Version 1.1

Fridge Filler Design Document Final

2/15/21

Team:
Shangyi Dai
Jonathan Derr
Travis Flake
Gage Gabaldon
Zhibang Qin

Project Sponsors:
Dr. Richard Rushforth
Sean Ryan

Mentor:
Sambashiva Kethireddy

Overview

This document outlines the implementation plan for the Fridge Filler project. This will include what we are going to do to get the app working and how the project will proceed in the future.

Table of contents

Introduction 3

Implementation Overview 4

Architectural Overview 5

Module and Interface Descriptions 8

Implementation Plan 14

Conclusion 15

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 sponsors, Richard Rushforth and Sean Ryan are looking to solve. Partnering with Saint Mary's Food Bank (SMFBA), They want to better feed the people the food bank is serving. The process of delivering and managing food for families all across the country is no easy task and one that is vital to continuing to help those in need.

The general workflow of the 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. 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.

This is where our group. Fridge Filler, comes in. 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. The main features of the app are:

- 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.

This app has gone from concept to starting to implement the project. In this document, we will be outlining the design of the project and what we will be using to bring the project to life. Leading to a template that we can use to help focus the timeline of the project and to concretely take the project to reality.

Implementation Overview

Going from concept to implementation requires us to give an overview of our implementation. This is to help solidify the concepts we will be using and to help to give a brief overview of the needed technologies of the project. To make sure that all the functional requirements are met and help bring the app together.

Client-Server Approach

We will be using a client-server approach for our application where the app gets information from a server to show to the user and also sends data back to the server.

Javascript and other web languages

The project is mostly going to be coded in Javascript, CSS, HTML, and TypeScript. Javascript and Typescript will handle most of the API calls, dynamic CSS, and other logic of the app. HTML and CSS will handle the look of the app.

Front End

Ionic and Angular Frameworks

The app is built-in Ionic and Angular frameworks, which will serve as our foundation for the user experience. As Ionic gives us good app templates that give us nice-looking web pages along with many different technologies to help build our app for production. Angular will help with the backbone of the app and will help with routing, dynamic CSS, and other safeguards and logic for the frontend.

Capacitor
Capacitor is a native runtime that will help build our app into multiple different platforms. So we can have support for Android and IOS. Capacitor also has many useful plugins to help interface with the device such as storage and notifications.

Google Maps API and other APIs
We will be using Google Maps API for the map of the distribution sites. Google barcode API. We will be using PDFMake to convert our recipes into printable pdfs. Lastly, we will be using Firebase for the analytics of our app.

Back End
Fridge Filler API
We will be using our own custom API to help serve the app with recipes and to manage access. We will be using Node.js to handle the API calls. These will then access the MySQL database and pull information that is needed or insert information into the database.

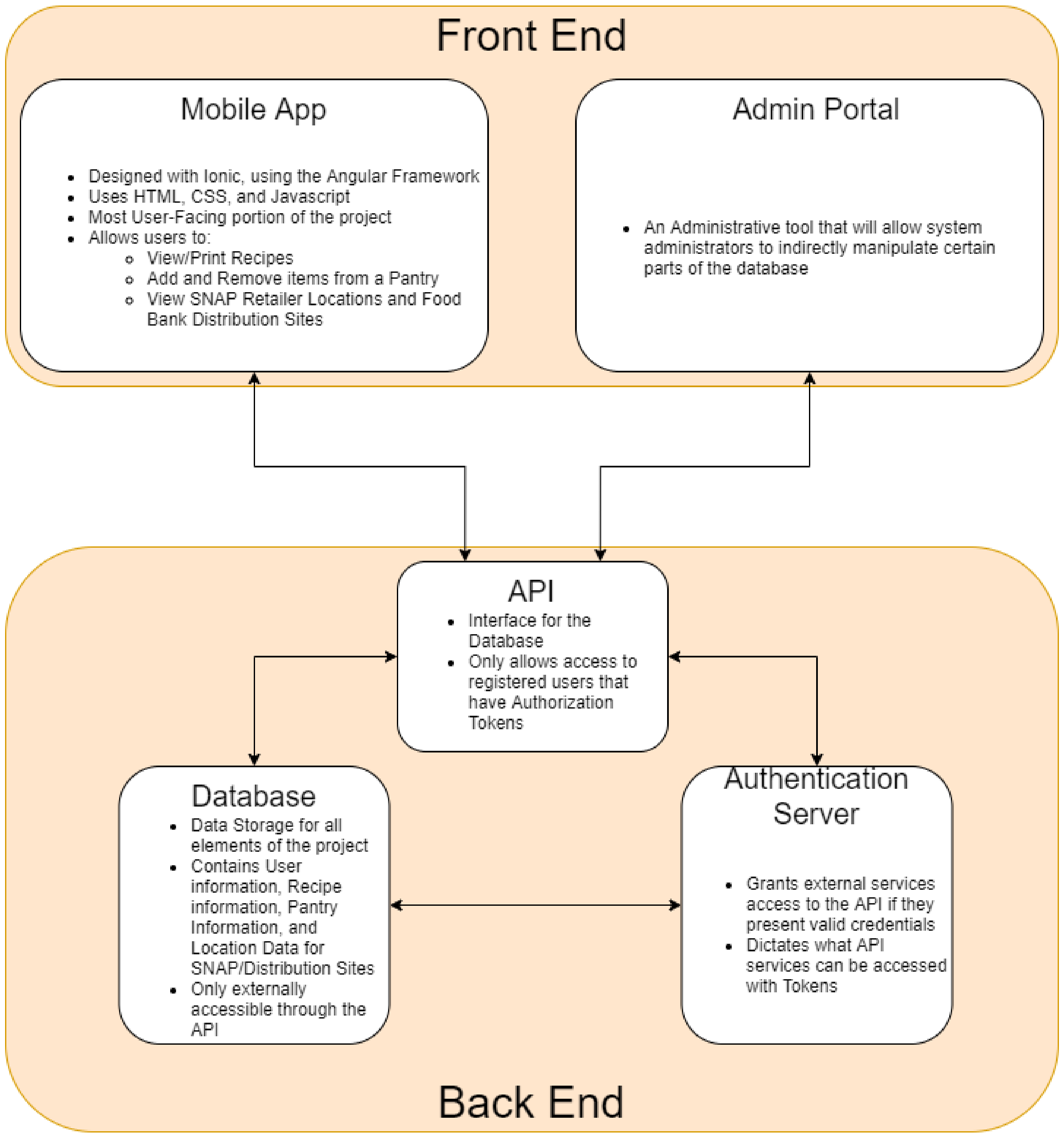
SQLite Database
This will serve as an in-app storage for some of the recipes and the user pantry so this is all accessible offline.

MySQL Database
MySQL Database is an easy to use database allowing for table creation, standard queries, and ease of scalability. It will oversee the storage of most data, including user activity, recipe activity, site activity, and the creation and removal of user pantry.

Apache Server
We will host the fridge filler API and the backend website on the apache server. Along with the MySQL database. This will allow us to retrieve information and edit information that will be used for the application.

Architectural Overview

The following high-level diagram further explores the connection between the frontend and backend of the application, and the inner workings of each aspect of the app, including libraries/APIs, used, language choices for design, and which parts of the backend interface with which components of the frontend.


Front End
Mobile App
Designed with Ionic, using the Angular Framework
Uses HTML, CSS, and Javascript
Most User-Facing portion of the project
Allows users to:
View/Print Recipes
Add and Remove items from a Pantry
View SNAP Retailer Locations and Food Bank Distribution Sites
Admin Portal
An Administrative tool that will allow system administrators to indirectly manipulate certain parts of the database
API
Interface for the Database
Only allows access to registered users that have Authorization Tokens
Database
Data Storage for all elements of the project
Contains User information, Recipe information, Pantry Information, and Location Data for SNAP/Distribution Sites
Only externally accessible through the API
Authentication Server
Grants external services access to the API if they present valid credentials
Dictates what API services can be accessed with Tokens
Back End

Front End

Administrative Portal

The Administrative portal will be responsible for facilitating the viewing and updating of data within the database, specific user information, food distribution site information, and recipes, allowing for both editing, creation, and deletion of these items. The information the administrator portal will be editing will be hosted on the MySQL server, and as the server is also communicating with the application, the data changed by administrators through this portal will also be changed within the application. On page features for the administrative portal will be developed using HTML, CSS, and Javascript, as the simple nature of the needs of the admin portal did not require an advanced framework for development.

Mobile Application

The mobile application will be the user-facing view of this project. This is where users will log in, access their digital pantry, navigate to food banks, search for recipes that use ingredients they specify or own, and print the recipes that interest them. This application will be developed within the Ionic framework, and features will be designed using HTML, CSS, and Javascript (in our implementation, specifically AngularJS). Google Barcode API will be used for scanning of UPC/Barcodes to enter items into the digital pantry, and Google Maps API will be used to generate google maps directions to nearby food distribution sites. For the printing of recipes, the PDFMake API will be used to format recipe data shared from the database into a print-friendly, PDF format. All recipe data will be stored in the MySQL database, and GET requests will be used to communicate to the application. Local data for recipes and ingredients will be stored within the app using an SQLite server, which can hold instances of “saved” recipes and ingredients the user adds to the pantry.

Note: UPC Scanning is a stretch goal for development, however, we feel it is feasible enough to have prepared for the main release of the project.

Back End

Database

The database for the application will store all of the information that the app needs to function properly. This includes user information, recipe steps, and ingredients required, Foodbank and SNAP retailer locations, and most importantly, Pantry contents for each registered user. The database will be implemented using MySQL and will be hosted on an external Linux server. The database will be indirectly mutable by both regular users and administrators through the API we’ll be building, which will ensure that the database is not accessed directly and will allow us to control access to the data stored within it. Users will be able to change the ingredients they have in their pantry as well as their contact information (optionally), while administrators will be able to add, remove, and update recipes, change what ingredients are included in food boxes, as well as adding or removing food bank locations if necessary through the Admin portal.

API

The API will be the primary point of contact for the database and will be used by all external components of the project. It will be hosted on an Apache HTTP server on the same Linux server that will be hosting the database and will use RESTful HTTP methods to query the database using any one of a number of predefined methods. This will allow us to control what queries are sent to the database, therefore allowing us to control the flow of data.

Analytics

Analytics on app usage and (possibly) database contents will be done using Google Firebase analytics. This is a free-to-use service that will allow us to both collect and store analytics information, as well as allowing that data to be displayed in an easily understandable way. Multiple recording events will be embedded within certain button click events in the app, which will trigger an automatic update in the Firebase system for the relevant analytic data. Of course, users will be able to opt-out of these analytics to save on data, or if they simply aren’t comfortable with data being collected on them.

Module and Interface Descriptions

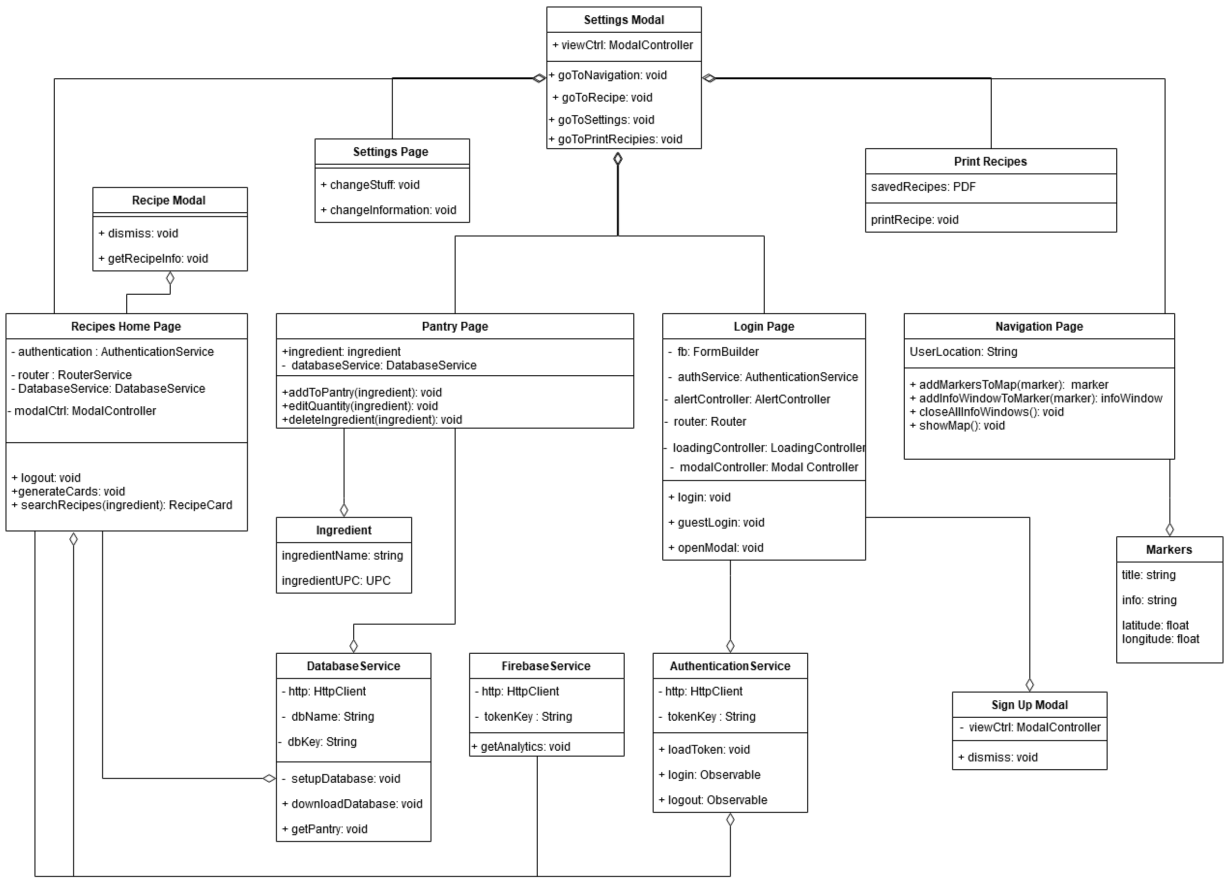
The module components and interfaces are extremely important for a functioning application. As such, each component described here contains a more extensive diagram that further expands on the operations of each component.

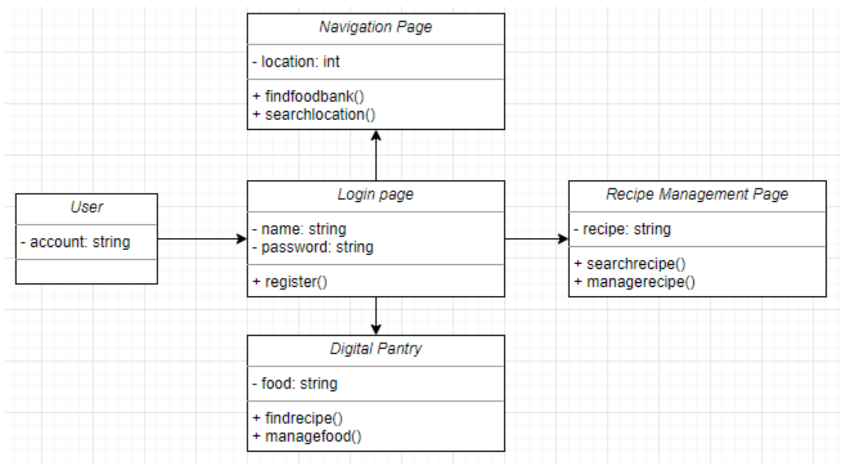
Front End

The front end mainly consists of the mobile application which is designed for users and the administrative portal which is designed for admins.
In the mobile application, users will be able to log in to their account on the login page. After they logged in the accounts stored in the database the data related to their accounts in the database will be accessible which includes their favorites recipes, pantry food.The navigation page will have access to users’ location and show nearby food banks. The digital pantry allows users to store food. Recipe management will show recipes so users can know how to make use of what they have.
In the admin portal, admins will be able to change the data for recipes and sites after logging in. The recipe management page and sites management page will allow admins to get access to the database and make changes.

Mobile Application

The mobile application is designed for users to allow them to make use of their food. Users will be able to check the food they have, search recipes for their food, and find nearby food banks to get food. It is mainly made up of the functions below.





Login Page

The login page will allow users to log in to their existing account or register for a new account. After users click the login button, the userID and userKey will be checked in the database system to allow them to log in to their own account. If a user registers a new account, the account information will be updated in the database.

After logging in, they will have access to the navigation page, their own digital pantry, and recipe management page.

Navigation Page

There will be a map depending on a Google Map which shows the location of the user and shows the nearby food banks. Users can see markers on the map which are managed by Saint Mary's. Users can see the information about markers to help them get the food they need.

Digital Pantry

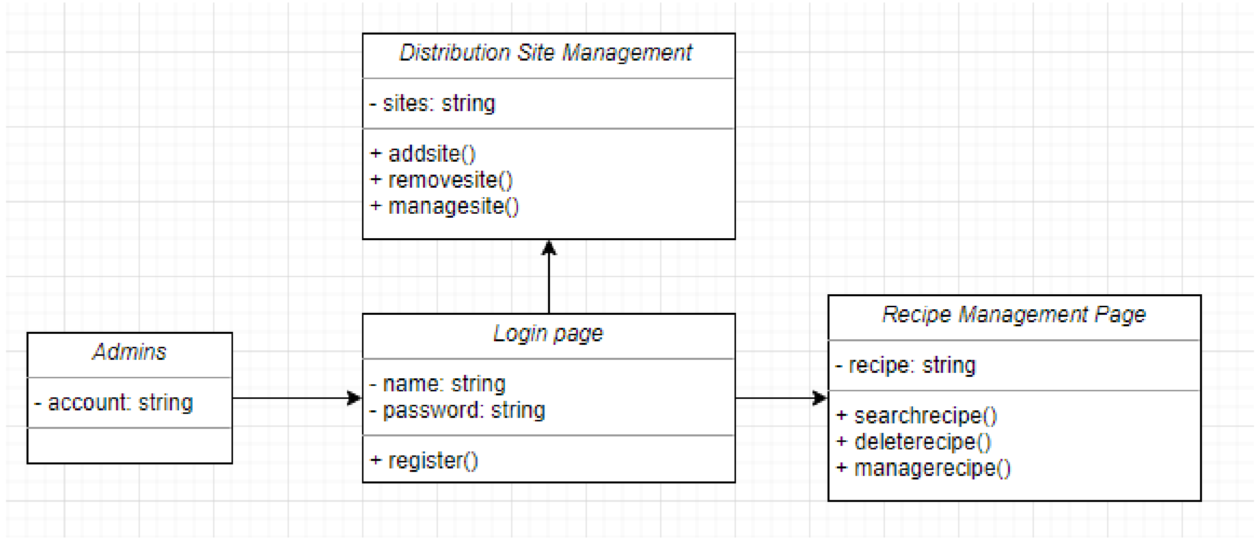
This shows users their digital pantry, which stores the information about their food.
The userID will be used to access their own pantry, which stores their foodID in the database. Then users can see what food they have, the date the food added, and the recipes related to the food to help them decide what to cook.
Users can add new food by entering the food name or scanning the UPC code and deleting existing food. Then the information will be updated in the database.

Recipe Management

The userID will be used to access their stored recipes in the database, then users can view their favorite recipes.
Moreover, if users want more recipes, they can search recipes by filtering them by cuisine type, time to make, difficulty to make, and ingredients.
Users can open a recipe to see more information which is stored in the database. They will be able to see the detailed information of the recipes, add recipes to their favorite, rate the recipes and print them out.

Admin Portal

The admin portal is designed for the administration to manage the database. They will log in to their manager account in the admin portal and then have access to the management pages.



Login Page

The login page will require them to log in to their account. Admin can also use Google account to log in if it has access. After logging in, admins will have access to the food distribution site page and recipe management page.

Food Distribution Site Page

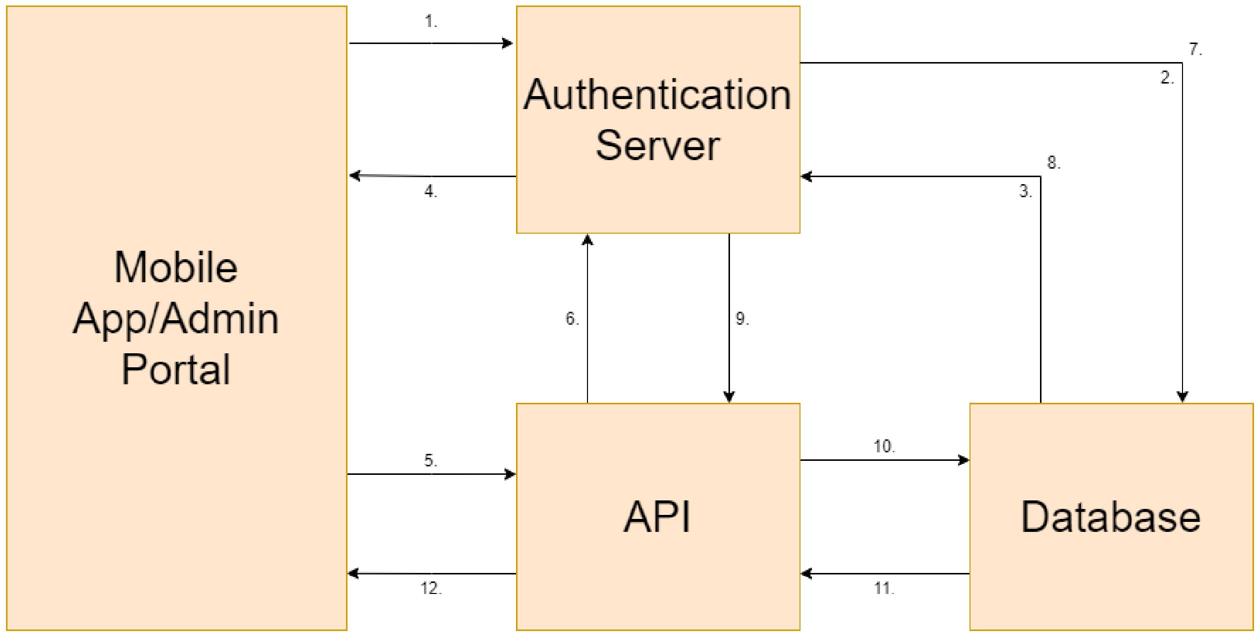
On this page, admins will see the list of the food distribution sites.
The information of food distribution sites will show up and admins can view the information of them or search for a certain site.
Admins can make changes to the detailed information of the sites, add new sites or remove sites.

Recipe Management Page

On this page, admins will have access to the database of the recipes.
Recipes will show in cards containing pictures, basic information, and ratings.
Admins can manage the detail of recipes, add recipes or delete recipes, the information will be updated in the database.

Backend

The diagram below outlines the back end portion of the project and the data flow between the different components.



The steps for using the Back end are as follows:

1. On login, send a request for an Authentication Token to the Authentication Server.
2. The Authentication Server queries the database to check the credentials included in the request.

3. The database returns the necessary information
4. The Authentication Server generates a Token based on the credentials of the user and their permissions, and sends it back to the service that made the request
5. The service that wishes to access the backend sends a request to the API, with the token that was provided to it by the Authentication Server
6. The API sends a request to the Authentication Server to check if the token is valid
7. The Authentication Server checks the database to verify that if the token is valid
8. The database returns the relevant information
9. The Authentication Server sends a response to the API with an Affirmative or a Negative
10. The API sends the specified query to the database
11. The Database returns the results of the query
12. The API converts the results of the query to a JSON string and sends it to the requester.

To allow the back end to work effectively, several components are needed to provide the needed functionality and maintain data security. These components will be explained in further detail below.

Authentication Server

The Authentication server will give a token to the app, check the token sent from API, and send back the approve/deny result to API, then the API will send SQL query to the database to get the result and return it to the app.

The transition between all the steps shown above is using HTTPS which makes them more secure.

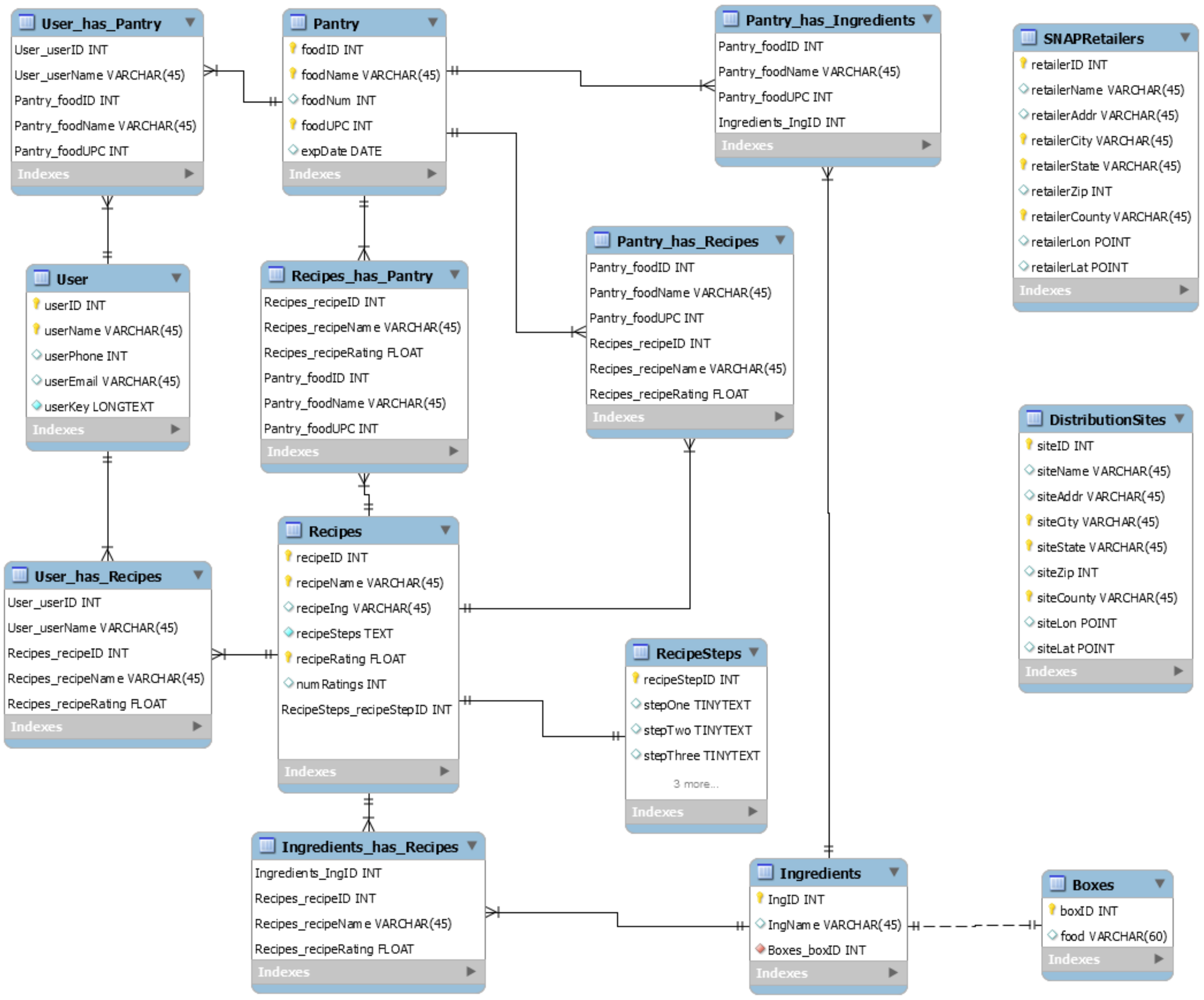
API

Our custom API is designed to help serve the consumer with recipes and to manage access. By using API instead of connecting to our database directly will make data transport safer.
We will use Node.js to handle the API calls. These will then access the MySQL database and pull information that is needed or insert information into the database.

- Get/Post/Delete/Put: The API will have get and post modules that will allow our API to do these things: recognize the request from our application, send SQL commands to our database, get a response from our database, send results to the front-end.

MySQL Database

The Enhanced Entity-Object Relation (EER) Diagram below shows the tables in our database and the relationships between them. The schema described by the diagram contains two different types of tables, Content Tables and Relational Tables. Content Tables Store information, while Relational Tables maintain relationships between two or more tables to allow easier access to the information contained in them.



User Table

This table is a content table that contains all the information about users, it supports user signup and login features. It has a connection with the Pantry table and Recipes table.

- userID: The userID is an identifier, it allows the user to access corresponding User_has_Recipes and User_has_Pantry tables.
- userName: The userName will be displayed after login. We will only require a first name to minimize the amount of data we will be responsible for protecting.
- userPhone: The user's phone number. This is optional.
- userEmail: The user's email address.

- userKey: The password which is necessary for logging into the application and getting authorization keys.

User_has_Recipes Table

This table is a relational table that is created by many to many identifying relationships between User Table and Recipes Table, it supports the user's favorite recipe and recipe print feature.

The detailed information stored in this table is the connection between the User Table and Recipes table. For example, User A has three favorite recipes, and this table will store User A’s id and the three recipes’ ids.

Once a user adds one recipe to his/her favorite list, this table will add the corresponding recipeID. If a user wants to print a recipe, this table will provide recipeID to help to search.

User_has_Pantry Table

This table is a relational table that is created by many to many identifying relationships between User Table and Pantry Table, it supports the user's pantry feature.

The detailed information stored in this table is the connection between the User Table and Pantry Table. Once a user wants to add food to his/her pantry list, this table will add the corresponding foodID. It helps to show data in the pantry table.

Pantry Table

This table is a content table that contains all the information about foods, it supports the UPC scanning feature. It has a connection with the Recipes table and Ingredients table.

- foodID: The foodID is an identifier. It is a primary key.
- foodName: The name of the food, it will be shown in the pantry feature.
- foodNum: The number of food, it will be shown in the pantry feature.
- foodUPC: The UPC code for this food, will be used for the UPC scanning feature.
- expDate: food’s expiration date, it can be used for expiration date notification feature.

Pantry_has_Ingredients Table

This table is a relational table that is created by many to many identifying relationship between Pantry Table and Ingredients Table. It supports the pantry table.

The detailed information stored in this table is the connection between the pantry table and the ingredients table. For example, there are two elements in the pantry table, and this table will store the pantry id and the elements’ corresponding ingredient ids.

It allows us to store more specific information about ingredients.

Pantry_has_Recipes Table

This table is a relational table that is created by many to many identifying relationships between Pantry Table and Recipes Table. It supports a recipe searching feature.
The detailed information stored in this table is the connection between the pantry table and the recipe table. For example, it will store the pantry's id and corresponding recipes' id.

Once a user wants to search for recipes according to his/her pantry list, this table will provide the corresponding recipe information for the results post.

Recipes Table

This table is a content table that contains all the information about recipes. It supports recipe searching and printing features. It has a connection between the User table, RecipeSteps table, Pantry table and Ingredients table.

- recipeID: The recipeID is a primary key and it is an identifier.
- recipeName: The name of the recipe, it will be shown in the recipe searching feature.
- recipeIng: The ingredient of this recipe, it will show in recipe detail.
- recipeSteps: The number of steps, it will show in recipe detail.
- recipeRating: The total rates of this recipe, will be used for showing the recipe's average rating feature.
- numRatings: The number of ratings, will be used for calculating the average rating.

Recipes_has_Pantry Table

This table is a relational table that is created by many to many identifying relationships between the Recipes Table and Pantry Table. It will support the recipe searching feature.

The detailed information stored in this table is similar to the pantry_has_recipes table. The detailed information stored in this table is the connection between the pantry table and the recipe table. For example, it will store the recipes' id and corresponding pantry's id.

RecipeSteps Table

This table is a content table that contains the steps of the corresponding recipe. It will show in recipe detail. It has a connection with the Recipes table. It supports the recipe detail and recipe print feature.

- recipeID: This a primary key and it is an identifier.
- stepNum: This identifies which step of the recipe this entry represents
- stepContent: The actual directions for this step of the recipe

Ingredients Table

This table is a content table that contains all the ingredients. It has a connection between the Pantry table, Boxes table and Recipes table. It supports recipe detail and food detail features.

- IngID: It is a primary key and it is an identifier.
- IngName: The name of the ingredient, it will show in recipe detail and food detail.

Ingredients_has_Recipes Table

This table is a relational table that is created by many to many identifying relationships between the Ingredients Table and Recipes Table. It supports a recipe searching feature.

The detailed information stored in this table is the connection between the ingredients table and Recipes table. It will store the ingredient's id and its corresponding recipes' ids.

Boxes Table

This table is a content table that represents the emergency food boxes handed out by Saint Mary's. It contains a number of preset ingredients that are typically handed out in these boxes, and will allow users to easily add the contents of a box they received to their pantry. It has a connection between the Ingredients table.

- boxID: The boxID is an identifier, it is a primary key.
- foodName: It is designed to show food names in the box list. It helps to show foods in future.

Once a user clicks the box in the app, it will automatically add all the box's foods into his/her pantry.

Distribution Site Table

This table contains all the information about distribution sites. It supports a navigation feature. It has no connection with other tables.

- siteID: It is a primary key.
- siteName: The name of the site, it will show in the navigation detail.
- siteAddr: The address of the site, it will show in the navigation detail.
- siteCity: The city where the site is located, it will show in the navigation detail.
- siteState: The state where the site is located, it will show in the navigation detail.
- siteZip: The Zipcode for the site, can be used for site searching.
- siteCountry: The country where the site is located, it can be used for site searching.
- siteLon: The longitude for the site, can be used for navigation.
- siteLat: The latitude for the site, can be used for navigation.

SNAP Retailer Table

This table contains all the information about SNAP retailers. It supports a navigation feature. It has no connection with other tables.

- retailerID: It is a primary key.
- retailerName: The name of the retailer, it will show in the navigation detail.
- retailerAddr: The address of the retailer, it will show in the navigation detail.
- retailerCity: The city where the retailer is located, it will show in the navigation detail.
- retailerState: The state where the retailer is located, it will show in the navigation detail.
- retailerZip: The Zipcode for the retailer, can be used for retailer searching.
- retailerCountry: The country where the retailer is located, it can be used for retailer searching.

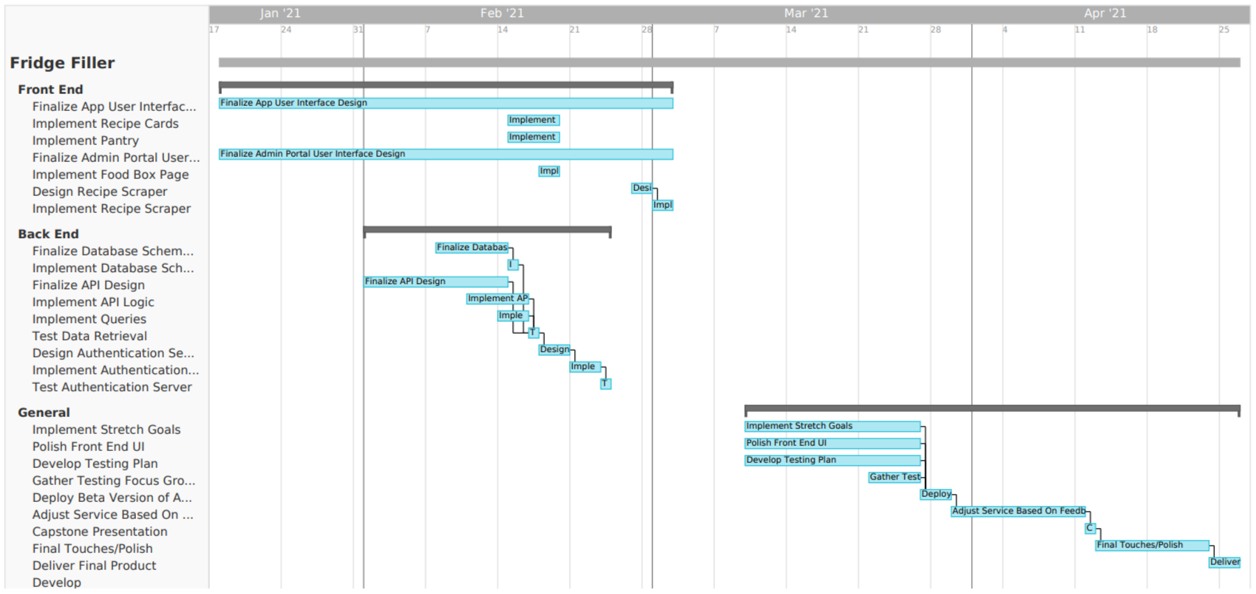
- retailerLon: The longitude of the retailer, can be used for navigation.
- retailerLat: The latitude for the retailer, can be used for navigation.

This table is optional for the front-end. It can just use the information from SNAP Retailer Locator.

Implementation Plan

Currently, our plan for implementation is very open-ended, to allow us to be as flexible as possible with development. Because of our relative lack of experience with most of the tools and frameworks we're using, we are having to learn as we go, which means that unexpected roadblocks could appear at any point. As we learn more, we are solidifying our development plan, but listed below are the general steps we know we will need to take to get our project to a completed state.

- **Alpha Development (Finished by March 9th)**
    - **Complete Final App User Interface Design (February 12th)**
    - **Complete Final Admin Portal User Interface Design (February 12th)**
    - **Complete Final Database Design (February 12th)**
    - **Complete Final API Design (February 12th)**
    - **Implement API Server (February 19th)**
    - **Establish/Test Communication Protocols between the Front End and the Back End (February 19th)**
    - **Implement Authentication**
    - **Test Data Retrieval/Updates from Front End (February 26th)**
        - **Test Recipe Population**
        - **Test Pantry Population/Update**
- **Beta Development (Finished by April 9th)**
    - **Start Implementing Stretch Goal Features**
    - **Polish Front End User Interface Design**
- **Testing/Finalizing (Finished by April 16th)**
    - **Gather Focus Group**
    - **Deploy Beta version of the app to Focus Group**
    - **Tweak Beta based on feedback from Focus Group**



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.

With the help of our sponsors, Richard Ruthford, Sean Ryan, and Saint mary's food bank we hope to help those people served by the food banks. To better use the food given to them and to help Saint Mary's feed more people effectively.
Our solution is a mobile application and website that 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.

The document served as the blueprint for the remainder of this project. We have laid out a detailed implementation and architecture plan for our project, so we can better capture design flaws early on, which will save time in the development step of the project. Using various technologies like MySQL, Angular, Ionic, and various API's we will be slowly integrating and bringing together everything for the app. To support all the features such as recipes, user pantry, navigation, and printable recipes.

The project now has a clear path of development and a plan to get there. We are happy with the progress of the project and excited to continue our work and deliver the application that we want to serve to our sponsors.