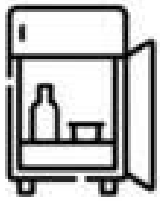
Technology Feasibility Report



Team Fridge Filler
10/21/2020

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 technological feasibility of the “Fridge Filler”. This will include major technical challenges that may arise during the production process, as well as some possible methods that will be utilized to overcome such challenges.

Table of Contents

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 had trouble recommending and suggesting healthy recipes. Along with making sure that food is not thrown away and wasted. As the uncertainty of food that comes through the food banks and the distribution of food products is not uniform. Taking the United States, for example:

- 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 to help with food efficacy. With our sponsors, Richard Rushforth and Sean Ryan, we 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.

These challenges are diverse and can lead to many different solutions based upon what is decided and what is technologically feasible. In the technological challenges section, we will analyze each of these diverse challenges in detail and the rationale for choosing

a particular solution; In the technology integration section, we will be looking at how all these solutions will come together to make a successful project.

Technological Challenges

The technological challenges of the project are mostly in the realm of backend database management and integrating various technologies that will help users to make various food from what's available in their house, in the food bank, and nearby stores.

- **Compatibility:** We will need to create a mobile app that is user friendly and usable on multiple platforms.
- **Backend Storage:** We will need to select a platform to use for our Database. Ideally, this should line up as close as possible to the databases already maintained by Saint Mary's Food Bank.
- **User Interface:** We will need to create the UI for the web-based access portal.
- **Administrative Tools:** We will need to create access tools for the database so that administrators can make changes to the databases.
- **Security:** We will need to secure our database from unwanted access.
- **Analytics:** We will need to select an analytics service that can display various pieces of data in a graphical format to administrators.
- **Formatting Recipes:** We will need to find a way to format the recipes from the database as PDF files/possibly some other web-friendly format.
- **Navigation:** We will need a way to implement navigation to the nearest food bank, which means we will need to choose a Navigation API to support the most users effectively. Should be able to function with either GPS data on the device or zip code.
- **UPC Scanning:** We will need a way to use UPCs to search for food items and add them to the user pantry.

Technology Analysis

In this section, we will explore the issues in greater detail and give some ideas about how to solve the problem. We will first give an introduction to the issue as a whole to better flush out the problems we are dealing with and the desired characteristics we want. Then we will outline the potential alternatives and give the reasons they are a good alternative. Lastly, we will prove that the technology we choose is the most feasible and the best option for this project.

Issue 1: Application Framework

- **Intro the issue:** We need to develop a cross-platform mobile application. There are many ways to make a mobile app, and we are aiming to create a fast application that is the same cross-platform, preferably with a clean user interface. To do this, we have chosen Ionic, with React JavaScript as our JS framework.
- **Alternatives:** There are several options we are investigating for development framework and native development. For cross-platform development, we are choosing between Ionic, React Native, and Flutter. These are very good frameworks but flutter is not as popular and requires us to learn a new language. Ionic on the other hand isn't native and is shown to lose in performance testing to native apps. React Native is based on React but appears to have a much steeper learning curve than our other options, which could inhibit the development process.
    - React Native is a development framework created by the creators of ReactJS, and it’s grown very popular. It uses native code, and while it is popular and based partially on javascript for rendering (namely React JS), we felt the learning curve for this was much steeper than Ionic to produce the same results.
    - Flutter is a Google-made development framework that uses its language dart to create multi-platform apps. Not popular but very good performance.
    - Ionic is easy to use and is based upon HTML and CSS. It is a powerful tool for app development but overall sees worse performance at runtime. Ionic is also open-source and has a large and helpful library of APIs for us to work with.

Feature Comparison	REACT Native	Flutter	Ionic
Ease of Use	2	3	5
Feature Set	4	3	4
Familiarity	1	1	3
Performance	4	4	3
Total	11	11	15

- **Table Discussion:** In the table above, we measured a few criteria for our choice of which mobile framework we chose for our application. Ease of use is important to us, and the Ionic application setup was by far the simplest out of the ones tested. App creation is fast and allows us to get right into design work. As far as features are concerned, Ionic and REACT are fairly similar in contents, and both have a larger library of features that are made accessible to the framework than flutter. The team is notably more familiar with Ionic since it is based within CSS/JS/HTML, where the other top options are entirely new frameworks with their languages. The performance of the frameworks, while measurable, does not ultimately detract from the runtime performance on a mobile app of the scale we are developing when comparing Ionic and REACT Native, and flutter only performs a bit worse than these options. Ultimately we felt Ionic had the best performance in these categories, especially because of its wide range of features for UI, and the ease of development, which we suspect will allow us to spend more time refining the less visual components of the application.
- **Chosen approach:** Ionic
- **Proving feasibility :** To prove the feasibility of Ionic app development, we have installed Ionic, and will experiment with the intuitive UI libraries to create an experience that is accessible to the less technologically literate users, and moving forward development in Ionic will ease the curve of learning new technology stacks, facilitating more in-depth work on the features of the app. The Ion UI libraries will lessen the burden of graphic design on the team, meaning our demos will be more fleshed out technically.

Issue 2: Database Platform

- **Intro the issue:** We will need to select a platform to use for our database. Ideally, we will develop a database system and let it work well with the frontend application and backend control portal. There are many platforms for us to choose from: MySQL, MongoDB, PostgreSQL, etc. Our goal is to choose a platform that is powerful and familiar for all teammates.
- **Alternatives:** There are several platforms for us to use:
    - MySQL is an open-source relational database management system. It is one of the most widely used relational databases, powering nine out of ten websites around the world, and about half the developers prefer it.
    - MongoDB is a cross-platform document-oriented database program. It is a NoSQL database that stores data as JSON-like documents and uses the MongoDB query language for access.
    - PostgreSQL, also known as Postgres, is an open-source relational database management system. It has many numerous features such as native partitioning, parallel query, support of foreign data wrappers, powerful JSON features, streaming and logical replication, and the availability of many open source tools for HA, backups, and monitoring.

Feature Comparison	MySQL	MongoDB	PostgreSQL
Usability	4	3	3
Functionality	3	3	3
Familiarity	5	2	2
Total	12	8	8

- **Table Discussion:** All of the choices above have similar features and serve the purpose of storing data. All database management systems listed are rather user oriented and highly usable from the perspective of a database administrator. The functionality of each DBMS will be nearly identical, as the relations needed for the app we are developing are rather simple in design, and do not require advanced algorithms to be applied to increase efficiency between interacting databases. As far as familiarity goes, the team is most familiar with MySQL, as

we all have a moderate amount of experience and knowledge of MySQL from the CS345 database design course, and as such we feel most comfortable with it.
- **Chosen approach:** MySQL
- **Proving feasibility:** Going forward we will be developing our database in MySQL, and connecting it to the frontend app that we are creating in Ionic, aiming to link user, recipe, and pantry databases to the application for prototyping. We will host all the databases we develop in MySQL. We will develop a pantry database for users, a site database for admin. The pantry database will be able to contain what users have and the site database will hold the information about the sites.

Issue 3: Administrative Portal

- **Intro the issue:** For this project, we need to design an administrative access portal for system administrators from SMFBA to edit food distribution sites and other user data. We want this portal to be simple and accessible to minimize the risk of edits that will cause problems.
- **Alternatives:** The backend will be a website, and we have a few options for this. We could use a Javascript framework, such as react or angular, however, the issue at hand may be simple enough to not require an intricate framework, as the feature will not see wide-scale use. Rendering from ReactJS is a great feature, but we likely won't need the intricacy of a javascript framework to make this happen.

Feature Comparison	JS Framework(React, Angular, etc.)	Simple JS/HTML/CSS
Functionality	5	4
Simplicity	2	5
Total	7	9

- **Table Discussion:** JS frameworks are collections of JavaScript code libraries that provide developers with pre-written JS code to use for routine programming features and tasks. However, owing to a large set of powerful features, most frameworks tend to be bulky regarding functions and codebase. Meanwhile simple JS is a client-side language and can also satisfy what we need in the

administrative portal. What we are going to use is simple javascript for it is relatively simple to learn and implement. We will be using that.

- **Chosen approach:** Basic Javascript
- **Proving feasibility :** Going forward we will design an administrative portal with basic HTML/CSS/JS, and going forward will link it to our databases for user data and food bank location/recipe data to validate our choices being accessible and editable from the administrative portal. If needed, this can be changed to ReactJS, but the complexity of this issue has yet to present itself in a way that demands a framework.

Issue 4: Database Access Tools

- **Intro the issue:** For the database, we work on, we need to create access tools for the database so that administrators can make changes and the app can access it. The tools should be connected to the remote database system, also work smoothly with the administrator portal. Also, the tools need to adapt to the SMFB's database system.
- **Alternatives:** There are many languages we can choose to write the tools, here're the available languages for us:
    - PHP is a popular general-purpose scripting language that is especially suited to web development, it's the most popular to use PHP with MySQL.
    - JavaScript is now also being used on the server, with the engine called: Node.js. It's a lightweight, interpreted programming language and well known as a scripting language for Web pages, also many non-browser environments.
    - Java / JSP is a server-side programming technology that enables the creation of a dynamic, platform-independent method for building Web-based applications. JSP has access to the entire family of Java APIs.
    - Python has the Python standard for database interfaces which is called Python DB-API, and it supports a wide range of database servers.
    - Ruby has the Ruby DBI which is the database-independent interface for Ruby, it provides an abstraction layer between the Ruby code and the underlying database, allowing you to switch database implementations easily.
    - ASP.NET - usually with the C# programming language. ".NET" is a developer platform made up of tools, programming languages, and libraries for building many different types of applications. "ASP.NET" extends the .NET developer platform with tools and libraries specifically for building web apps.

Feature Comparison	PHP	JavaScript	Java/JSP	Python	Ruby	ASP.NET
Functionality	4	4	4	4	4	4
Usability	3	3	3	3	2	3
Familiarity	3	5	2	2	1	1
Total	10	12	9	9	7	8

- **Table Discussion:** Of the languages above, PHP, JavaScript, Java, and Python are the most familiar one for us, also it is very common to use these languages to connect with database systems. Because we want to use web portals to control databases, JavaScript is the most suitable choice for us, it can be used on both the server-side and admin side, or it can just stay on the admin side and send requests to the server.
- **Chosen approach:** JavaScript
- **Proving feasibility :** We are going to use JS for the Administrative Portal, as it will be easier to manage the database if JavaScript is also used for accessing the database. We won't directly connect to the remote database by JavaScript and remote HTTP(S) backends are used to provide the client-side apps with the data. Therefore, to provide front-end apps with data from a MySQL database we need to implement a server-side backend and make the front-end apps use it. We have tried MySQL Connector/Node.js and it looks good. It is a free-to-use, open-source database that facilitates the effective management of databases by connecting them to the software. After testing and finding some examples about it we believe it is stable and reliable for the database access in our project.

Issue 5: Database Security

- **Intro the issue:** Because our database will contain information about users, it will need to be secure from unwanted access. We also need to control who has access to the database and who has permission to access it. We want a system that will be secure from unauthorized access.
- **Alternatives:** There are many things to help secure a database some of these options include encryption, access control, and some other security measures that are native to a DBMS. The option of separating the user database from the main database will also work wonders.

Feature Comparison	Encryption	Access Control	Separation of Databases
Prevents Users From Accessing Harmful Data	5	5	5
Controls Who Has Access	1	5	4
Hides Personal Data	5	0	1
Total	11	10	10

- **Table Discussion:** The table shows that each one of the ways to secure databases has its perks and cons. However, these are all parts of a secure database system and must be used in conjunction with each other to be effective.
- **Chosen approach:** Encryption, Separation, Access Control
- **Proving feasibility:** We will be using a combination of encryption, separation, and access control. We will be encrypting the user information so that the user information can be private and no one knows what it is. We will also use the Access Control to control who can access the database and make sure that only the administrator has access to the database. The Separation of Databases makes sure that all our different databases are self-contained and don't interfere with each other.

Issue 6: Application Analytics Service

- **Intro the issue:** We will need a way to implement basic analytics to track the usefulness of an application. The analytics we will aim to collect may include the popularity of features within the app, as well as what audiences utilize which portions of the app. The analytics tool should also specifically aim to display analytics in some graphical format.
- **Alternatives:** For this issue, there are several strong contenders for analytics that have great coherence with the Ionic framework.
    - Google Firebase/Firebase Analytics: Google Firebase Analytics has a paid option but is largely free and the paid services are designed to assist

large scale businesses. Many of the features for the paid version target monetization of pages, which is something we do not need. Firebase also has a graphical representation of data built-in, which meets our requirements for data visualization.

- AWS X-Ray: Amazon X-Ray is only free on a smaller scale, and an analysis of over 100,000 traces removes you from the free plan of AWS. The cost of this is low, but should the app be further developed after we handle it and grow in user base this could be a problem for SMFB.
- Datadog: Datadog is a great tool, but also comes with a steep price for large-scale development, which is an issue for an app developed for a non-profit food bank, as we aim to develop something that is not going to produce future costs for the Saint Mary's Food Bank.

Feature Comparison	Google Firebase	AWS Xray	Datadog
Functionality	5	4	5
Accessibility	5	3	4
Cost	5	3	1
Total	15	10	10

- **Table Discussion:** This table examines a few key requirements of our analytics software choices. We need it to have high functionality for SMFBA to create user flows they wish to monitor for the food bank users. Datadog appears to offer more detailed breakdowns of data and user flow setup than X-ray and Firebase. The accessibility refers to how easily usable the data is, and while it has a learning curve both Firebase and Datadog have quite nice graphical data analysis interfaces for users. AWS has a slightly less polished UI but shows the data accurately. The cost requirement was more unique to this feature, as most well-maintained Analytics software has subscription costs. Google Firebase offered by far the most coverage for its price, so it scales higher than the other two options, which require payment for a much smaller count of users analyzed.
- **Chosen approach:** Google Firebase Analytics
- **Proving feasibility:** Some of the team has used firebase before and firmly believes it can solve basic analytics requests presented by the Saint Mary's Food Bank and will proceed in development. Going forward we will aim to design event listeners and demographic sets that will provide useful data for the food bank

when the app is released, and in testing produce analytics data to show how basic collection will work.

Issue 7: Recipe Formatting

- **Intro the issue:** We will need to find a way to format the recipes from the database as PDF files/possibly some other web-friendly format. This will have to be done programmatically through JavaScript, and fortunately, there are several APIs we could use for this:
- **Alternatives:**
    - PDFKit - PDFKit is an object-oriented PDF generation library that is based on Node. It includes all the basic functionality required for generating PDF documents, but it can also generate vector graphics and encrypt the documents it generates. It is heavily objects and function-based, with function calls used to do most things with the document objects, which could limit our options in terms of implementation and portability.
    - Pdfmake - Pdfmake is another PDF generation library based on javascript. Unlike the other two libraries listed here, it features limited utilization of functions, relying more on attributes to change things about the document. It is also much more feature-rich than the other two options listed here, and also appears to be the most complete. It has a wider variety of standard text and image manipulation tools, and can also generate QR codes, include custom fonts, generate images using the Scalable Vector Graphics format, embed document metadata, and embed watermarks, just to name a few features. This library can also be used on both the server and the client-side of interactions, which would give us more options when it comes to implementation.
    - jsPDF - jsPDF is another PDF generation library-based on Javascript that takes a more object-oriented approach to generate PDFs. Like PDFKit, it is much more object and function-oriented than Pdfmake, with documents existing as objects, which are manipulated using functions. There is not a lot of documentation for this library, so it is difficult to judge how well it will work for our purposes at face value.

Feature Comparison	PDFKit	pdfmake	jsPDF
Functionality	4	5	3
Ease of Use	3	4	3
Total	7	9	6

- **Table Discussion:** Pdfmake leads in functionality, having a greater variety of options to use when creating and formatting documents. It also leads in Ease of Use, as its approach of not relying so much on function calls is likely to make the document creation code less complicated. PDFKit is a close second, and its reliance on functions and rich feature sets may still find some use with this project if we run into any unforeseen issues with pdfmake. jsPDF seems to be the weakest library of this selection, mainly because it's so hard to find easily accessible documentation on it. WIth pdfmake's function-based design, it will be much easier to produce printable recipe PDFs based on the ingredients that users have available to them, which is the main goal of this feature.
- **Chosen approach:** Pdfmake
- **Proving feasibility:** Pdfmake appears to be the best option for us. Its multiple implementations on both client and server-side platforms and its multitude of features will hopefully allow us to be flexible with how we implement the requirements that use this library. Going forward we will design functions that format recipes based on what ingredients users possess to create print-friendly versions of desired recipes.

Issue 8: Foodbank and Store Navigation

- **Intro the issue:** We will need a way to implement navigation to the nearest foodbank. Since the creation of mobile phones, there has been a large number of navigation apps developed, some of the more prominent being Google Maps, Waze, and Bing maps. We require the maps integration to work on the most phones possible and ideally support offline use and pathing that can be done without needing a vehicle.
- **Alternatives:** We considered Waze maps, Google maps, and Bing maps for this facet of the application. All 3 options have rather high-quality mapping capabilities, but out of the 3 Google Maps is the only application that provided pathing for bicycles, which is a rather popular mode of transportation and could alienate users if it was not included as an option. Google Maps also offers offline

functionality, which can help users who need to use the public internet to get the map, but still would like to have it available to them without printing it out.

Feature Comparison	Google Maps	Bing Maps	Waze Maps
Navigation Options	5	3	2
Device Compatibility	4	2	3
Familiarity	5	2	2
Total	14	7	7

- **Table Discussion:** The table above compares navigation options from each of the 3 considered map APIs, factoring in how many modes of travel are offered and how accurate the travel data will be. Google Maps scored highly on this for having a wider offering of travel options compared to Bing and Waze, as well as relatively more accurate travel data for each option. The device compatibility section of the table is to consider how well the map API will work on mobile devices, including deprecated/outdated machines, and new models. Google Maps is one of the oldest map APIs that works with mobile phones and has great continual support.
- **Chosen approach:** Google Maps
- **Proving feasibility:** Going forward we will be requesting access to the Google Maps API, and from there developing a simple “Food Banks Near Me” search tool and a search for local grocery stores in future technological demos to prove our choice for feasibility was correct.

Issue 9: Scanning Food Items to Add Them to the Pantry

- **Intro the issue:** We need to create a method for users to scan in the food items using their barcodes (UPCs) to add food to the user’s pantry to allow good recommendation of recipes. This feature needs to be highly functional and work using device cameras. For this, we are considering the Google Barcodes API, the Barcode Lookup API, and the Cloudmersive Barcode Lookup API.

- **Alternatives:** It is very important for the Barcode API we choose to be portable to the largest range of devices possible, since there is no way we can predict the age or make of user devices, but do not want to exclude those with older phones.
    - The Google Barcode API supports a wide range of devices, and multiple forms of UPC Barcodes (Both UPC A and UPC E).
    - The Barcode Lookup API allows searching of products using their UPC Code or name and has a database of over 200 million items to be searched. It is not limited to only food objects, however.
    - The Cloudmersive API allows the scanning of barcodes to convert them into data, as well as the creation of barcodes to assign them to objects. This feature is not required but could allow interesting features for the foodbank to catalog items without UPCs for ease of access to food bank recipients. A caveat of this is that the Cloudmersive API has a price for apps that call it over 800 times a month, which is below what our app will see use on the scale of upon successful release.

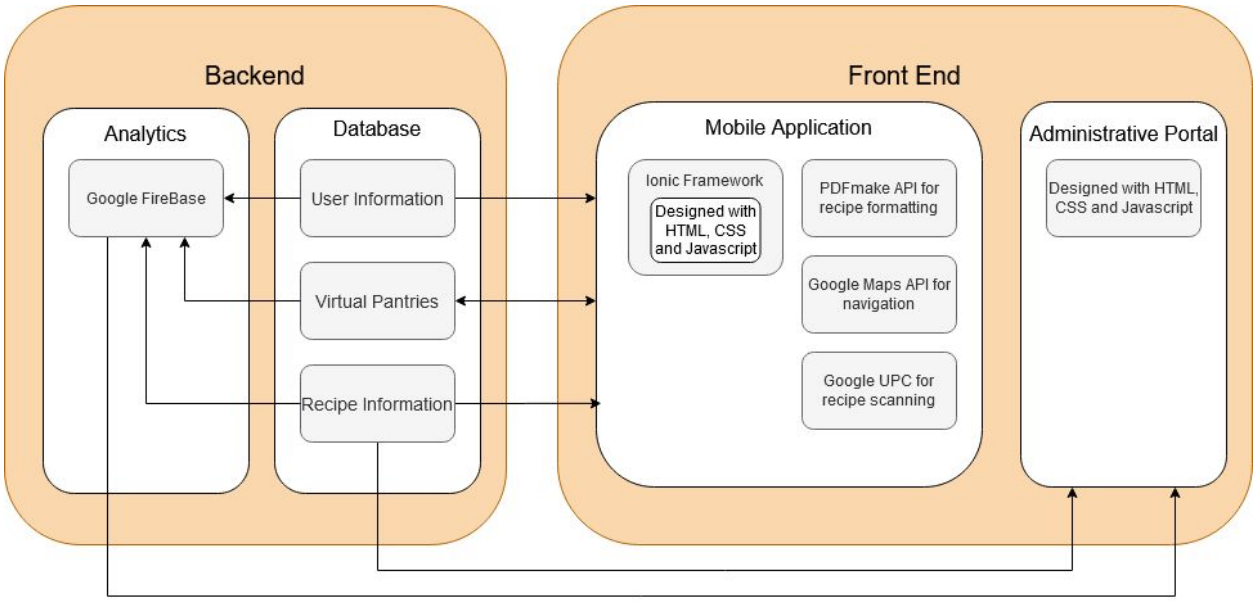
Feature Comparison	Google Barcode API	Barcode Lookup API	Cloudmersive Barcode API
Functionality	5	4	4
Barcode Scanning Features	5	4	4
Accessibility	5	3	2
Total	15	11	10

- **Table Discussion**: This table compares the functionality of the bar code lookup APIs, considering how well the API will work on a range of devices. The Barcode scanning features are looking at features of each Barcode API, and considering the Unique concepts, each has built into the API. Accessibility is the ease of use and compatibility that each API offers, as well as how easily we can offer this service on a wide scale developed project. While the features of Google Barcodes API were a bit lacking compared to the contemporary options, it is widely accessible and highly functional.
- **Chosen approach:** Google Barcodes API
- **Proving feasibility:** While it lacks some of the features we are seeing in the other Barcode APIs researched, this is the most widely accessible barcode API

that we can trust will work on cross-platform applications, and it does not have the problem of being a paid API, unlike Cloudmersive. Going forward we will integrate a barcode search and select feature for our tech demos, in which users will be able to add food items into their “Digital Pantry” for recipe finding.

Technology Integration

Now that we have all the technology in place, the only job that remains is organizing all of that technology to create our final product. The following diagram outlines our plan to create an effective solution for this problem.



The system diagram for the application and supporting technologies.

The mobile application itself is built in the Ionic framework using Javascript, CSS, and HTML, and our backend will use the MySQL DBMS. The app has a database for the recipes that will be used in conjunction with the user’s pantry. This will help to determine what recipes are suitable for the user and what ingredients are needed. We then will use A few different API’s to help with the recipe portion of the app. The UPC API will help to add various food items to the user’s pantry using the barcodes found on each item. We will use the PDFmake to format the recipes to be printed out in the app. Google maps will be great to provide the nearest location to pick up missing food items either from a food site or a local store. Lastly, the Google firebase API will be used to

provide the necessary analytical information about commonly used ingredients, recipes, and information about the app and the user. The app also relies upon and uses the administrative portal to get user information and to change different database information.

The administration portal will be made in HTML, CSS, and Javascript. This portal serves as a way for administrators to access the backend databases to update recipes, add new recipes, update food sites, and overall make changes to the databases. We will also keep track of user information in this backend with a database that the app will have access to.

Conclusion

The Fridge Filler project is aiming to provide an efficient and technical way for food banks and users to use food resources. We want to help solve the inefficiency and uncertainty that comes with food banks. Our project will create one application for customers and a web client for the administrator. The application should allow users to add what pantry they have and provide search and navigation functions. The administrator portal will give the administrator the right to control the whole database system. This document discusses the technological feasibility of the “Fridge Filler” and addresses the multiple technical challenges that we will face during production.

What we envision: by the end of the development process, we hope to have a cross-platform application, with a clean and fast user interface, that will facilitate finding local food banks and recipes that can be made from the food you own in combination with the food you can obtain from nearby food banks or restaurants to enable healthier eating using the AZ Health Zone recipe database.

We are pleased to present our research, which we believe will make a successful software development. Here’s the summary of our findings:

Technical Challenge and Solution Table	
Technical Challenge	Solutions
Make our application look good on multiple platforms.	Ionic: Ionic works perfectly between different platforms.
Choose a suitable platform for the database.	MySQL: Is the one we are going with as it is widely used and we have the most familiarity with it.
Build a web-based access portal for administrators.	HTML/CSS/JavaScript: we will combine them together, use JS to implement functions.
Create tools for administrators to control databases.	JavaScript: JavaScript allows us to send requests to the server and change data in the database.
Ensure database security.	Separate user database from the primary database and Heavily encrypt user information.
Implement basic analytics to track the usefulness of our application.	Google Firebase Analytics: It provides us a basic way to solve basic analytics problems.
Navigate the user to the nearest foodbank.	Google Maps: Google Maps API allows us to implement search and navigation.
UPC Lookup and Scanning	Google Barcodes API: Google Barcodes API allows fast and clean searching and scanning of UPC barcodes.

The table above shows a list of technical challenges that we hope to overcome. It also displays the solutions we find and they will help us in future development.

Fridge Filler is excited to work on this project and look forward to developing and incorporating our proposed solutions into this project. Even though it may take some time for us to implement our features, we are confident to make them work.

References

Bing Maps VS Waze - differences & reviews? (n.d.). Retrieved October 22, 2020, from https://www.saashub.com/compare-bing-maps-vs-waze

Blog Details. (n.d.). Retrieved October 22, 2020, from https://realclear.software/google-maps-vs-bing-maps/

Dove, J. (2020, September 23). Waze or Google Maps: Which Navigation App Is Best for You? Retrieved October 22, 2020, from https://www.digitaltrends.com/mobile/waze-vs-google-maps/

Ionic framework. (n.d.). Ionic Article: Ionic React vs React Native. Retrieved October 22, 2020, from https://ionicframework.com/resources/articles/ionic-react-vs-react-native

Ionic framework. (n.d.). Ionic Article: Ionic vs Flutter. Retrieved October 22, 2020, from https://ionicframework.com/resources/articles/ionic-vs-flutter-comparison-guide

Says:, M., & Says:, R. (2019, October 08). React Native vs. Ionic: Which one is right for you? Retrieved October 22, 2020, from https://blog.logrocket.com/react-native-vs-ionic/

MySQL Connector. (n.d.) Retrieved October 22, 2020, from https://dev.mysql.com/doc/dev/connector-nodejs/8.0/

Datadog vs Google Cloud Platform. (n.d.). Retrieved October 22, 2020, from https://www.softwareadvice.com/app-development/datadog-profile/vs/google-cloud-platform/

MongoDB: The Database for Modern Applications. (n.d.). Retrieved October 22, 2020, from https://docs.mongodb.com/

Deploy Cloud Applications with MySQL Database. (n.d.). Retrieved from https://www.oracle.com/mysql/

MySQL. (2020, October 06). Retrieved from https://en.wikipedia.org/wiki/MySQL

The World's Most Advanced Open Source Relational Database. (n.d.). Retrieved from https://www.postgresql.org/

7 Database Security Best Practices. (n.d.). Retrieved October 23, 2020, from https://www.esecurityplanet.com/network-security/6-database-security-best-practices.html

Pdfmake: Client/Server-side PDF printing in pure Javascript (n.d). Retrieved October 23, 2020, from http://pdfmake.org/#/

Anon. Barcode API Overview | Mobile Vision | Google Developers. Retrieved October 24, 2020 from https://developers.google.com/vision/android/barcodes-overview

L.LC Cloudmersive. Cloudmersive. Retrieved October 24, 2020 from https://api.cloudmersive.com/