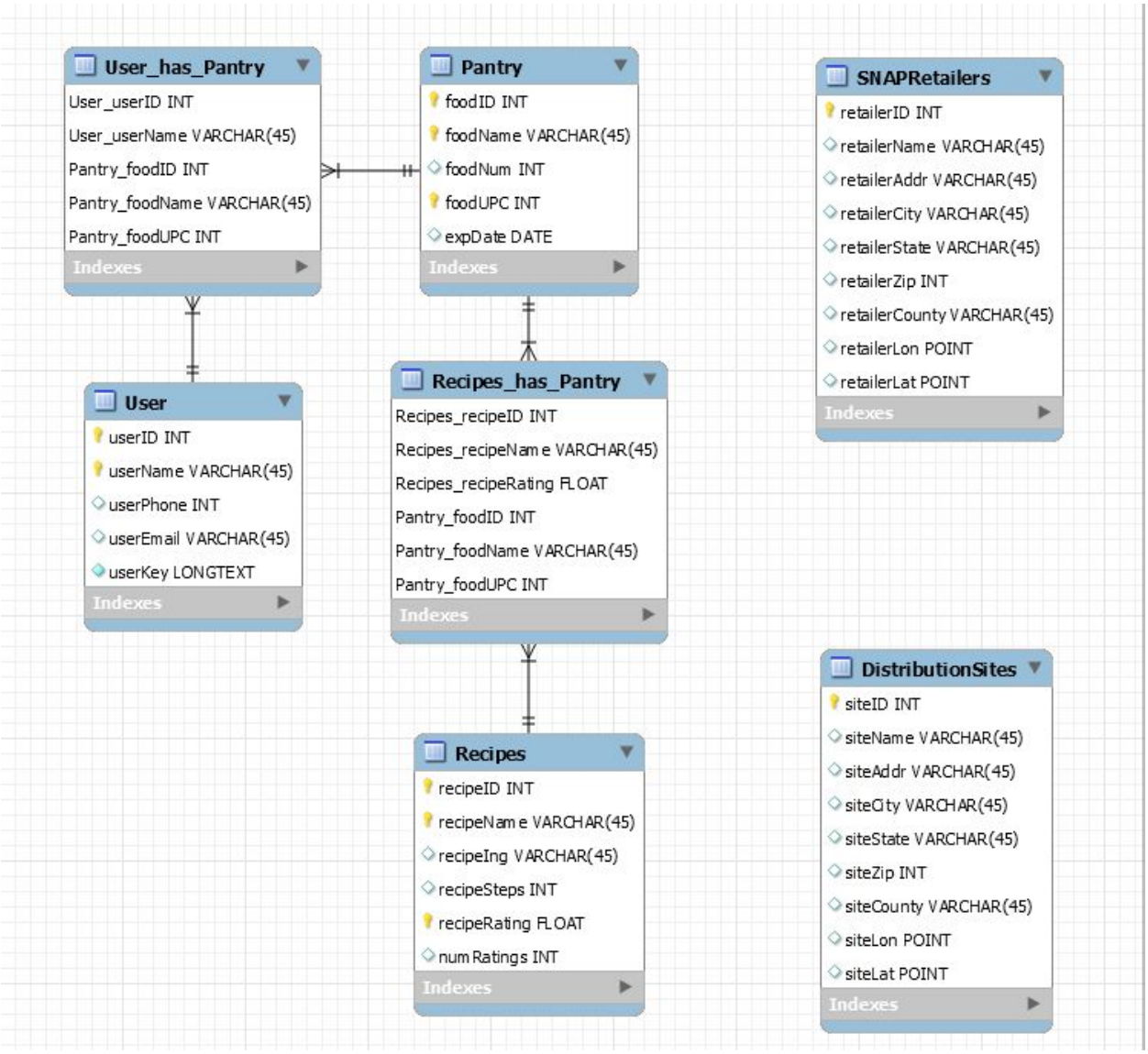
userID(PK)	userName(SK)	userPhone	userEmail	userKey

(Key: PK = Primary Key, SK = Secondary Key, FK = Foreign Key)

Server

- The Database will need to be hosted on a server of some description. For initial testing, we are going to build a local server and try to connect it from another device. After that, we will build a MySQL server on our own, and do further testing.

The Diagram



Use Case: Administering the Database

Mike needs to do some administrative work on the database, which includes manually updating some information, as well as removing some old data that isn't used anymore. The Admin portal doesn't support this functionality, so he will have to access the database directly. He can do this with the MySQL Shell, or MySQL Workbench. Using either one of these tools, he simply enters in his admin credentials, and queries for the data he's looking for, sending delete or overwrite commands as necessary. He can also manually add data here if the admin portal is down, or if he needs to edit the information that isn't shown in it.

Non-Functional Requirements

Maintenance

- This database should be easy to maintain, it is designed for long time use. The tables in the database can be modified and transformed quickly. The whole database system will be able to import and outport between different machines.

Environmental Requirements

Offline Ability

- Some of the data stored on the database will need to be stored on the device locally. The reason for this is because we need to allow our service to work during periods where there is no internet connection available, which we can achieve by storing some of the data locally.
- Seeing as an internet connection will be required to get the app in the first place, it will be safe to assume that the user will be able to at least perform some of the setup tasks they will need to do before using the app and that they will be able to browse the recipe list before service is lost. Because of this, we will elect to only store recipes that the user has "Favorited" to their device.

Analytics

The Analytics portion of the project will collect data on how the app is used through Google Firebase Analytics.

Functional Requirements

Data Collection

- Data will be collected using the Google Firebase framework, which allows data points to be collected using “Event Listeners”, which “Listen” for a certain action to be performed (such as pressing a button or entering text), which triggers a response.
- Data Points to be collected:
  - How often the app is opened/started by a user
  - Location Data for users
  - The type of device used (Android, iOS, etc)
  - How often users select a certain cuisine type
  - How many times each recipe is opened
  - Items added to the virtual pantry
  - How often items in the pantry get used
    - Additionally, if those items were part of a food box or not
  - How often a user rates a recipe/what rating they give

Data Analysis

- Firebase will unpack and keep track of the various bits of information sent by the app through the event listeners, which can then be shown through a console.

Viewing Data

- The collected data will be viewable in the Google Firebase Console through an administrator account, which will have to be created during the development phase.

Use Case: Viewing Data

Mike has to prepare data on how the app is used for Saint Mary’s quarterly report. Fortunately, the Firebase console will give him access to everything he needs, including raw data, charts, and other ways of visualizing the data that he can use in his report. To access this, he simply logs into the Firebase console with his admin credentials and navigates to the Analytics tab, where he will find everything he needs.

Non-Functional Requirements

Performance

- The logs sent out by the Event Listeners that we will be using should take no more than a few seconds to send.
- If they take longer than expected, the event will be cached on the device with a timestamp, and the contents of this cache will be sent out when network conditions allow the events to be sent out within the expected timeframe
- The event logs should not interfere with the normal functionality of the app

Response Time

- The response time of the analytics system will be largely out of our hands once the data has been sent, but it is reasonable to assume that any data collected will be viewable within a few minutes.

Environmental Requirements

Data Usage

- In keeping with our goal to use as little internet-based data as possible, we will attempt to conserve data in both the strings used in the bundles sent with the Event Listeners as well as the number of elements contained within those bundles.
- We will also take extra care to ensure that these Event Listeners are called only when necessary; or in other words, avoiding unnecessary repeat transmissions

Opt-Out

- As part of our efforts to conserve data, and to give users a choice in the matter, we will also allow users to opt-out of data collection altogether. If they do, no event listeners will be sent out by the application at all, and no data will be collected.

API (Application Programming Interface)

The backend API will be the part of this project that does most of the heavy lifting. The API and the Server it is hosted on will handle all of the requests for information that are sent from the Mobile App, and will also allow for the information stored in the

database to be modified through the Admin Interface. The API will need to be hosted on a web server, which will allow it to receive the requests sent by the app and send out Queries to the Database.

Functional Requirements

Methods

- The API will consist of methods that will send queries to the database depending on what the method does. For example, one method will have to search for and return a selection of recipes based on the ingredients specified by the function call, while another one will have to check the login credentials of the user.

Server

- The API will need to be hosted on a web server, which will allow it to receive requests from the app. For testing purposes, we will host the API locally on one of our personal computers, but it should be able to work with any major hosting provider.

Non-Functional Requirements

Simplicity

- For the sake of ease of maintenance, we will attempt to use as few functions as possible without making the API confusing or clunky to use.

Intuitive Use

- To make the API easier to use for other programmers, we will make all of our function and field names intuitive; that is, the names will describe what the function does or what the variable is for.

Environmental Requirements

Accessibility/Cost

- Since this project is being developed for a Non-Profit organization, we need to keep costs down as much as possible to help make sure the organization (either Saint Mary’s or Feeding America) can afford to run it for an extended period.

Potential Risks

As with any application and with any business venture, there will be risks associated with them. The risks for this project will not have a major impact on society but still, need to be considered. The major risks for this project are as follows:

- App Acceptance
- Application Maintenance
- Privacy Risks
- Overcomplexity

Application Acceptance

The first major risk for the project will be app acceptance. As with any app that hits the market the user is the one that decides whether the app succeeds or fails. This app will be very helpful for SMFB but the user might not like to mess with an app every time they go to the food bank. They also might prefer another recipe app and don't want to change apps or they think the app is too intimidating to use. Either way, the app will need to be supported and given enough recommendations to be adopted. The accessibility and functionality of the app particularly those in rural areas might just be what sets the app apart.

Application Maintenance

The second major risk of the project is application maintenance. The maintenance is going to cost some money to keep the app running and future proof. This includes stuff like paying for servers to host the databases, paying for the use of the APIs on a big scale, and lastly paying people to manage the backend. All this maintenance along with having people manage the app can be a hassle but if the app is a hit; It will be a sound investment.

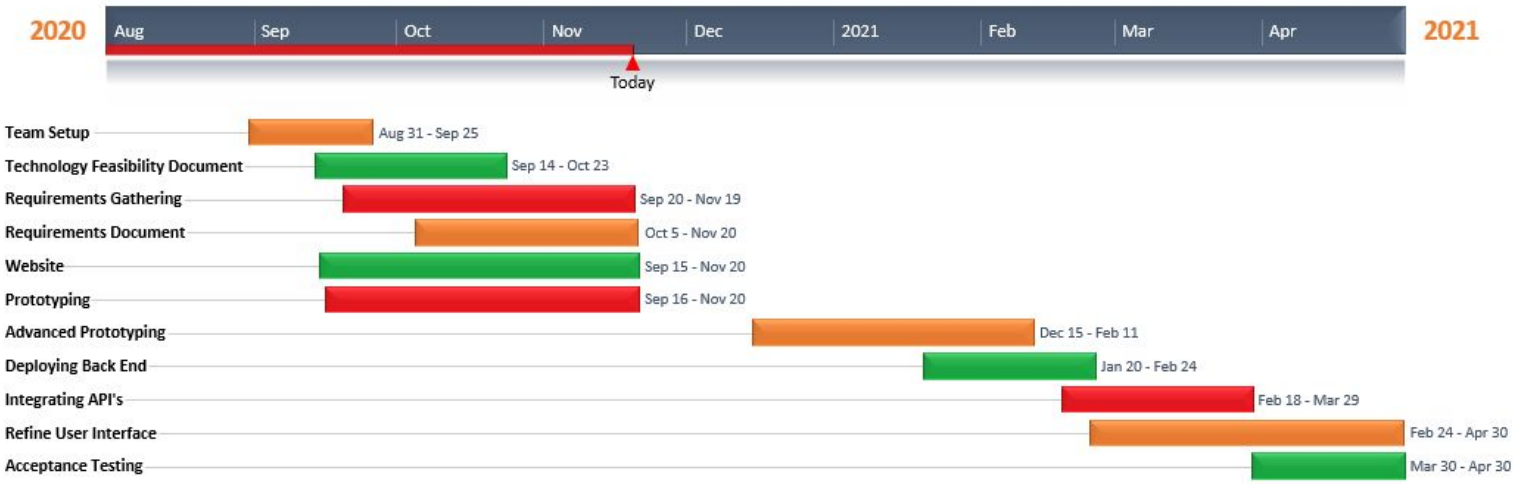
Privacy Risks

The third major risk comes with privacy. We will be using the app to gather information about the various recipes usage. Along with the surveys can be a touchy subject to those using the app. The application will have an opt-out option about data collection, however, passwords and usernames will still have to be stored to access the user pantry and other fancier functions of the app. If a data breach would happen it might lead to passwords being taken. Of course, this can be mitigated by the users using unique passwords, but the risk is there for those who reuse passwords.

Overcomplexity

The fourth major risk is that the project will become over-complex. Some of the features like navigation will not be used often or be difficult for some of the users to use effectively. Since the app has to work offline it will also make some of the features not usable until a connection is established. This is not necessarily a bad thing but might make the app bulkier and more complicated for users to use.

Project Plan



Project Inception

In the earliest stages of development, our primary goal will be to prove the application is feasible to develop. This also includes stuff like setting up team documentation and working on technical documentation. Our early milestones are:

- **Team Documentation**
- **Team Website**
- **Team Introduction**
- **Requirements Gathering**
- **Technical Feasibility Document**

Requirements and Prototyping

In the middle stages of development, we finish our requirements gathering and start with requirements documentation. We will also start building prototypes and overall start getting the project ready for next semester. Along with presenting the design to our

mentors and sponsors. This is where we currently are and will be done within the month.

- **Requirements Documentation**
- **Prototyping**
- **Learning the Technology**
- **Design Review**

Development

The future goals of the project are to continue to prototype and slowly add more features to the app and further enhance our software design. Developing the front end and the back end to an acceptable level. Lastly, we will start working on the MVP and application acceptance.

- **Prototyping**
- **Obtain Recipes for Database**
- **Host and Deploy Back End Functionality**
- **Implement APIs**
    - **Google Maps**
    - **Barcode API**
    - **Firebase Analytics**
- **Refine User Interface**
- **Acceptance Testing**
- **Stretch Goal Development**

This should outline a schedule of the milestones we have accomplished so far, what we are working on, and what we are working towards in the future.

Conclusion

Fridge Filler aims to create a mobile application that can serve as a go-between for the food banks and the people and families that use the food banks. In this way, families can cook with the ingredients they have through simple, easy-to-follow recipes. That will help the families use more of the food given to them.

Our mobile application and website will help solve the food efficacy problem. The app will provide simple, easy-to-follow recipes that will help consumers to use the food given to them. It will also contain a Virtual Pantry to help keep track of what food the user has

and helps recommend recipes based on that. Additionally, it will also include information that will help users navigate to both food bank sites and food retailers, and will also feature analytics about the way the users use the recipes and what foods are getting used where, which could potentially allow Saint Mary's Food Bank to improve its service.

To implement this functionality, we have decided to start with the front end and back end to design our app. These parts will be designed separately and will be connected up later in the development cycle.

For the front end, we have two main things to design: the mobile app and the admin portal. The mobile app version of this project is the main front of the app. In the mobile app, all user functionality will be present. The administrator portal will alter the information on sites and recipes provided to the food bank app users.

For the backend, we have three sections to focus on: database, analytics, and API. The database is the base of our application and website. It supplies us with the data and sends data to users and admins. The Analytics portion will collect data on how the app is used. The API will be used to handle all of the requests for information and allow for the information stored in the database to be modified through the Admin Interface.

So far, we have made good progress on the planning for each of these parts, and have even started working on a prototype. We have also analyzed some of the potential risks we may face, both during development and after release. We got a rough schedule to follow and we believe that this project can be completed successfully.