University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2020

Food Alert

Gabriel Priest, Nick Waterworth, Jasper Harrison, Bentley Lager, Alycia Carey, Jiamin Lin

Abstract

Food waste is a prominent problem in many countries, including the United States. Americans regularly throw away huge portions of their groceries. They forget a tomato at the back of the refrigerator, or potatoes in the cabinet and find them rotten a few weeks later. Not only is this a financial drain on Americans, but due to explosive population growth, there will be a global food shortage within a decade [4]. Wasted food production in the US needs to be freed up for export to other countries, whose agricultural production falls behind. If the problem of food waste is not addressed in a few years there will be food shortages in many growing countries. The objective of the project was to create an easy to use application that would help consumers track and plan their food purchases, to reduce food waste. To accomplish this, we employ both React Native and Firebase technologies. We used a React Native framework for page creation and navigation, while Firebase was used to hold user and food data. The application allows users to scan a receipt, then the app calculates the expiration dates of the items and uploads items to Firebase. The user can track these expiration dates on a calendar and manually add or delete items.

1.0 Problem

Each year, Americans waste over 133 billion pounds of food, which equates to 218.9 pounds for each person in America. This results in over 161 billion dollars in food wasted every year. This money could be used for other essential purchases in homes where people live paycheck to paycheck. The USDA has set a goal to cut food waste by half before 2030; FoodAlert could assist in addressing this issue [3]. If we continue to waste food at the level we have been for the past several years, the world could be irreversibly harmed [1]. Food waste is an exponential loss of resources, not only in America, but globally. According to FOA (Food and Agriculture Organization of the United Nations) [5], "Global quantitative food losses and waste per year are roughly 30% of cereals, 40-50% for root crops, fruits, and vegetables, 20% for oilseeds, meat, and dairy plus 35% for fish." Food waste has become part of everyday life in countries like America, however, actions taken by such a large population have an affect on the rest of the world.

The world's population is exploding, and the question that needs to be answered is: "How do we feed 9 billion people by 2050?". This question evolved after food prices hit an all time high in 2008 [11]. In the last 40 years China and Africa have gone from net calorie exporters to

net calorie importers. That is, they produce fewer calories than are consumed. This is not a statement on malnourishment, simply demand has exceeded agriculture production, which is driven both by population growth and economic growth in sectors other than agriculture. Gro-Intelligence projects that by the year 2027 the world will be at a 214 trillion calorie shortage, and the world is not equipped to make up this deficit [4]. This looming global food crisis cannot be solved by governments alone, it will take change on the consumer level. Consumers must recognize the importance of being responsible stewards of the Earth and its resources, or the population at large must face the consequences.

2.0 Objective

The objective of this project was to create an easy to use cross platform application, which would help users keep track of food and reduce their food waste. The application would integrate React-Native and Firebase technology to track this information and enable the user to improve their food purchasing habits. The camera would allow the user to scan a receipt and the application would parse the information. The parsed information would be used to cross reference an expiration date database, then be pushed into the User database, updating the calendar with the item's expiration date. The application should also have an easy to use list view page where the user can easily select and remove food items from their inventory. The purpose is to make the application as easy to use as possible, so users can integrate this into their normal food purchasing routine.

3.0 Background

3.1 Key Concepts

React Native [6]: React Native is a platform used to create native apps using React for Android and iOS. React is a javascript library, which is used as a tool to build UI Components. The basis of React is the UI components that are then rendered within the HTML DOM. This Framework is supported by Facebook and is continuously under development to streamline the development process. Animations and multiple fragments are included in this framework to make the production of applications extremely simple. Fragments allow you to group a list of children elements without adding extra nodes to the DOM. To create application elements, you must declare each one, and render the element with a render() method. Each rendered element is immutable and must be rerendered each time it is changed. Due to how components are rendered, the application can be used on any platform seamlessly. The development cycle of ReactNative applications also allows for an almost instant view of changes and allows for simple debugging as problems arise. The difference between React and ReactNative is the type of components that are used within the application. React uses web components and ReactNative using native components. There are plenty of benefits to choosing ReactNative vs native development for Android and iOS. React is known for optimal performance because it is connected to native components for both operating systems. The framework also has support for a wide range of third party plugins due to the lack of some components in the main framework. For example, if the application needs to have any Map, ReactNative allows the developer to connect to a plugin with a native or third-party module.

Firebase [2,10]: Firebase is Google's mobile application development platform that houses a variety of analytic, authentication, database, configuration, file storage, and push messaging tools. All of the services are hosted in the cloud and scale easily without much hassle from the developer. Client SDKs provided by Firebase allow developers to interact directly with the backend Google services with no need for middleware between the app and service. Firebase focuses on three main mission points: “Build Your App”, “Improve App Quality”, and “Grow Your Business”, that coupled together, allow for the creation of a steady and powerful application. For FoodAlert, a combination of all the toolkits above were utilized. Once the main framework of the application was created and had secure user login and data storage, focus shifted to using the ML kit to parse text from the receipts and Analytics and Predictions to enable the user to better manage their food wastage.

Text Image Recognition and parsing AI[8]: Using Firebases ML-kit we can create local neural network models within our react code to handle text recognition through and image. First, we get a wrapper so that we can use their ML-kit library within the react code. From there their library allows us to build models locally on our app, such that the user can scan images with their camera and get instant text feedback for their purchases. By using the Firebases library, it prevents us from having to implement our own model, train it and pass it onto a mobile model or the cloud. This reduces the burden of serving a neural net, and needing a connection to get results for queries. React only comes in the form of an Android or iOS SDK and not an official web-SDK, but there are open source wrappers and libraries that are compatible with react. ML-kit also allows for the use of custom TensorFlow-Lite models that lets users easily port them onto the mobile platform, meaning if normal text recognition doesn’t cut it a custom model can be made.

3.2 Related Work

There are many applications out there that are related to food, and it is either a mobile application or a website. Such as FoodKeeper APP [9]. This app was developed by the USDA’s Food Safety and Inspection Service, with Cornell University and the Food Marketing Institute. The main function of FoodKeeper is to provide the user with information on how to store the food that will have a maximum expiration date, and maximize freshness and quality. This app is available in three platforms: iOS, Android and Web application.

No Waste [3] is an app that has similar functionality, developed by KH Creations IVS. The app includes food inventories, barcode scanning, receipt scanning, and recipe suggestions. This app does have many aspects that work well. However, during the testing of this app, we discovered that the receipt scanner has a bug where it will enter a deadlock and scan indefinitely. When the app enters this state there is no way to break the deadlock and the app must be exited entirely. Once a receipt is scanned all expiration must be entered manually. Our app will calculate the expiration dates and auto-input the information. This app is also exclusively available on Apple platforms and our app will be built to work across platforms. This app was originally developed in Sweden and there are some parts of the app where some of the writing is still in Swedish and some of the monetary notations are still in the Swedish Krona.

There is another mobile application called Fooducate [7]. This mobile app was developed by Fooducate, Ltd. It is available on iOS and Android. The main function of this application is

when the user scans a food item, the system will assign a grade for a food item based on its nutritional value and with a breakdown of what is actually in the food. It will also give the user alternative options and it tracks daily calories on the side.

However, the FoodKeeper tells the user how long will the food stay fresh, and the Fooducate tells the user nutritional values. None of them will keep track of the expiration date of the food that the user purchases. Our FoodAlert app will allow the user to take a picture of the food items and keep track of food expiration dates. It will also notify the user the day before a food item expires, the notification day would be calculated and scheduled. In the end, FoodAlert will provide a solution that would help the user keep track of their food and food expiration.

4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

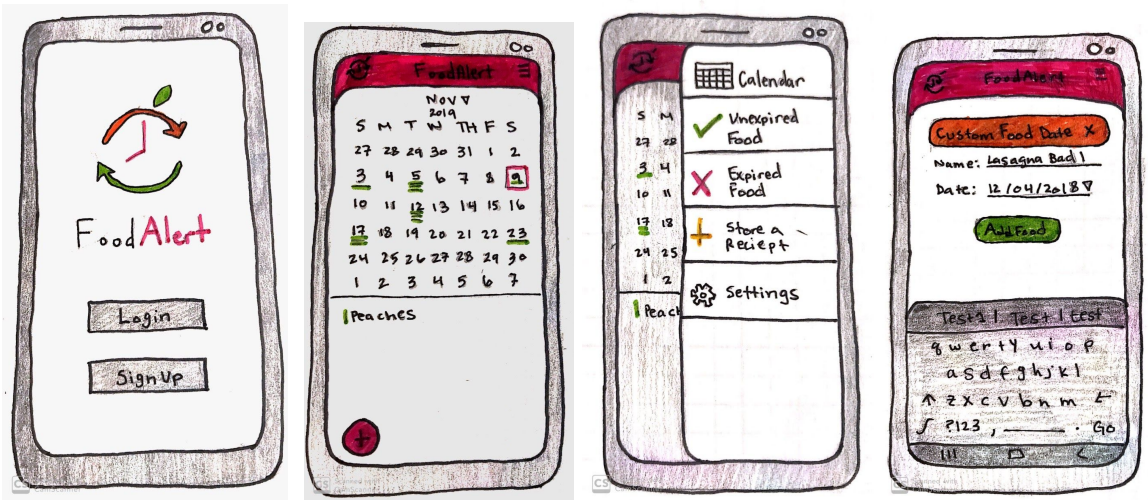
Functional Requirements

- Scan receipts and store parsed food information into Firebase.
- Provide the user with the ability to manually enter an item and date into the calendar.
- Allow the user to choose a product and manually remove the item from the food item list.
- Calculate the expiration date of items from Firebase and output to the calendar and item list.

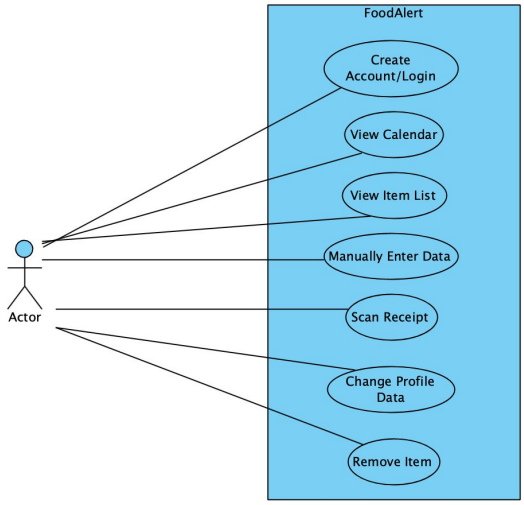
Nonfunctional Requirements

- A user interface should be easy to navigate and conform to android and iOS norms.
- Startup should complete within 5 seconds.
- Receipt scanning and parsing should complete within 15 seconds
- The calendar should display the expiration dates of food.
- The items list should display both the purchase date and expiration date.

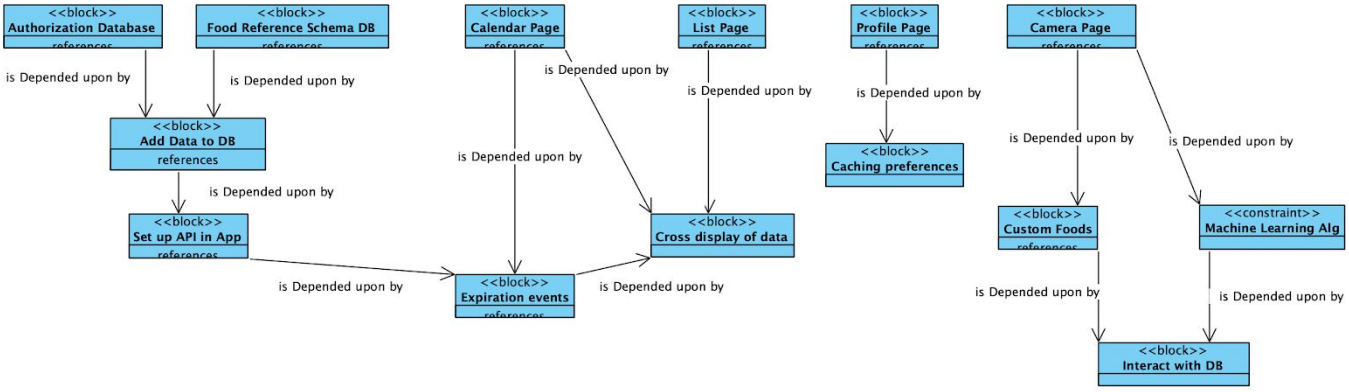
Original UI Designs



Use Cases



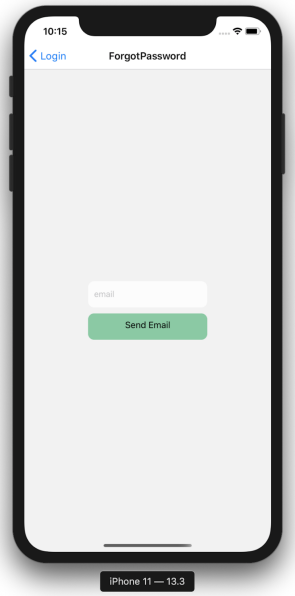
Dependency Map

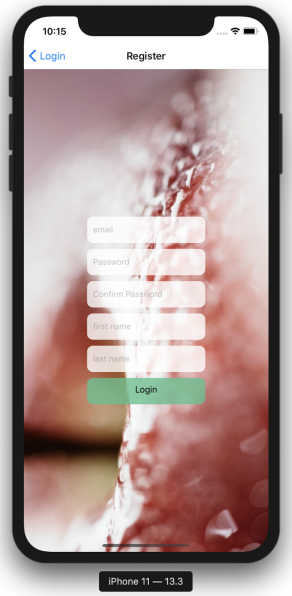


4.2 Detailed Architecture

Our architecture consists of many interconnect pages that make up a react-native application. Using the navigation framework provided within react native, users can navigate between pages with our defined behaviors. Each page has unique interactions with one another with information being passed between them. Most pages have a navigation bar at the bottom, that is used to traverse between the given pages. This bar is implemented as a global layout within the react native application, and can be applied to pages it should be drawn on. React native works by having a navigation stack that creates accessible pages for the page you are on, and pages you visit. The framework appends the given state to the stack, when you leave the page, it is removed from the stack. Each main page was first implemented as a skeleton, for navigation testing that interconnected each defined page. Listed below are all the pages and their functionalities.

- **Login Page**: The login page allows the user to login or register, using Firebase as the backend. This page leverages Firebase's authentication package. The application initially checks the user's Firebase token to see if it has expired or not. If the token is still valid, the user will be directed to the profile page, else the user will remain on the login page where they can input their credentials or create a new account. The user is notified via alerts if the credentials they put are not associated with an account, or if the password is invalid. If the user were to forget their password, there is a "Forgot Password" button on the login screen that will have Firebase send a password reset email to the email the user inputs in the text field. Once the user is logged in they are directed to the Profile Page where they can update their account credentials and view/modify their profile image. One of the main issues with using Firebase with React Native, is having to install native dependencies for iOS and Android. This was done by going into their build scripts, *build.gradle* for Android and *podfile* for iOS. Each of these scripts needed to have all the firebase packages manually added to them for the app to function correctly. Firebase provides a JSON file that includes the api key, auth domain, database URL, project ID, and more in order for the application to send/receive data from Firebase correctly. This also needed to be natively added to each project for Android and iOS. Below are screenshots of the login and register views.

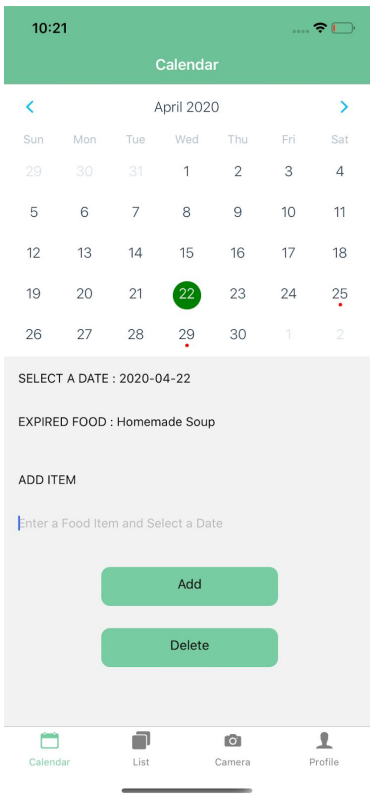


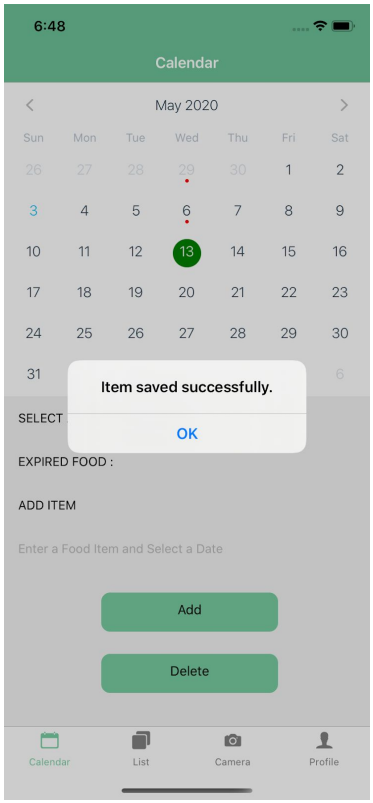


- **Calendar Page**: The calendar page shows all the expired items that are associated with their account during a particular month. It allows the user to manually add either custom made items and expired dates, such as homemade food, or any groceries. It also allows the user to delete all items on a day (where List Page allows the user to delete individual items). When the user clicks on a day, it will list all the expired items under the calendar. The red dot represents that some foods are expired. It allows the user to have a better idea

of when and what food items are expiring. The calendar portion of the page was built on top of the react-native-calendar module.

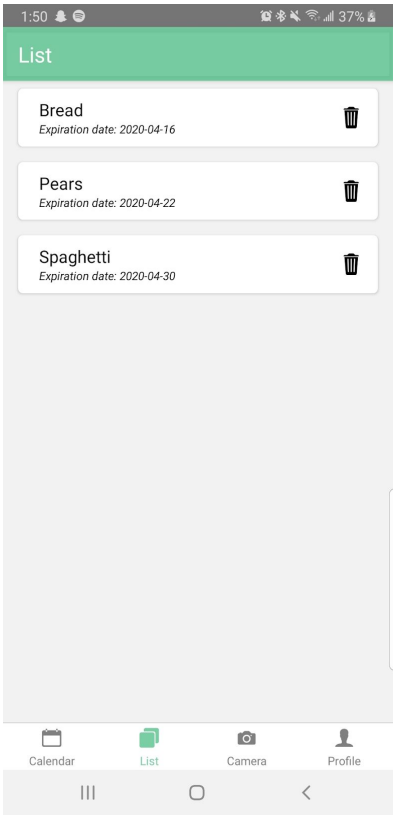
- Add button: If the user is going to add an item, the date must be selected and the ADD ITEM text field must be filled. If either one is missing, the application will alert the user to input missing data. The add function is accomplished by first making an API call to FoodData Central, using the item name that the user entered. The API then returns a JSON payload. The program uses the returned fdc_ids to make a call to Firebase firestore. It then compares the returned fdc_id from FDC with firestore, then uses the expiration days of the first matched fdc_id. Once the application retrieves the days it will perform a date calculation. After calculation is complete, the application uploads the date, item, and status of expired to the realtime database. If the program does not find a match (i.e. the items are custom made), it will skip the calculation and upload the selected date, item, and status of expired to the realtime database. If the function is finished without error, it will notify the user that the item has been saved successfully. Else, notify the user that it failed to save the item.
- Delete button: If the user is going to delete all items on a day, the date must be selected. If the date is not selected, the application will alert the user the date field is missing. The delete function is accomplished by first making a firebase call using selected day and pulling down all the items of that day. Then, it uses the Firebase built-in remove functionality to delete all the items. If deletion is finished without detecting an error, it will notify the user deletion is complete. Else, notify the user the deletion is failed.
- Future work: Set a checkbox to differentiate if the item is grocery or custom made. When the user adds a custom food item, the program will calculate the difference in days between current date and selected date, the difference will be the length of expiration days. Use Machine Learning algorithms to learn this item and expiration days. After enough trails, it will notify the user that this item is now in the database, and it will estimate the expired date for the future.





- **List Page**: The list page's main functionality is to provide a view of all the items a user has associated to their account and their corresponding expiration dates. This is accomplished by first making a call to Firebase and pulling down all of the items under the user's name. This call has an active listener attached to it, meaning that whenever another item is added to the users account, or equally if an item is deleted, then the list will update to show the most accurate view. After retrieving the items, they are displayed as a custom flatlist view that allows the user to easily see the item and expiration date. This is done by pushing the food item and its expiration date to a list, and then using the list to populate the custom flatlist. In addition, the user has the ability to delete the item from their account by clicking the trash icon on the right hand side. The most difficult part of generating the list view page, was making sure all the processes happened in the correct order. At first, the page was trying to display before the data was fully populated, causing it to throw errors. To fix the error the code had to be modified to be asynchronous. This would make the page wait to display until all of the data had been pulled back from Firebase.

  This view is rather simplistic, so in the future it would be beneficial to add a picture of each item to the list item. This can be accomplished by having an image database of common foods and searching the database with the food name and then pulling and displaying the correct picture. In addition, it may be helpful to change the font color to red if the food is past its expiration date, or include a toggle to only display expired or unexpired foods.



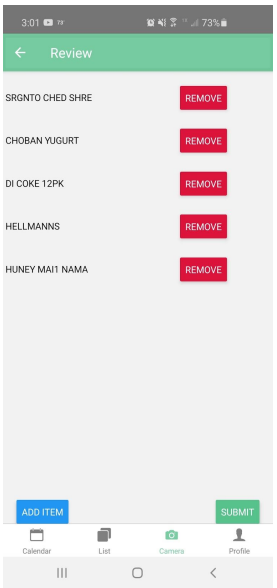
- **Camera Page:** The Camera page's main functionalities are the ability to scan receipts, and send the information in them to review. The primary objective of this page was to apply the use of text recognition software in order to scan the text of receipts. In order to utilize the users' camera, the module react-native camera was used. This library has a default implementation of text recognition using firebase mlkit which is already good. However, a period of time was spent trying to find an improvement model that could be used as a substitute to the default. Two models in particular were explored, the first being Aster. The model was experimented with, but we ran into the issue of different tensorflow models and dependencies from the past that were too far back to convert the model into a recent TFLite model. The other model that was attempted was the Decoupled-attention-network. This framework was implemented in PyTorch, which given a few libraries could have been transferred over to a TFLite model. However, the same issues from the previous attempt also persisted with this network along with the loss of information on the conversion of the model from the PyTorch framework to Tensorflow. With these developments, no other Text-Recognition models were searched for to perform better than the google model because of dependency and framework differences. Also no AI model will be delivered. The base Google model also performed exceptionally on text of strong form(ie not fading). Another attempt was made to try and combat the AI driven text detection by using Tesseract OCR, an optical character recognition engine through a react-native port. The difference between OCR and Neural Networks is OCR is a set of handcrafted methods and transformations used to extract the

text embeddings. Once Tesseract was set up app-side with the react-native camera, we conducted experiments on its performance, just on basic text with bad results. The engine outputted poor results due to the lack of any pre-processing of the image to make the text more contrasted from the image. But, due to the fact that the engine was extremely slow to process compared to the ML-kit method, it was decided to not be a viable path compared to the already implemented ML-kit version. This finding further reinforced the use of React-Native Cameras MLKit.

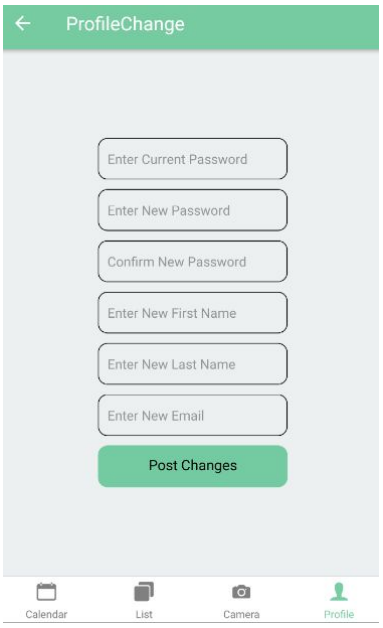
This model always runs device side on android but for IOS, ML-kit runs exclusively through firebase. No solid performance could be found for the given text recognition model on ML-kit, through experimentation on full form(not faded) text a high accuracy above 90 percent was inferred. The text data outputted follows the form of each line being its own object, all a part of a text block object. However, due to the variable nature of receipt text, methods were created to handle the mostly matching text in the Review-Page. Once the text data from a given receipt was scanned, the user is sent to the next page which is the ReviewPage. If there was not any text, the user gets an error and stays on the Camera page. The main takeaways from this approach involved time spent in trying to utilize other methods. When researching and trying to implement other models, it should have been apparent that the Google model was going to be the best model for most text-recognition tasks on mobile implementations. Even when taking receipts into account, the main issue with scanning them is faded text and lack of information which is not a Machine Learning issue but an image processing and data quality issue. The implementation of this page turned out to be standard with other applications and did not have any technical impact. Future work here would include taking the image that is being passed into the network and see if there is any pre-processing that could further increase receipt scanning accuracy.

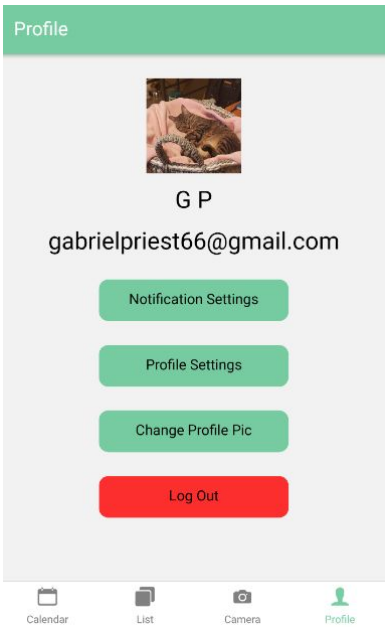
- <u>Review Page:</u> The Review page is where the user gets sent after a receipt/text is parsed for its text data. This page can only be accessed once the user scans a receipt that has text information to be parsed on it. A block of text output from the output from the Camera page is passed to this page and is parsed immediately. Due to the fact that the receipts that are being parsed could be faded and missing important markings for characters, methods like fuzzy search and regular expressions were employed. Fuzzy search works by using a matching system to score most similar strings to the current string you are trying to match. In order to implement the fuzzy search functionality, the Fuse.js library was imported. Fuzzy search was used to find the text closest to "SUB-TOTAL" on a receipt, since the text nationally is used at the bottom of the food item list on receipts. Other strings found on Market Street receipts like "You saved!" have entries to be removed in our string removal approach. We then use this index as a marker for where the receipt needs to be cut off from the bottom and remove unnecessary data from what the algorithm is parsing. From there regular expressions are used to take out specific items of only numbers and identify price tags of the form "#^*.##". This regex only finds floating point values, which are also used nationally to depict the value of a purchased item. Our implementation uses the price tags in the receipt to total how many items were purchased for the last step of the text parsing. Here,

the count of items using the prices added up is used to determine how many items are then kept in the remaining array of strings. Once the actual purchased items are removed, the items are passed into a loop that creates them all into food objects to be passed into the database and also shown on the Review screen. After the function is done parsing the receipt objects, each item is stored into the state of the page and displayed onto the screen. From there the user can edit, remove and add more items. The text recognition network and algorithm are not perfect since many different receipts from all providers have many formats. An example of this is Walmart having numerical codes in between food items and their price on the same line, which may or may not be matched together by the ML-Kit model. Thereby users need to be able to interact and correct data that our algorithm did not detect or remove. Below is an example of the Review screen. To create this display, Flat list entries were used that parsed our stored food item list. When the submit button is pressed, the firebase helper call is used to iterate through the list of food items and pass them to the firebase database. Our implementation uses a conjunction of neural network output, along with a custom receipt parsing algorithm, to create a novel approach for scanning receipts. Our implementation has many areas of improvement, including accuracy and scope. In respect to accuracy, the fuzzy search strings are not the best way to remove subject strings but works well on the Subtotal string. Regular expressions also struggle even on text that has some variability like what is found on walmart receipts. For scope, more dynamic storage of receipt templates could improve a case by case performance of how receipts are parsed based on where they are from. Using firebase, the information of vital strings to remove text blocks from the neural network output could have been utilized dynamically instead of using handcrafted cases. Also, the GUI could have been formatted better to make it a more user-friendly experience on the review page by having better segmented entries.



- **Profile Page:** The Profile page is where the user can see their current information and access settings screens. It allows users to change information, such as their password, name, and email address. The Profile page is the first page that is seen when the user logs into the application, and allows the user to check they have been logged in correctly. This is achieved by pulling data from the Users table in Firebase, which contains the user's unique identifier, email and name. The page displays the user's name and email address on the top of the screen and also displays a profile picture for extra personalization. The profile picture was the most complex part of the Profile page because it involved the use of two new frameworks, Firestore and Image Picker. The Image Picker opens the device's photo gallery, and allows the user to pick an image from their existing images. When one is picked, it returns the user to the Profile page, and uses Firestore to upload the selected image to a Firebase collection. At that point the profile page must be reloaded and the picture is pulled from the collection to be displayed. The biggest conflicts that came from creating this page, came when profile pictures were added. Originally, the user was going to take the picture in the app and that new picture would be uploaded straight to the database. However, this proved to cause some issues with the processing due to the pictures not being saved on the phone, so we instead looked at the Image Picker API which allowed the upload of the profile pictures to run much more smoothly.
    - <u>Profile Settings Button</u>: From the Profile page, the user can access the profile settings page, which contains text inputs which the user can use to change their entry in the User table. The profile settings page also allows the user to change their password, but they must reconfirm their current password for this functionality to be available to them.





4.3 Risks

Risk	Risk Reduction
React libraries come with many dependencies, which may cause problems.	Minimize the amount of unnecessary web-app dependencies that could have security faults in their open-source codes.
Lost database additions.	Firebase has an offline local database so users can store data locally and sync with our servers when connected again.
Incorrect parsing from image scanning.	Create front end functionality to allow a user to check items and adjust when necessary.

4.4 Tasks

1. Firebase set up
2. Set up page navigation
3. Set up user authentication using Firebase
4. Create User database
5. Create Expiration date database
6. Write a script to push data into Expiration Date database
7. Create Calendar Page
8. Create Registration Page
9. Create navigation bar
10. Create List Page
11. Create Profile Settings Page
12. Create Camera page
13. Implement text recognition with the camera
14. Implement auto item adding with the camera
15. Create a list view with all foods
16. Styling for profile page
17. Styling for login screen
18. Styling for sign-up page
19. Front end for the Calendar Page
20. Create API for Firebase
21. Implement custom calendar event creation
22. Create front end event creation page

23. Allow users to select and remove expired items

4.5 Schedule

Tasks	Dates
1. React-Native setup/research	12/30-1/10
2. Basic Skeleton App	1/10-1/16
3. Firebase Setup	1/16-1/21
4. Login/Registration/Authentication	1/16-3/9
5. Navigation Toolbar	1/21-1/23
7. List page	1/21-3/12
8. Profile Page	1/21-3/12
6. Calendar Page	2/03-4/30
9. Camera Page	2/03-2/18
10. NN Research	2/18-3/20
11. Review Page	4/05-4/30
12.Firebase helper	4/13-4/21
13.Styling	4/20-4/30
14. Documentation and functional Testing	4/25-4/30

4.6 Deliverables

- Design Diagrams and UI drawings
- Database scheme
- React Native code
- Final Report

5.0 Key Personnel

Alycia Carey - Carey is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed relevant courses such as Software Engineering, Database Management Systems, and Computer Networks. She has worked for the University of Arkansas Network Engineering Team as a Network Security Technician as well as participated in several cryptography research projects. She will be responsible for front-end development.

Jasper Harrison - Harrison is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses like Data Mining, Artificial Intelligence, Software Engineering, Mobile Programming,

and Database Management Systems. He has worked for a social media startup, a healthcare company, and for one of the big banks as a developer. He will be responsible for unit testing and integration testing as well as connecting the different tiers of the app together.

Bentley Lager - Lager is a senior Computer Science and Mathematics major. She has completed relevant courses including Database Management, Information Security, Software Engineering, Computer Networks, and Multimedia Compression and Delivery. She has worked for a social media start up, a healthcare software company, and a management and technology consulting firm. She will be responsible for the database development.

Jiamin Lin – Lin is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed relevant courses like Software Engineering, Wearable & Ubiquitous, and Database Management. She has worked as a software developer and IT support for private and public companies. She implemented the functionality in calendar page and helper functions, as well as limited database tasks.

Gabriel Priest - Priest is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses like Artificial Intelligence, Mobile Programming, and Data Mining. He has worked as a Java developer at Cerner. He will be responsible for the camera-based functionality of the application.

Nicholas Waterworth – Waterworth is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses like Artificial Intelligence, Software Engineering, and Algorithms. He has worked for a social media app startup and a Department of Defense contractor. He will create the AI network for text character identification in application to shopping receipts and contribute to UI design.

23.0 Facilities and Equipment

Android Phone: For testing and development of the FoodAlert app. Multiple members of the team have android phones with different operating system versions that can be used for testing purposes.

iPhones: For testing and development of the FoodAlert app. Multiple members of the team have iPhones with different operating system versions that can be used for testing purposes.

7.0 References

[1] Center for Food Safety and Applied Nutrition. “Food Waste and Loss.” *U.S. Food and Drug Administration*, FDA, www.fda.gov/food/consumers/food-waste-and-loss.

[2] “Firebase Products | Firebase.” *Google*, Google, firebase.google.com/products.

[3] “Food Waste FAQs.” *USDA*, www.usda.gov/foodwaste/faqs.

[4] “Global Agricultural Data & Analytics Software.” *Gro Intelligence*, gro-intelligence.com/.

[5] "Key Facts on Food Loss and Waste You Should Know!" *Food and Agriculture Organization of the United Nations*, www.fao.org/save-food/resources/keyfindings/en/?fbclid=IwAR1ZgQw9MCzbHgqRdt6OQSJvpAaCGBnsSWX3ruoArvcGoYf7bRpawiVjtV0.

[6] "Learn the Basics · React Native." *React Native Blog ATOM*, facebook.github.io/react-native/docs/tutorial.

[7] "Lose Weight & Improve Your Health with a Real Food Diet." *Fooducate*, www.fooducate.com/.

[8] "ML Kit for Firebase | Firebase." *Google*, Google, firebase.google.com/docs/ml-kit.

[9] Public Affairs. "FoodKeeper App." *FoodSafety.gov*, 12 June 2019, www.foodsafety.gov/keep-food-safe/foodkeeper-app.

[10] Stevenson, Doug. "What Is Firebase? The Complete Story, Abridged." *Medium*, Firebase Developers, 25 Oct. 2018, medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0.

[11]"The World Food Crisis." *The New York Times*, The New York Times, 10 Apr. 2008, www.nytimes.com/2008/04/10/opinion/10thu1.html.