Project Name: Preserve - Reduce Waste at Home
Document Version: 2.5 **Document Date:** 04/11/24

Executive Summary

Food waste at home is a major problem that continues to occur globally and most people do not realize the substantial waste that one household can produce. At the micro level, we believe that having a tool to keep track of ingredients within a refrigerator and pantry and sending out necessary notifications of when an item is about to go bad can serve as an actionable solution towards solving this problem.

The project, Preserve, is an iOS mobile application that aims to bring this tool to a user's phone. The user can import ingredients and food from the physical world to the digital world through barcode scanning, receipt scanning, or manual entry. Each item will also be assigned an estimated shelf life based on a dataset from the US Department of Agriculture (USDA). Users would get notifications on their phone for when an item is about to reach its shelf life. Additional features include recipe recommendations based on the current ingredients.

Problem Statement

The Food and Drug Administration (FDA) indicates that the typical household wastes around 32% of purchased foods while it is estimated that 35 million people across the US suffer from food insecurity. As nearly a third of produce goes to waste, this issue poses economic, environmental, and social problems for the general public. Based on the USDA's Economic Research service on food loss on consumer and retail levels corresponds to around $161 billion worth of food in 2010 while this number is around $1,600 for a family of four. Outside of economic loss, resources such as water, labor, and energy used on wasted food also end up squandered, when could have been employed for other purposes.

The source of this problem in households can stem from many reasons including not knowing how to read expiration dates (such as the difference between '*best by*', *'expires on'*, or *'use by'*) resulting in throwing good produce away for precautionary reasons or as simple as forgetting about the product in the fridge until they go bad.

Our project aims to prohibit this large scale issue by targeting small scale users, in this case household inventories. By sending users reminders on produce that are about to expire or have expired, we aim to lower food waste by a considerable amount. To make utilizing these ingredients easier, our system will recommend techniques or dishes that use produce that are soon to go bad in the users inventory.

While some have found creative ways of lowering food waste such as dumpster diving (the act of rummaging for edible produce in trash), we aim to provide more intuitive ways of preventing the loss and waste of food.

Client Requirements

Preserve aims to be a standard iOS application that anyone who owns an iPhone or device that runs iOS can download from the app store. Creating a user account with an email address and password from the user is required as authentication will be needed for accessing and storing fridge data. We will be using Google's Firebase to process the backend authentication and database operations. Firebase is a set of backend cloud computing services and is integratable with iOS applications.

Although manual entry of food into the database is available, the app's convenience shines in the barcode scanning and receipt scanning import functionalities. Therefore, the user must be willing to give camera access in order to take advantage of these features. Furthermore, in order to maximize the app's utility, the user must commit to engaging with the status of items (used/unused) and assigning shelf life values for items that do not have a match in the shelf life dataset.

Technical Requirements

Data

The application needs a food shelf life database in order to assign each product its predicted expiration date. We are currently making use of USDA's CSV file that categorizes produce to categories with their respective shelf life.
One important aspect of the application is the user allowing the application to use some of their data. In order to make the user aware that certain data will be collected from them, the application will ask for permission before carrying out certain tasks, like enabling the phone's camera.
Another significant feature of the application is to be able to send push notifications to the user, reminding them about the products that will be going bad a week before, three days before, and the day of the expiration date.

Software

The user would have to have a stable Wi-Fi connection in order to connect to a virtual server. A server is needed to run multiple aspects of the application. With the virtual server, the application will be able run a variety of different machine learning (ML) models that are essential to the application. For instance, in order to categorize produce, semantic language models must run in the background as the application is processing.
In order to store the collected data, the application will utilize the Google Firebase database. Valuable information about the user, what produce they own, the quantity, and shelf life data will be stored in Firebase.

System Design

Application Overview

The application will be developed using Swift and the Xcode IDE. Swift and the SwiftUI framework allows developers to conveniently develop and organize pages or “Views” of the application into individual Swift files. Thus, we aim to maintain high cohesion and low coupling by approaching each page of the application as an individual Swift file and importing the necessary resources from other folders. The main “View” would eventually have switching mechanisms for displaying the desired page based on where the user navigates in the app.

Import Methods

A general theme and challenge of the importing methods is that getting the shelf life of a product given the product’s name requires a language model that would perform semantic search with the product’s name and the thousands of keywords in the shelf life excel dataset in order to find a match for the best estimates shelf life. The goal is to develop an abstraction of this process by hosting a backend server API which would run Python language models. The app can then communicate with this API by posting the product name and receiving the shelf life in return. This API is developed using the Flask Python library and utilizes a semantic search model found on HuggingFace. By taking in a food name as the query, the model performs the search on all the keywords given for each food item in the USDA excel sheet and returns the closest match and its corresponding attached shelf life. This process occurs after the name of the product or food is obtained through the methods mentioned below:

Barcode Scanning

In the barcode scanning import method, the application utilizes a third-party Swift package dependency called CodeScanner. This package allows developers to easily embed a scanning screen capable of detecting multiple types of barcodes. Once the barcode string is obtained, the application needs a way to get the product name with the barcode. This is achieved through the OpenFoodFacts API which provides a plethora of data from product name to nutritional values given a food’s barcode.

The API can be called with the url *https://world.openfoodfacts.net/api/v2/product/{barcode}* where *{barcode}* is the string obtained from CodeScanner. The API would then return a JSON response containing various information on the product. We are specifically interested in the “productName” and the JSON response is then parsed into a Swift struct object in order to access the name and be passed onto the shelf life processing mechanism.

Receipt Scanning

For receipt scanning, the application utilizes basic Optical Character Recognition (OCR) methods in order to convert analog to digital text. This is implemented into the application with Swift's Vision and VisionKit frameworks. After OCR is completed, a Python script scans through the output to parse through unnecessary data such as price or store information. Once the data is parsed, an NGram Natural Language Processing algorithm goes through each line to find abbreviations and converts them to their full form.

As this preprocessing of the receipt is done, the Spoonacular API is called to detect the food items in the data. The API can be called with the url: *https://spoonacular.com/food-api/docs#Detect-Food-in-Text.* The process is completed by assigning the food items' respective shelf life and adding them to the user's fridge database.

Recipe Generation

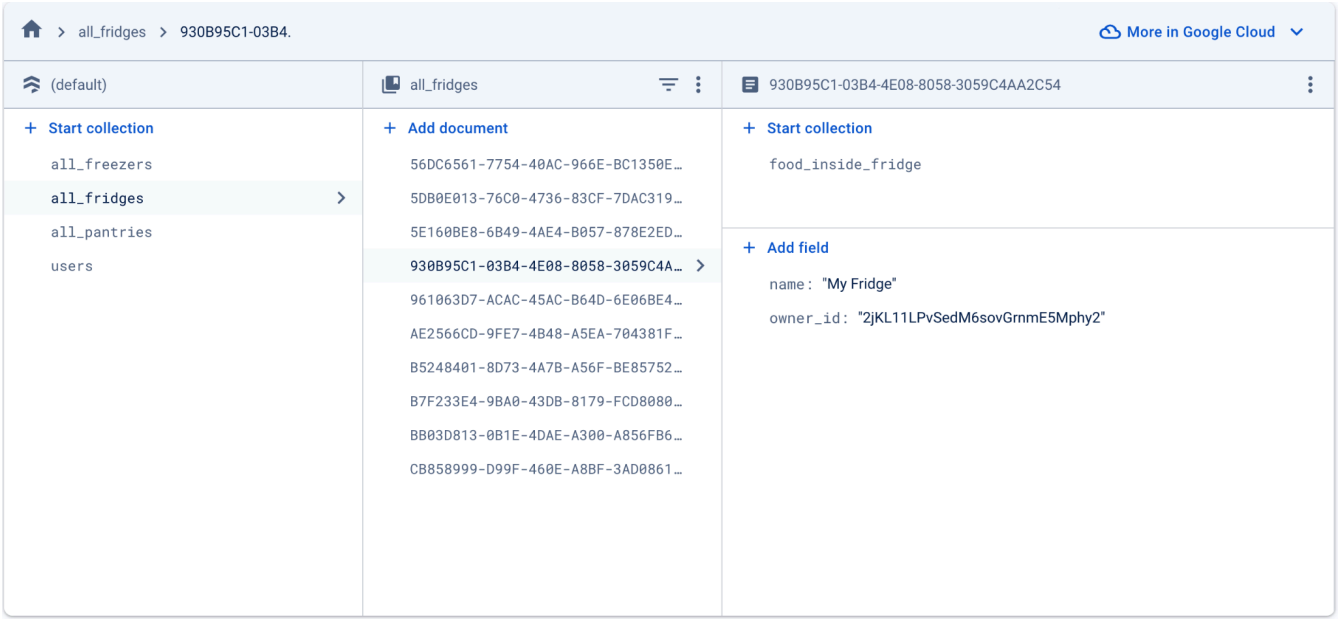
For image recognition, we use the same API model for food detection, Spoonacular. To be able to generate recipes with the existing product in the inventories, the application makes two calls to the API. The first call is to generate recipes with the given ingredients, where the call is made through the link: *https://spoonacular.com/food-api/docs#Search-Recipes-by-Ingredients*. The second call is to get the descriptions of the recipes, which is called through the link: *https://spoonacular.com/food-api/docs#Get-Recipe-Information*.

Authentication, Database, and Backend Processing

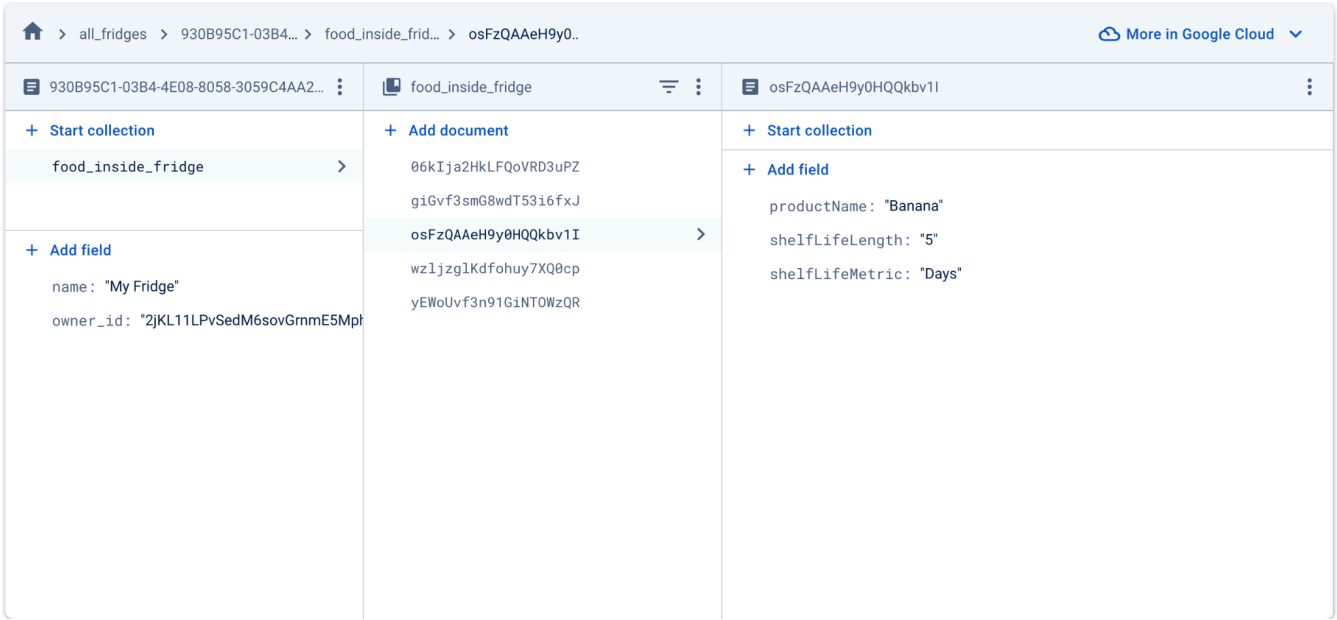
The application's backend processes can be split into two major components: authentication and database management. These functionalities are powered by Google's Firebase backend cloud computing services. Firebase can be integrated with the iOS platform and the authentication service provides an end-to-end identity solution and supports user account creation using email and password accounts.

The application will be utilizing Firebase's Cloud Firestore database solution to store inventory (fridge, pantry, freezer) and user data. As shown below, Firestore is non-relational and can be organized into *collections* containing items called *documents*. The sample schema is from Firebase's web admin console and displays a collection of all created inventories, with each individual inventory within the collection being assigned a document id. Each individual inventory shown contains

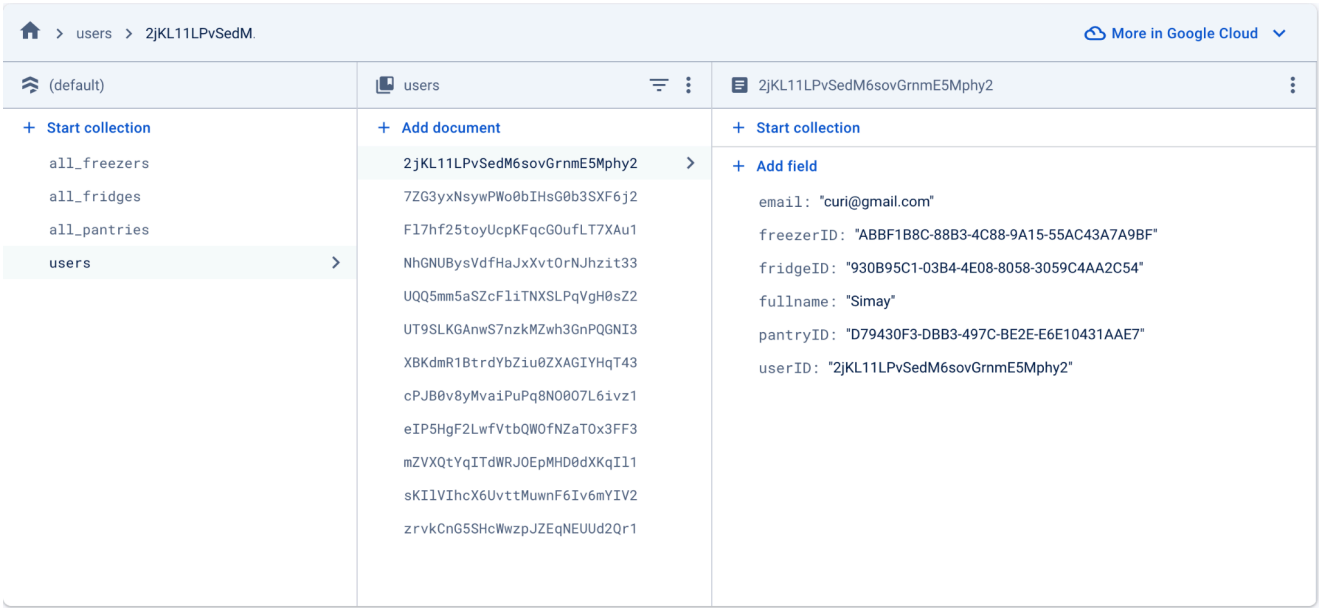
the properties *name* and *owner_id* which refers to the id of the user assigned in the authentication service.



Each inventory document will also contain a *subcollection* called *food_inside_[inventory]*. By expanding into this collection, data of the specific ingredients and their corresponding properties can be accessed as shown below.



In terms of authentication, Firestore also contains a *users* collection. As shown below, each document within this collection represents a user account and the associated data with that user:



By utilizing Firebase, no server-side code will need to be implemented. Due to the limited timeframe of this project, Firebase serves as a great backend solution for efficiency as more time can be spent on integration and application development

Installation Procedures

This application is readily available on Github and can be easily installed by anyone who owns a device that runs iOS. Given the limitations of the server provided by Colorado College ITS, features that utilize our language models can only be accessed on CC internet.

Operation Procedures

Day-to-day Operations

Once the application is installed into the user's device, they would have to have a stable Wi-Fi connection and update the application when needed in order to keep the system running. Otherwise, the usage of the application will be very straightforward as the application will deal with data storage, API calls, etc. on the backend. As the system keeps running the application will import user's data into Firebase.

Common Problem Remediation

Firebase provides an accessible and user-friendly admin console which can be accessed through the web in the case that problems arise in authentication and data. We can easily log into the admin console to do necessary user account management and database processes without invading privacy.

User Procedures

A typical user will ideally start using Preserve after a grocery trip. They can import their items either through barcode scanning, receipt scanning, or manual entry. Once they have stored all their items into the app, they are done with the import process. Notifications can be used as a way to quickly process and allow the user to update the status of an item. As time goes on, Preserve will send a notification for every item that is nearing its assigned shelf life. The user will also have the option to generate recipe recommendations based on ingredients that are about to reach their shelf life.