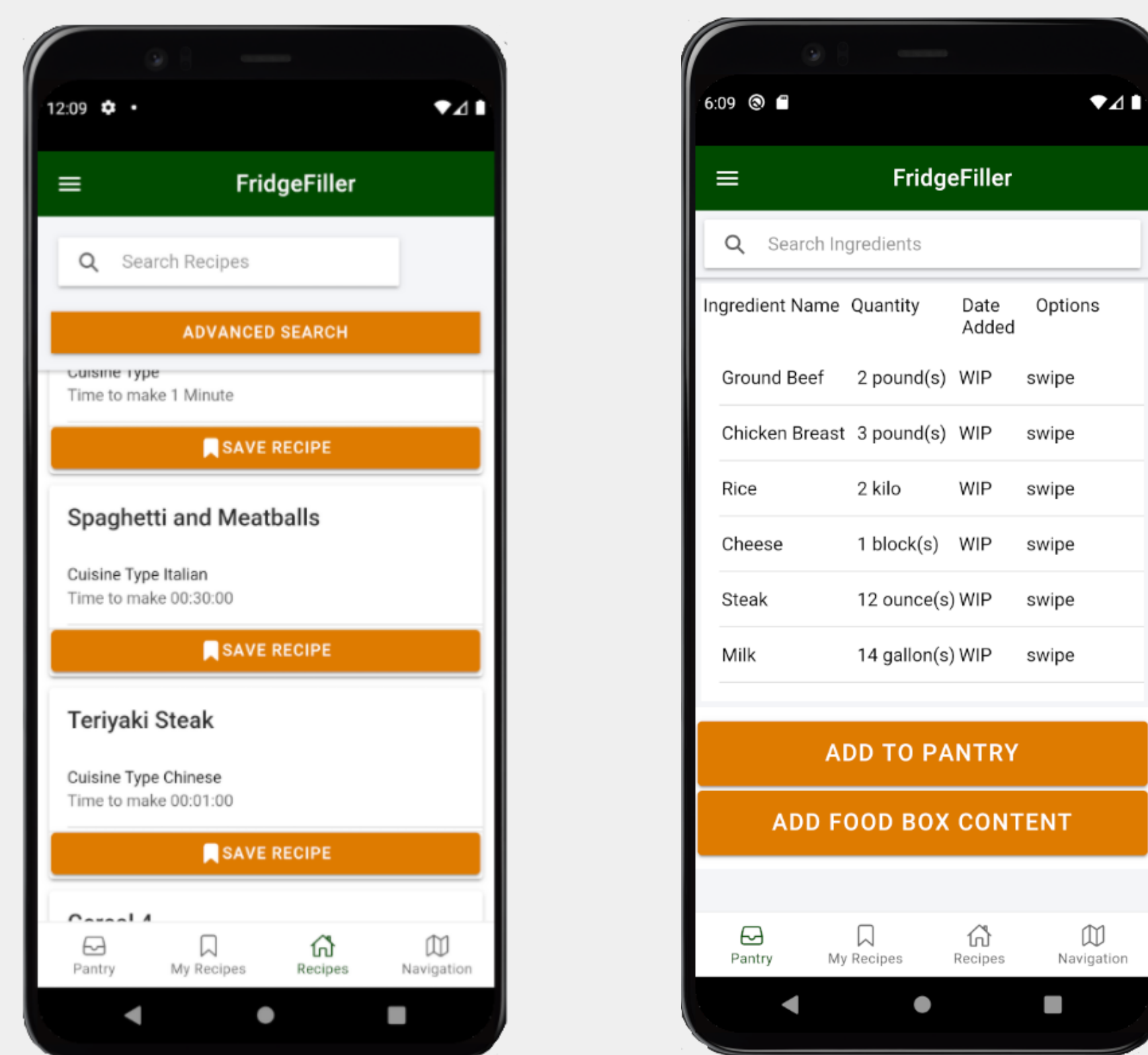# FridgeFiller: A Recipe App for Arizona Food Banks

Jonathan Derr, Travis Flake, Shangyi Dai, Gage Gabaldon, Zhibang Qin,
Sponsors: Dr. Richard Rushforth, Sean Ryan, Ceara Chirovsky, Saint Mary's Food Bank Alliance
School of Informatics, Computing, and Cyber Systems, Northern Arizona University

NORTHERN ARIZONA UNIVERSITY

College of Engineering, Informatics, and Applied Sciences

St. Mary's Food Bank®

Motivation

Food Banks in Arizona are facing a unique issue, which is leading to unnecessary food waste. This problem is largely due to their business model, which is designed around trying to get the products they work with from producer to consumer as fast as possible, which is necessitated by fact that much of the food they work with is perishable. This, combined with the fact that food banks don't often know what they will get, or when, means that employees and volunteers don't always have the time to prepare any sort of resources to go with them, which often leads to food getting thrown out unnecessarily.

This wastage is often a major source of frustration for not only the people that are meant to benefit from their service, but the food banks and their employees as well. Of course, there are services and websites that offer recipes that these people might be able to use to learn how to use any unusual items they get. However, many of the recipes offered by these services may require yet more unusual/uncommon ingredients, require a high level of cooking skill, or simply take too much time to make. Additionally, many of the people that would benefit from this live in areas with little to no internet connectivity, meaning that they wouldn't be able to access these services regularly.

So, another solution is needed. This solution would ideally take the form of one of those recipe services, but designed to be usable in areas with limited internet connectivity, due to the unique requirements of many that use Arizona's Food Banks. Additionally, it would also be nice if this solution would allow people to keep track of what ingredients they have, allow them to easily print/store recipes for later, and allow them to find other food banks or grocery stores that are part of the SNAP Program.

Fortunately, that is exactly what we have created.

Solution Overview/Key Features

Our solution to this problem is described as "a smart recipe app for food banks" by our clients. It takes the form of an Android application that provides recipes to users based on a set of criteria, as well as a number of supporting features and systems that allow it to function. Specifically, our project provides users with:

- Recipes and all associated data (steps, ingredients, cook time, etc.)
- A Digital Pantry for tracking ingredients
- Ability to Save recipes to the user's device
- Ability to Export recipes as PDF files for printing
- Links to resources maintained by the Arizona Food Bank Association

Additionally, our project encompassed all necessary support systems to allow this application to work they way our clients wanted it to, including:

- An externally accessible Database that holds all the app's information
- A web-hosted Interface for that database
- A web-hosted Authentication service
- An administrative service that allows System Admins to modify the data used by the app (except for any user-specific data).

Architecture

The specifications outlined by our clients heavily influenced our design. Chief among these was that the app should communicate with an external database. This meant we had to design this system as a web-based application divided into two parts: a Front End and a Back End.

The Front End of our project is the user-facing portion, where all the interactions between our service and its users take place. It consists of the Mobile Application, which is the main component of the Front End, and an Administrative Portal, where System Administrators can modify certain pieces of data that is used by the app.
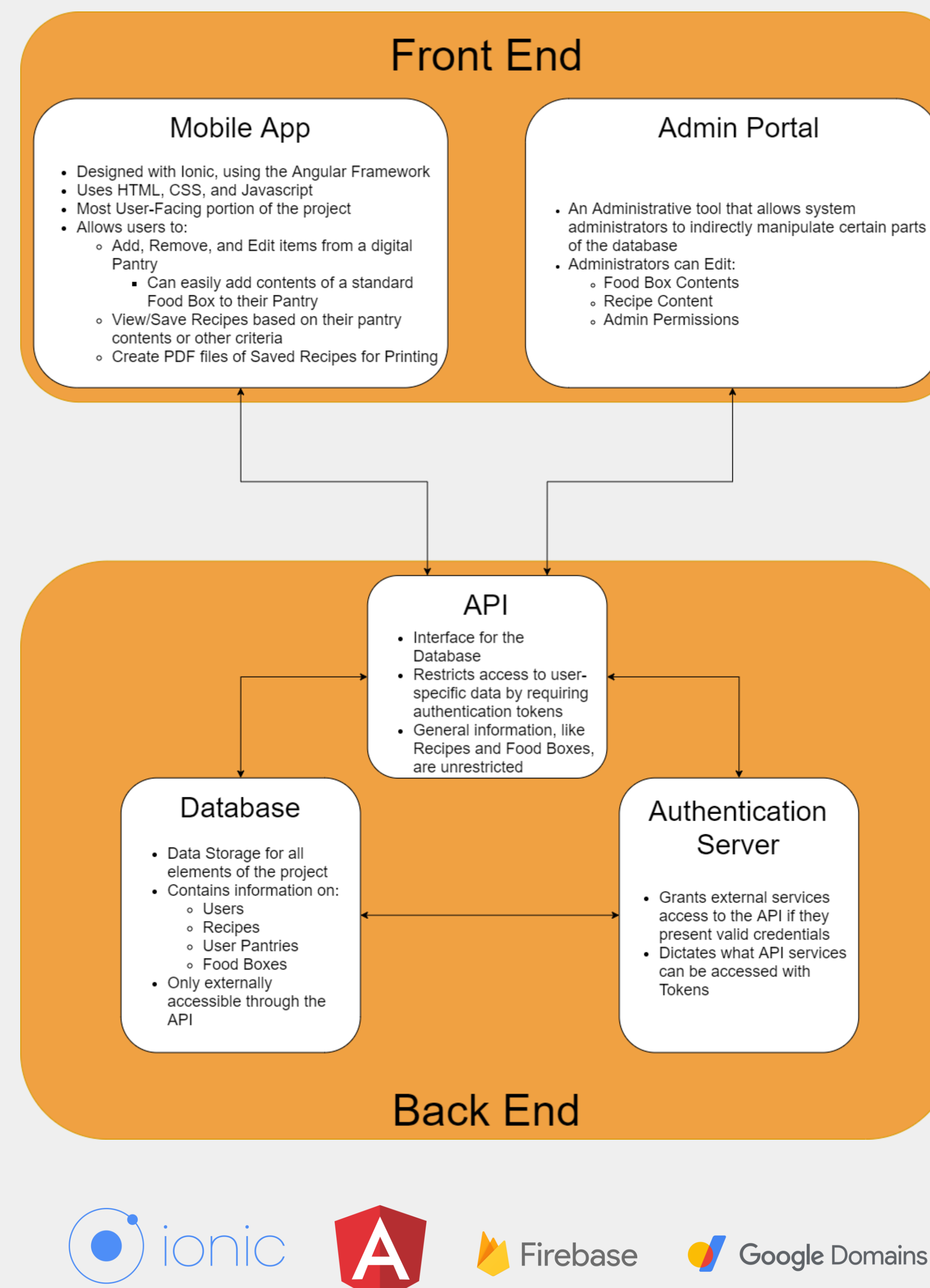
The Back End of our project is where the vast majority of the actual logic is, and is what allows the app to function the way our clients wanted it to. It comprises the database, the interface to access that database, and an authentication service to ensure data security.

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:
  - Add, Remove, and Edit items from a digital Pantry
    - Can easily add contents of a standard Food Box to their Pantry
  - View/Save Recipes based on their pantry contents or other criteria
  - Create PDF files of Saved Recipes for Printing

Admin Portal
- An Administrative tool that allows system administrators to indirectly manipulate certain parts of the database
- Administrators can Edit:
  - Food Box Contents
  - Recipe Content
  - Admin Permissions

Back End

API
- Interface for the Database
- Restricts access to user-specific data by requiring authentication tokens
- General information, like Recipes and Food Boxes, are unrestricted

Database
- Data Storage for all elements of the project
- Contains information on:
  - Users
  - Recipes
  - User Pantries
  - Food Boxes
- 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

Technologies

We had to use a number of different technologies to achieve the functionality our clients wanted. These included development frameworks that are designed with this kind of app in mind, data handling systems, a PDF generation library, and more. What follows is a list of everything we used during the development of our app:

- **Ionic**, a Mobile App design framework that allows for easy cross-platform app design
- **Angular**, a TypeScript platform that automatically generates code for use on multiple platforms
- **MySQL**, a Relational Database Management System that allows us to store and retrieve data in a very efficient way
- **Node.js**, a JavaScript Runtime Environment that allows JavaScript code to be run outside of a web browser, which is commonly used for writing server-side code
- **Express.js**, a JavaScript framework that extends JavaScript's base functionality and is designed for use in web and server applications
- **HTML** and **CSS**, which are the basic building blocks for any web site
- Raw **JavaScript**, for basic web programming
- **pdfMake**, a PDF generation library for JavaScript
- **OAuth 2.0**, a widely-used User Authentication standard defined by the Internet Engineering Task Force
- **Google Firebase**, a hosting platform for mobile and web applications that provides a wealth of functionality
- **Google Domains,** A Google Service that automates the setup and hosting of custom domains

Challenges

Unsurprisingly, developing an app of this scope and with this level of functionality was far from an easy task, both from a technical and organizational standpoint.

This was the first major project any of us had ever worked on, so we needed to figure out the best way for us to contribute to this project, how to divide up responsibilities, how to organize our workflow, and how to keep track of the various tasks we needed to accomplish. We elected to solve this by assigning specific roles within the team, with some level of overlap to avoid piling too much responsibility on a single person. These roles were divided amongst two teams: Front End and Back End. Each team had a "sub-lead" that answered to the overall team leader, while still maintaining some agency over what needed to be done within the team. We also made use of the Organization service Monday.com to help organize our tasks and define responsibility.

Technically speaking, this was a very challenging project to design and implement. This isn't entirely surprising, given the relative complexity of the system we had to design. Most of us had either never used the tools/languages we needed to use for this project, so we essentially had to learn as we went, while trying to make our code as clean, efficient, and close to defined standards as possible at the same time. We needed to learn Internet Communication protocols, how to set up the necessary tools to allow encrypted communication between two devices over the internet (such as SSL certificates), and standards for authenticating user information, just to name a few.

Future Development

Unfortunately, this project will not be going through the Capstone development cycle again. However, this is because our sponsors intend to pass off our work to their own development team to refine our project and prepare it for public release.

This team will be led by Anderson Moyer, who works for St. Mary's, and will expand to include several features we weren't able to include in this initial version, including social media login through Google, advanced analytics, an iOS version of the app, and more.

Although we won't be directly involved in its future development, our team is extremely excited to see what happens with our project in the future, and we are confident we've given them a solid foundation to build on.

Questions? Comments? Visit our Website by scanning the QR Code!

ionic • Angular • Firebase • Google Domains • node.js • ExpressJS • MySQL

Fridge Filler Admin Portal

Recipe Management

Recipe Name	Cuisine Type	Time To Make	Recipe Rating	option
Pork Carnitas	Mexican	00:01:00	8.0	
Roast Chicken	American	02:20:00	8.0	
Orange Chicken	Chinese	00:01:00	8.0	
Beef Brisket	American	03:00:00	8.0	
Pub style pork scratchings	British	04:30:00	8.0	
Egg Sandwich	American	00:02:00	8.0	
Teriyaki Steak	Chinese	00:01:00	7	
Spaghetti and Meatballs	Italian	00:30:00	4.1	