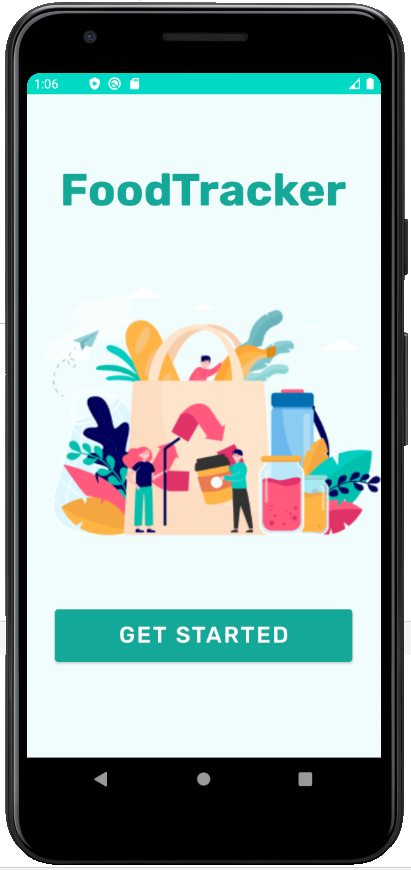
FoodTracker Mobile Application



Sara Eid Mohamed

Justin Dar

Moiz Lokhandwala

Josh Wong

**Table of Contents**

[**1. Description**](#_gcvtvpv2hezb) **3**

[**2. Operation Walkthrough**](#_r7vfue4tdjxj) **4**

[**3. System Component Diagram**](#_bmnamslml7kq) **7**

[**4. Component Descriptions**](#_jpkeomg9u6g2) **7**

[4.1 Cloud Firestore Backend:](#_nh68g9w1w8qu) 7

[4.2 Food Classification Model:](#_wrc33u6u1dkd) 8

[4.3 Locally Run Model:](#_bylx95mfvf8o) 9

[**5. Team Member Contributions**](#_mh2aqc6hip4q) **9**

[**6. References**](#_zadhaib7kv5q) **10**

#### 

#### 

#### 

#### 

#### 

#### 

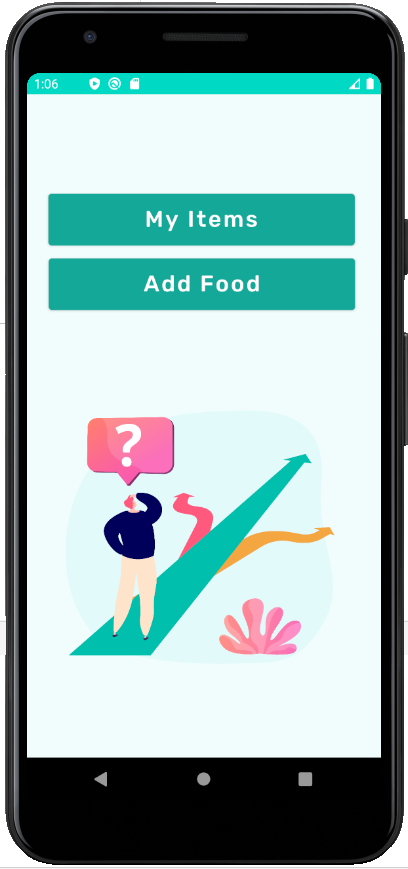
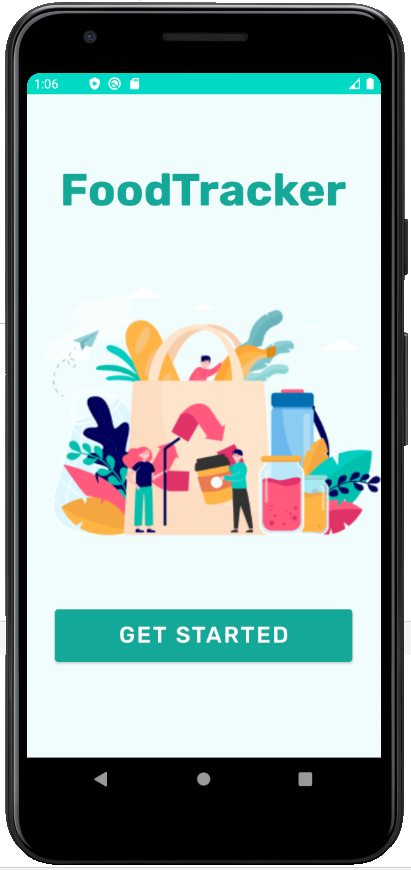
#### 

#### **1. Description**

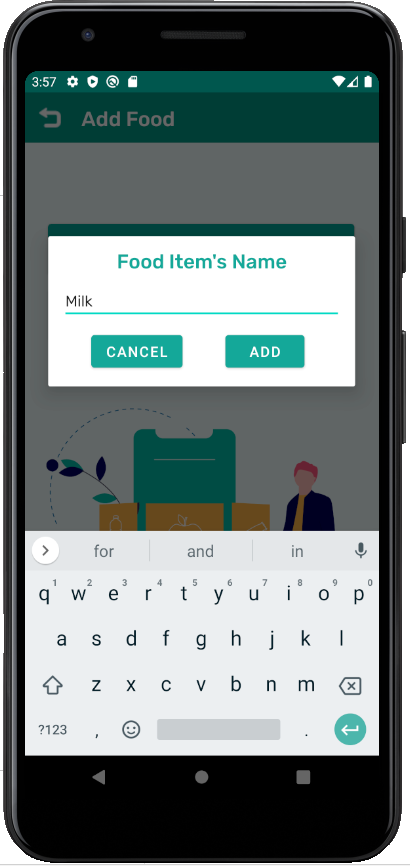
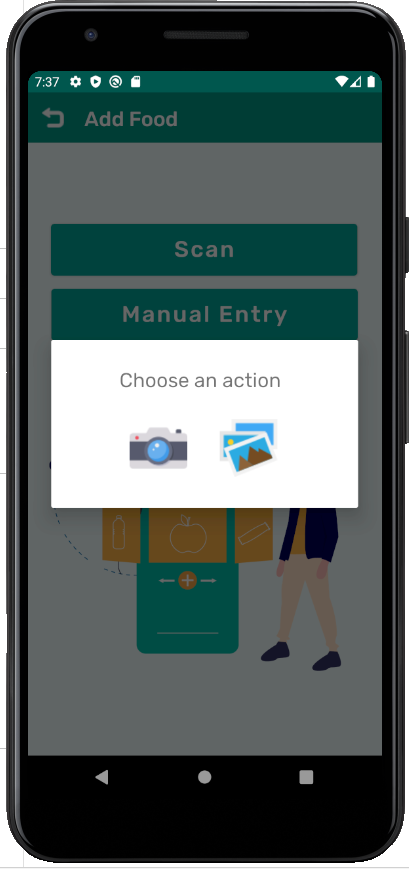
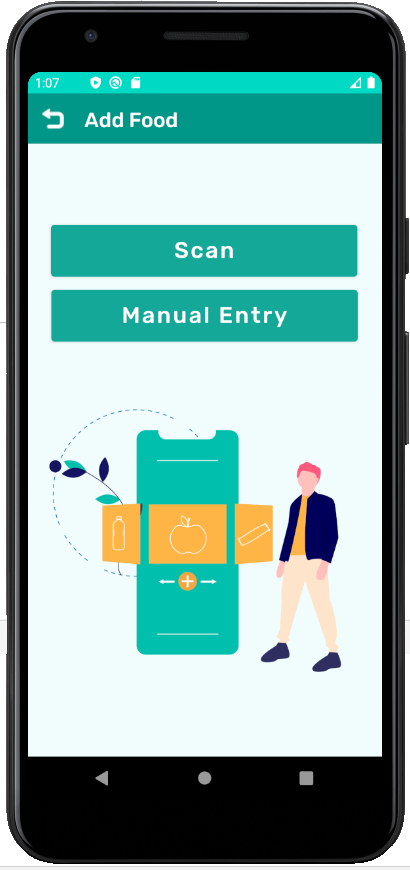
FoodTracker is a food waste management app that helps users manage their groceries and reduce food waste. The app holds an inventory of the current groceries that the user can view to tell which food items will expire soon. To add an item to the inventory, the app uses a machine learning model that recognizes food items in a photo taken by the user. The user can then verify the correct food item was identified and enter the corresponding expiration date. With these inputs, the app adds the item name and its expiration date in the items list which is stored in Google Cloud’s Firestore Database. The user can view and interact with their inventory of food with their expiration dates with the app’s real time listener to the cloud-hosted database. Moreover, FoodTracker includes a manual entry option in case of any inaccuracy in recognizing the item. This app uses Inceptionv3 pre-trained model for food items category classification.

#### **2. Operation Walkthrough**

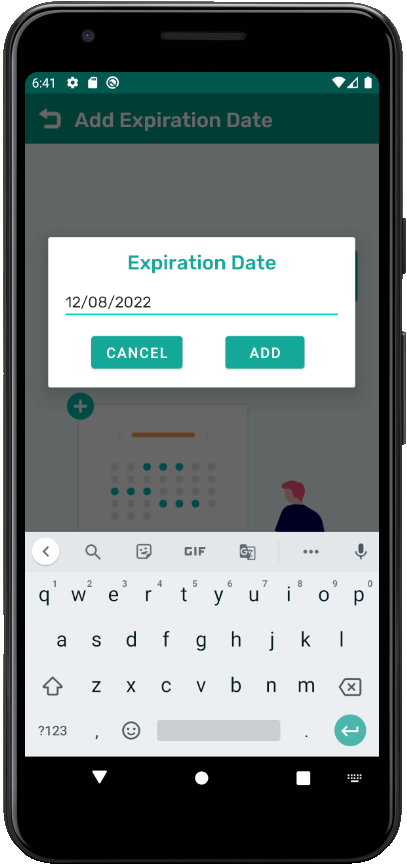
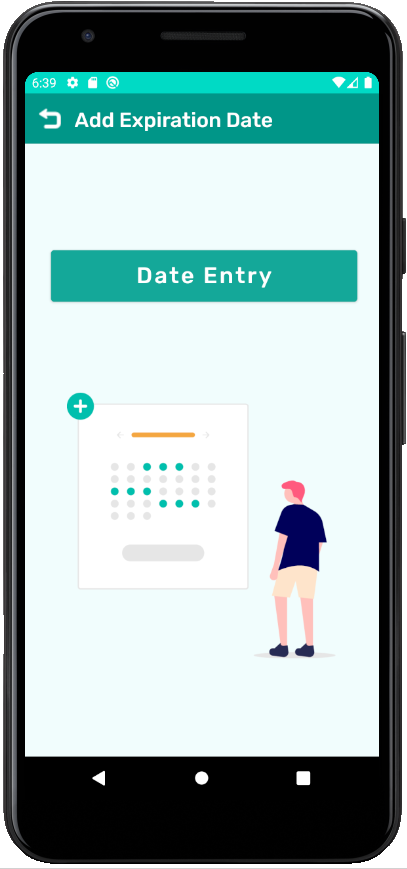
1. As the app is opened, the launch screen will appear.
2. By pressing Get Started Button, the app will move to the Main Menu screen. This screen includes moving to the inventory of the items by pressing the “My Items” button or adding a new food item by pressing the “Add Food” button.

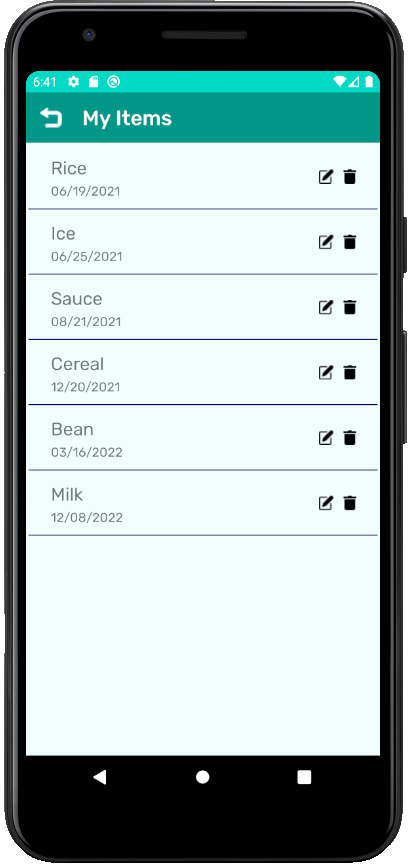


1. The “Add Food” button allows the user to add a new food item to the items list. As shown in figure , this screen has 2 options: Scan and Manual Entry. The scan button allows the user to take an image to the food item so the included machine learning model can recognize the item’s name and add it automatically to the list in the “My Items” screen. The Manual Entry button allows the user to write the item’s name manually in case of any inaccuracy to the food item recognition by the ML model or if the user prefers this option.



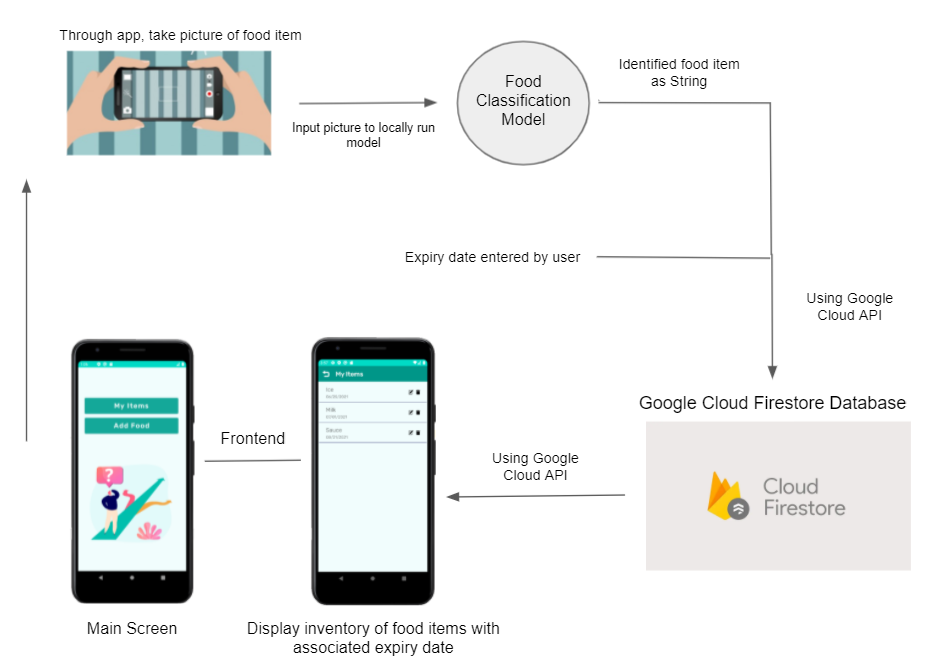
1. After adding the Food item’s name by pressing the “ADD” button, the app automatically moves to the Expiration Date menu. As shown in figure , the Expiration date menu screen has the “Date Entry” button that allows the user to write the item’s expiration date in the pop up dialog.





1. The “My Items” button navigates to a screen that includes the inventory list of food items and their expiration dates. It also includes delete and edit buttons for each item. As it can be seen in the image below, the food item that we added in this example, which is Milk, is included in the Items list with its expiration date.

#### **3. System Component Diagram**



#### **4. Component Descriptions**

##### **4.1 Cloud Firestore Backend:**

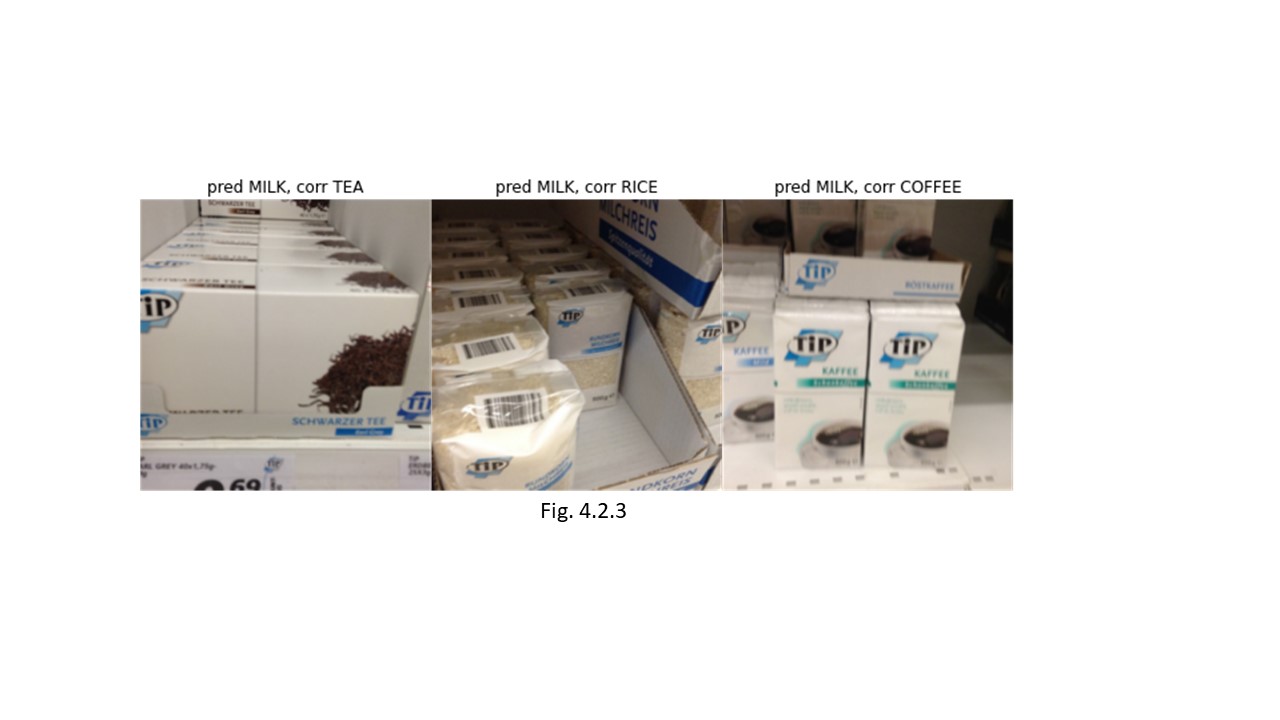
Using Cloud Firestore with our app allowed us to store food items and their expiration dates in a NoSQL, cloud-hosted database. We created and registered a Firestore database with our Android application and used the Google Cloud API to send data back and forth. The data in our database is represented with one main “collection” that contains “documents”. Each document represents a food item and its expiry date. Every time the user enters a new food item and expiry date, a new document is created from them and is added to the collection in Firestore. We implemented a real time listener to the database so when the user wants to view their inventory, they will always see the exact data in Firestore. Every document in the collection is read and converted into a custom “FoodItem” Java class. We then sort the FoodItem objects based on their expiry date and then display their inventory in the frontend.

##### **4.2 Food Classification Model:**

To implement the food item classification feature, we used the pre-trained InceptionV3 model, which was initially trained on ImageNet, and retrained it on the Freiberg Groceries Dataset. The model has 25 outputs corresponding to the 25 grocery classes in the dataset. As a result, the food classification model is limited to identifying the following 25 food items:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Beans  Cake  Candy  Cereal  Chips | Chocolate  Coffee  Corn  Fish  Flour | Honey  Jam  Juice  Milk  Nuts | Oil  Pasta  Rice  Soda  Spices | Sugar  Tea  Tomato sauce  Vinegar  Water |

Any other item would need to be inputted to the FoodTracker inventory manually. To train the model, we initialized all weights to 0 and to obtain better accuracy, we used all 5000 images in the dataset, 4000 of which were used for training. Using a batch size of 4, images were loaded into the model with a data loader and after training for 10 epochs, the accuracy converged to 59.2% with a loss of 0.0044. Upon experimentation, we found that no reasonable combination of image augmentation was able to improve the accuracy of the model. However, by using the AdamW optimizer, we were able to improve our accuracy to 62% and loss of 0.0032 after 10 epochs. After another 10 epochs, the accuracy increases slightly to 65% with a loss of 0.0029. Some of the categories that were involved with misclassifications were cereal, juice, and milk. However, having a look at these images, cereal was often misidentified as candy due to bright, colourful packaging with small bits of fruit that look like gummy candy (Figure 4.2.1). Also, juice and other liquids were sometimes mixed up since the images contained brightly coloured liquids in a bottle (Figure 4.2.2). Another interesting observation is that the model misclassified quite a few items as milk, but all of these objects were white and carton shaped (Figure 4.2.3), and even belonged to the same brand, Tip. When examining all the mispredictions, it can be noted that many of the misidentified items share similar packaging design features, such as jars, cans, bottles, and boxes; and displaying cups or fruit on the packaging. Other items with more distinct packaging, such as flour, chips, and spices, were rarely misclassified. In conclusion, the model still performed quite well despite the wide variety of different packaging designs for each class of food item and the many similarities between packaging designs of different food classes.



##### **4.3 Locally Run Model:**

The trained model is saved using Google Colaboratory and has been optimized to function on mobile. We then save the model in the assets folder in our android studio files. PyTorch and TorchVision dependencies are used for turning the bitmap image received from the camera into a tensor. The tensor is then run on the classifier model. The outputs are then sorted to get the prediction with the largest value. A class in Java that contains an array of strings is created as a lookup for the string associated with each prediction value we receive from the model (0-24). The string is then added to the Firestore database and thus is seen in the applications “Inventory” page.

#### **5. Team Member Contributions**

Sara:

Designing the User Interface of the app

Implementation of the Frontend

Integrating the ML model with the app

Moiz:

Implementation of OCR

Deploying the model to run it on the app

Fixing bugs in when training the model

Justin:

Implementation, debugging, and optimization of the Food Classification Model

Josh:

Implemented a Cloud Firestore database with the application to store the user’s data. Helped integrate these backend tasks with the frontend of the app. Performed research for different methods to integrate the machine learning models with our application such as deploying the model in Google Cloud’s AI Platform or AWS’ Sagemaker.

#### **6. References**

P. Jund, “PhilJd/freiburg\_groceries\_dataset,” *GitHub*, 11-Oct-2018. [Online]. Available: https://github.com/PhilJd/freiburg\_groceries\_dataset. [Accessed: 16-May-2021].