Cuisine Classification And Food Freshness Detection

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science And Engineering

by

PRABHAT CHANDA

18BCE2005

SUYASH AGARWAL

18BCE0273

Under the guidance of

Prof. / Dr. Swathi J N

SCOPE

VIT, Vellore.

May, 2022

DECLARATION

We hereby declare that the thesis entitled "Cuisine Classification And Food Freshness Detection" submitted by us, for the award of the degree of Bachelor of Technology in Computer Science And Engineering to VIT is a record of bonafide work carried out by us under the supervision of Dr. Swathi J N.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 26/05/2022

Prabhat Chanda

Suyash Agarwal

Signature of Candidate(s)

CERTIFICATE

This is to certify that the thesis entitled "Cuisine Classification And Food Freshness Detection" submitted by **Prabhat Chanda, 18BCE2005, SCOPE and Suyash Agrawal, 18BCE0273, SCOPE,** VIT University, for the award of the degree of *Bachelor of Technology in Computer Science And Engineering*, is a record of bonafide work carried out by him under my supervision during the period, 01. 12. 2021 to 31.05.2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Vellore

Date: **Signature of the Guide**

Internal Examiner **External Examiner**

Head of the Department

Programme

ACKNOWLEDGEMENTS

We would like to thank our guide Dr. Swathi J.N. for her continued guidance and mentorship, without her, our project could not have been completed. We would also like to thank all the teachers during the four years of our time here at VIT, who gave us all the knowledge about computer science and engineering. Many components of this project are only possible due to the knowledge imparted in many courses taught throughout the four years of this degree.

We would also like to give a special thanks to our respective friends and families for their continuous love, support and understanding throughout our time here in VIT.

Suyash Agarwal

Prabhat Chanda

Student(s) Name(s)

Executive Summary

In this project, we have attempted to create a low cost device that can be easily attached to a refrigerator and can keep track of food items inside it. It uses Machine Learning to identify a particular food item and predicts an expiry date for the same. This information is then used to remind the user about a food item that is about to expire so that it can be consumed and it does not go to waste, thereby saving money for the user as well. The project uses concepts of artificial intelligence, internet of things and internet and web programming to achieve its goal.

The device consists of a Raspberry PI, connected to a camera, which clicks the picture of the food item and sends the data to the server (Amazon EC2). Here, classification is done and the expiry date is predicted. This information is then relayed to the end point client via a web app which functions as a to-do list.

CONTENTS PAGE NO.

List of Figures

1. INTRODUCTION

1.1. OBJECTIVE

Our objective is to create a low cost, easy to use device that can take a picture of the food item, process the image to classify the category (cuisine) of the food item, use that information to predict the tentative expiry date of the food item, and set a reminder for the same on a web-app.

1.2. MOTIVATION

We aim to create a device that can be easily applied on any fridge like a fridge magnet. The device would be containing a battery, some buttons, a small camera and a raspberry pi. On the press of a button, the camera will take the picture of the food item being entered inside the fridge, and that picture is processed to classify the cuisine using machine learning, then this information goes to a database to get info regarding expiry which helps create a reminder for the user.

1.3. BACKGROUND

In our daily life, many times, some or the other food item goes to waste because it is not consumed while it's fresh. Even if the dish is being refrigerated, food does become unfit to consume after a few days. Modern smart fridges try to tackle this issue but they are ridiculously expensive and also don't offer that much functionality. So, to address this issue and to solve the root cause, we propose a small hardware device that can recognize what food item is being kept inside the fridge along with its cuisine and suggest a time period when it can be consumed. The user will also be given a reminder when a certain food item is going to get expired. The device would use image processing coupled with machine learning to classify what is the cuisine and type of food being kept. Then this information would go through a database which has records containing information regarding the expiry dates of those food items. A reminder would then be created on a web app that will accompany the hardware device and the user will be notified as and when the expiry date approaches. The hardware device can be simple put on the fridge

like a fridge magnet and on the press of a button, the food's image will be taken and a reminder would be fixed on a web-app.

We did the Literature Survey separately for Cuisine Classification and for Food Freshness Detection models and papers. In the case of Cuisine Classification, most of the models [1][2][3][4][5] used a pre-trained CNN Model. They applied a pre-trained model based on CNN architecture on the tasks of food / non-food image classification and food category recognition. They had essentially used CNN model to categorize different attributes for food such as the ingredients. For food recognition, they had gone for simple classification but for attribute recognition, they had used pre-trained vectors for the task. In another model, first, a set of transfer learning algorithms based on CNNs were leveraged for deep feature extraction. Then, a multi-class classification algorithm was exploited based on the performances of the classifiers on each deep feature set. Various Datasets were used in different papers and models, but the most common ones were Food-5k, Food-11, Food-101 and the UEC datasets.

When it comes to Food Freshness Detection, most of the models and papers used a technique to increase the accuracy and perfection of the food freshness detection with the help of size, shape and color based method with the union of Convolutional Neural Network (CNN). To restore physical examination of food, CNN was widely employed which gave them genuine, unbiased and constructive classification. In another model, they proposed a design of computer vision based technique using deep learning with the Convolutional Neural Network (CNN) model to detect fruit freshness. The specially designed CNN model was then evaluated with public datasets of fruits fresh and rotten for classification derived from Kaggle. In a completely different model [6], freshness of fruit was determined using Raspberry Pi [9]. Whenever the fruit was picked, it was placed in a conveyor belt, where it was passed through the sensor unit which could detect and display the complete freshness status of the fruit [7][8]. The proposed system started the process by capturing the fruit image by using Raspberry Pi. Then, the image was transmitted to the processing level where the fruit's characteristics like color, shape and size of fruit samples were extracted. After that by artificial neural network, fruit images went through the training and testing section.

2. PROJECT DESCRIPTION AND GOALS

Our device can easily be applied on any fridge like a fridge magnet. The device would be containing a battery, some buttons, a small camera and a raspberry pi. On the press of a button, the camera will take the picture of the food item being entered inside the fridge, and that picture is processed to classify the cuisine using machine learning, then this information goes to a database to get info regarding expiry which helps create a reminder for the user. The prediction part and the appropriate expiry date will be determined in the Firebase and that information will be passed on to the client. Our goal is to create a low cost, easy to use device that can take a picture of the food item, process the image to classify the category (cuisine) of the food item, use that information to predict the tentative expiry date of the food item, and set a reminder for the same on a web-app.

3. TECHNICAL SPECIFICATIONS

The following are the components used in the project:

(a) **Python:** The whole project is written in this language. Version used is 3.10. From the jupyter notebook to the web application, everything has been written in this language

(b) **HTML:** The index webpage is written in HTML along with inline CSS

(c) **Flask:** This is a python library which takes care of the backend part of the web app. Version: 2.1.2

(d) **Tensorflow:** This framework has been used during the training of the machine learning model. Version: 2.x

(e) **Keras:** This python API is used to write the jupyter notebook.

(f) **RaspberryPi 3B+:** This is the credit card sized computer used to take pictures of the food item and send it to the server

(g) **Pi Cam:** This is the camera module compatible with Raspberry Pi for taking pictures.

(h) **Amazon EC2 instance:** This is the cloud service running the database and backend program.

(i) **SQLite:** Database of the project. Version 3 (sqlite3)

4. DESIGN APPROACH AND DETAILS

4.1. DESIGN APPROACH, MATERIALS AND METHODS

As shown in Fig. 1, the device can be easily attached to the upper door of a fridge with the help of magnets. The camera unit is facing downwards which would take the picture of the food item being inserted at the press of a button. Inside the device, a Raspberry Pi 3B+ [9] with a camera attached to it takes the picture, processes the picture using machine learning to determine the cuisine of the food item, sends the data to a database online, which relays the data to the end user along with a reminder set according to the predicted expiry date. Since not all refrigerators do not have an orthodox two door design with the freezer unit on top of the fridge unit, we also came up with another design for these type of refrigerators as shown in Fig. 2. The box which houses the Raspberry Pi [9] sits on top surface of the fridge, with a camera unit protruding outwards to take picture of the food items. Rest of the working is the same.

The following are the designs for a 1-door and 2-door fridge setup respectively:

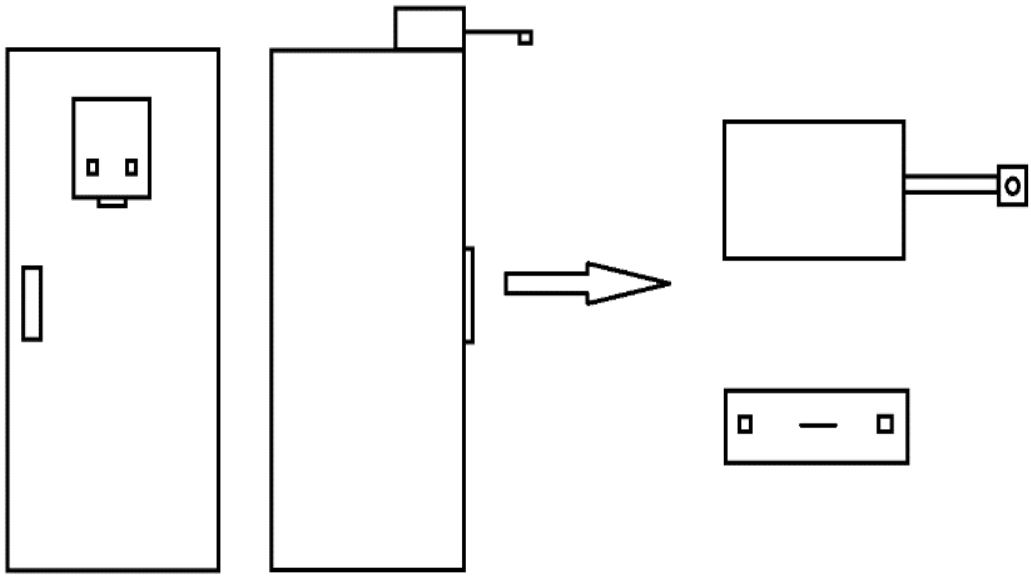


FIGURE 1: Design for 1-door fridge

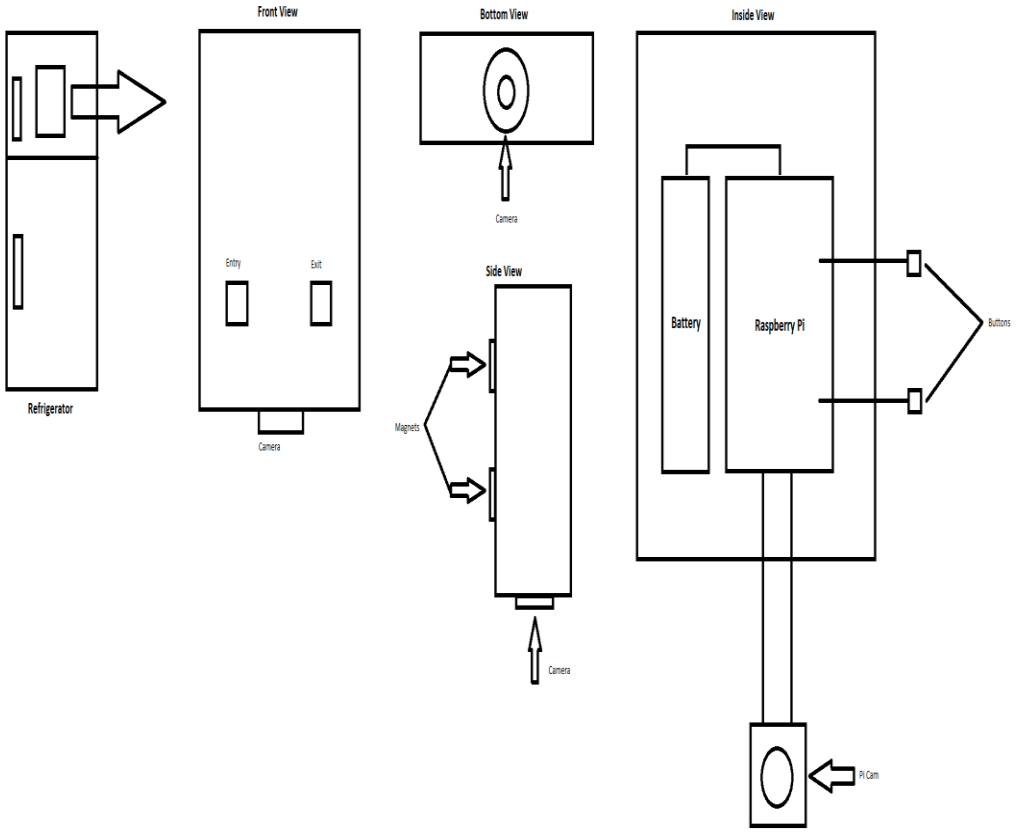
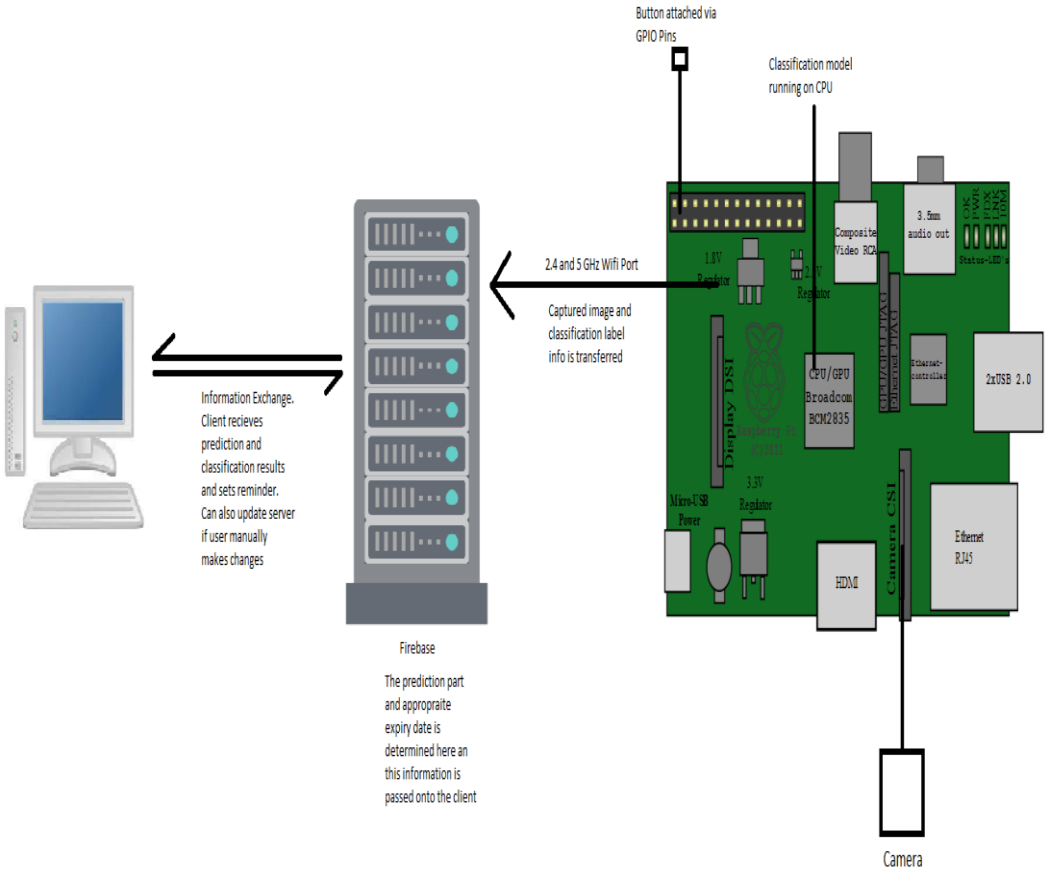


FIGURE 2: Design for 2-door fridge

We will be training a deep learning model to classify the cuisine of a specific food item and then running it on the raspberry pi. A small web app will be used as an interface for the user and all the components will communicate with each other via Amazon EC2 instance. Below is the architecture diagram and some UML diagram to explain the overview of this project.

The system architecture, as shown in Fig. 3, has three main parts, a Raspberry Pi 3B+, Google Firebase, End-User application. When a food item is entered in the refrigerator, the Raspberry Pi takes the picture controlling the camera module, classifies the food item according to the machine learning model and sends this data to the firebase for expiry date prediction and the database then relays this information to the client application which displays it in a user-friendly way. A set of UML diagrams (Fig. 4) (Fig. 5) (Fig. 6) will help better understand the methodology.

FIGURE 3: Architecture diagram



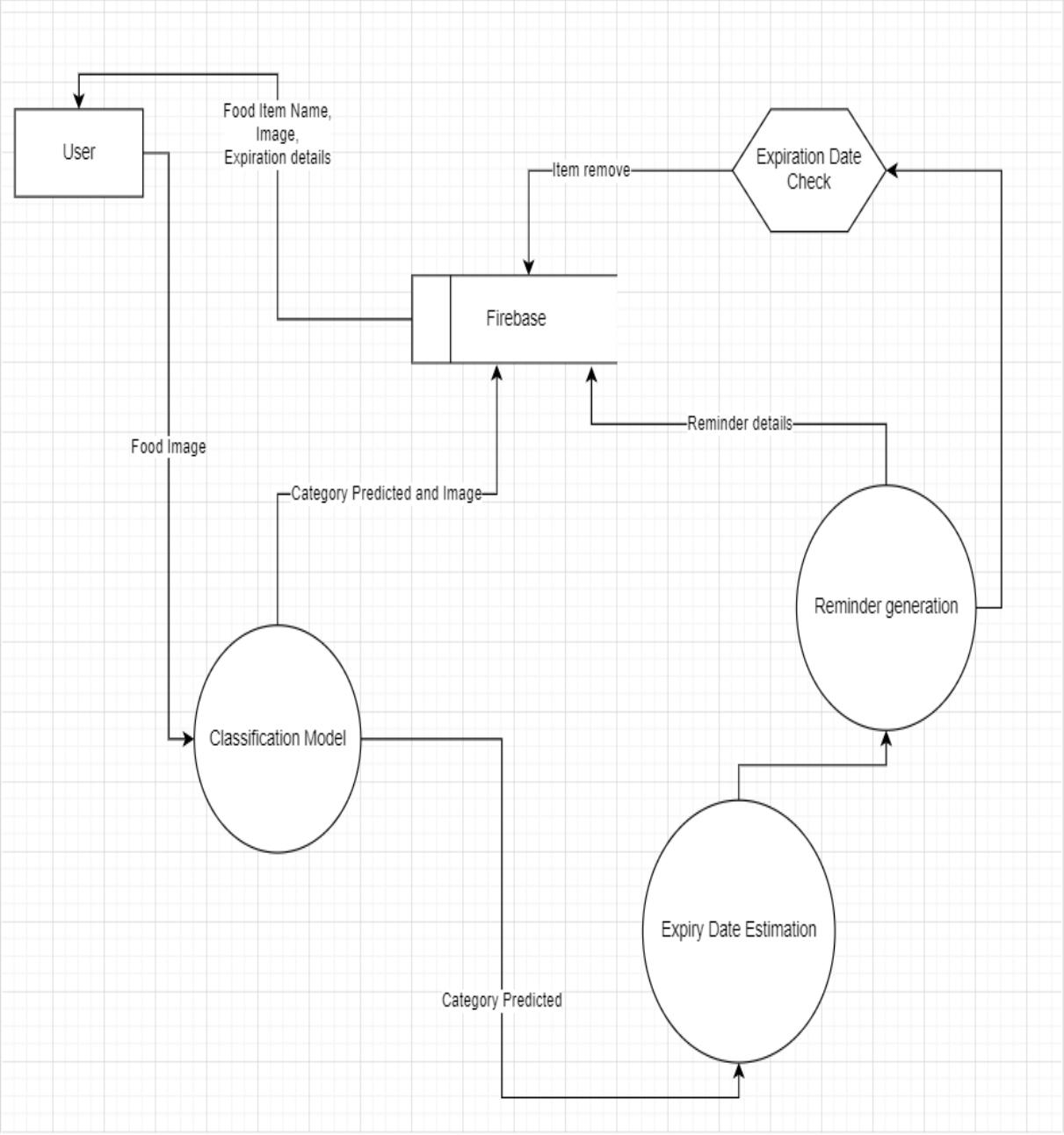


FIGURE 4: Data flow diagram

As shown in the dataflow diagram (Fig. 4), the user simply needs to press a button which activates a script that captures the image and sends it to the cloud computer which classifies the food image and sends this information along with the image to the database. The output also helps in the estimation of the expiry date which sets a reminder as well. As the expiry date approaches, the alarm reminds the user of the product to be consumed. The algorithm for it can be seen in (Fig. 5).

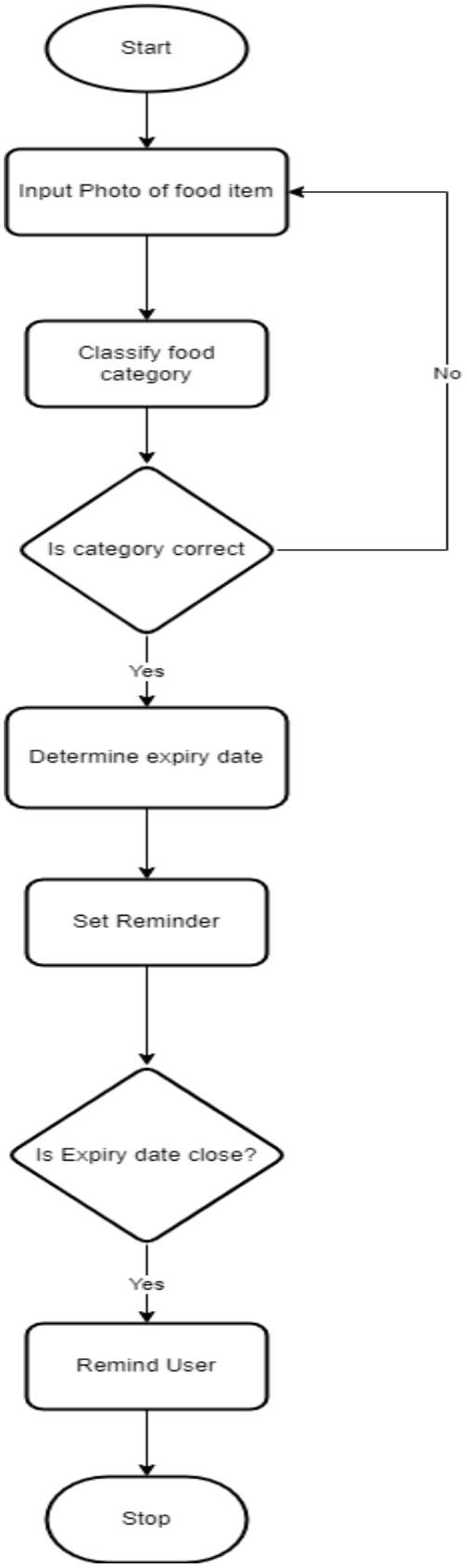


FIGURE 5: Flow chart

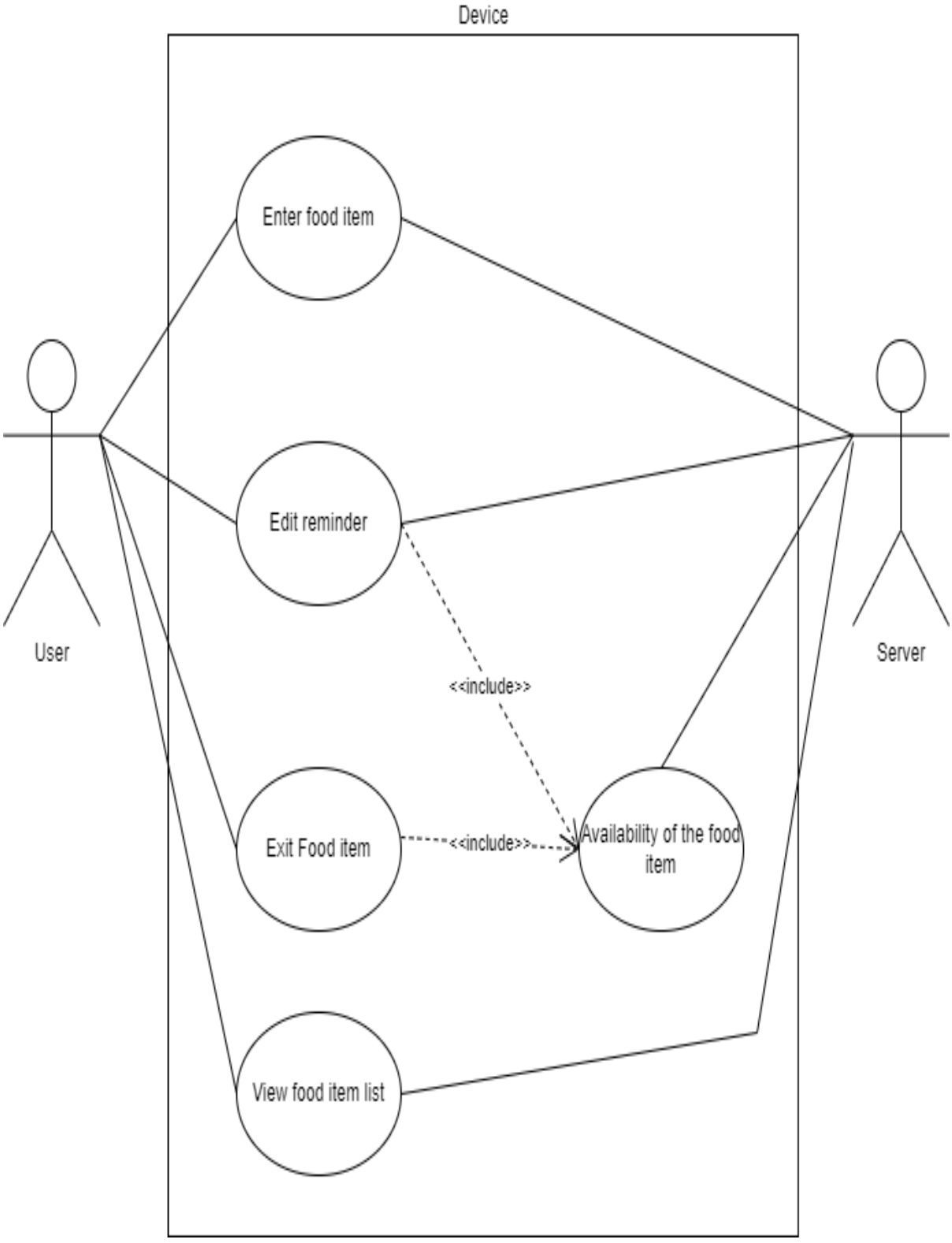


FIGURE 6: Use case diagram

Coming to the client side of the project, as illustrated in the use case diagram (Fig. 6), the user can simply press a button and the device takes the picture of the food held just below it.

Once this is done, the UI will display the name of the food item along with its estimated expiry date and the number of days recommended to consume within. There will also be a button to mark the food item as consumed which can easily work as an exit button. There is also and edit function where user can manually enter the details if they want.

MACHINE LEARNING ARCHITECTURE

The device is using a custom trained multi-classification convolutional neural networks model. The convolutional base used is of MobileNet V3 [11] with a set of classifier layers added to it in accordance with the dataset. The image goes through all the convolutional filters with pre-trained weights of the MobileNet base and then the classifier section is trained to classify according to the dataset. The end user application is a website created with the help of Flask [10]. All the communication APIs are also written in Python.

Going in a bit more detail, a convolutional neural network (CNN) is a type of artificial neural network that uses convolution filters (matrices that perform convolution operation) and feed forward perceptron network to analyse data, usually visual.

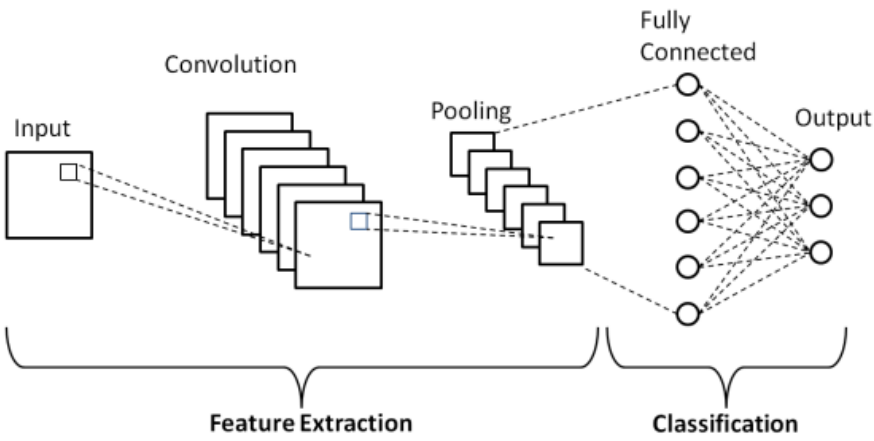


Fig. 7. CNN Model

LAYERS APPLIED:

MobileNet V2

Our CNN model is hybrid transfer learning model consisting of a pre-trained base of MobileNet V2 [11] which is trained on the Imagenet dataset. The idea behind using it because:

- Imagenet is a much broader dataset, hence the filter layers trained on it have better generalized features and are biased towards texture which ensures better accuracy and robustness [12].

- MobileNet in specific was created to be able to run on mobile devices with limited processing power

Global Average Pooling

Once the image goes through the multiple branching layers of mobilnet, all the matrices obtained are 2-D. So, global avaerage pooling applies average pooling on the spatial dimensions until each spatial dimension is one, and leaves other dimensions unchanged [14]. Hence condensing the model and also flattening it for feed forward layers

Dense Layers.

These are feed forward perceptron layers [15] added after pooling which do the final classification of the categories present in the dataset. In our model, we have added 3 layers with 128, 64 and 48 neurons respectively. Dropout is also added to ensure robustness.

DATASET

The dataset being used is Food-101 [13]. It is a collection of images 101 food categories, 1000 images for each category. In order to achieve higher accuracy, we reduced the amount of categories from 101 to 48. This worked exactly as planned and the model accuracy sits at 60.16% (validation).

END POINT

- **Client UI**

  The client UI as shown in (Fig. 10), is a simple web app created using HTML and CSS for frontend with backend running on Python Flask library. The web app is connected to an sqlite3 database where the information regarding food items is stored. The UI consists of a form for manually entering information and a list which shows the information stored in the database. The display consists of only the food items that have not been consumed yet. A button is also added beside each list entry to mark it as consumed.

- **Database**

  The database, as mentioned before, is an SQLite database (version 3). This is an SQL database that is stored locally as a .db file. The database consists of 3 columns, which are the name of the food item (String data type), expiry date (DATE data type) and a flag value of 0 or 1 (Boolean data type) to keep track if the food item has been consumed or not.

4.2 CONSTRAINTS, ALTERNATIVES AND TRADEOFFS

Throughout the making of this of project, we can across many challenges. Firstly, since our dataset had 101 food categories and not much data for each class, the accuracy (testing) we were getting was very low to reliably make a prediction. To improve that, we reduced the number of categories to 48 and managed to attain a decent 60% accuracy. Then we ran into various issues while running the prediction code (model_exec.py) on raspberry due to some libraries not having required version support on raspberry to run the model effectively. To counter that, we did the model heavy lifting on Amazon EC2 cloud instance where our website is also deployed.

One notable constraint of our device is that it can only keep track of cooked food items inside the refrigerator. It cannot yet track packaged food items which would make it a comprehensive tracker.

5. SCHEDULES, TASKS AND MILESTONES

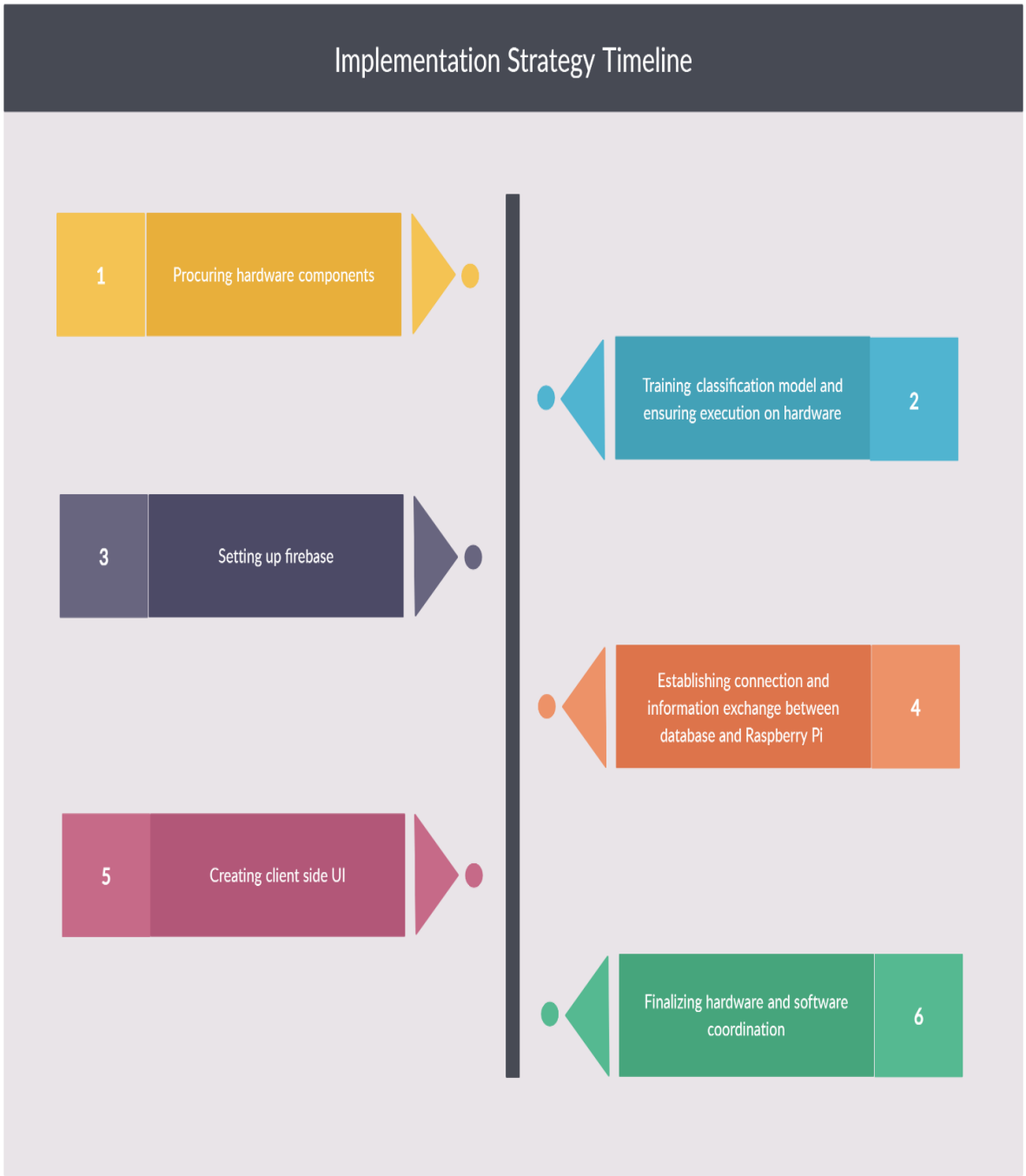


Figure 8: Implementation strategy

6. PROJECT DEMONSTRATION

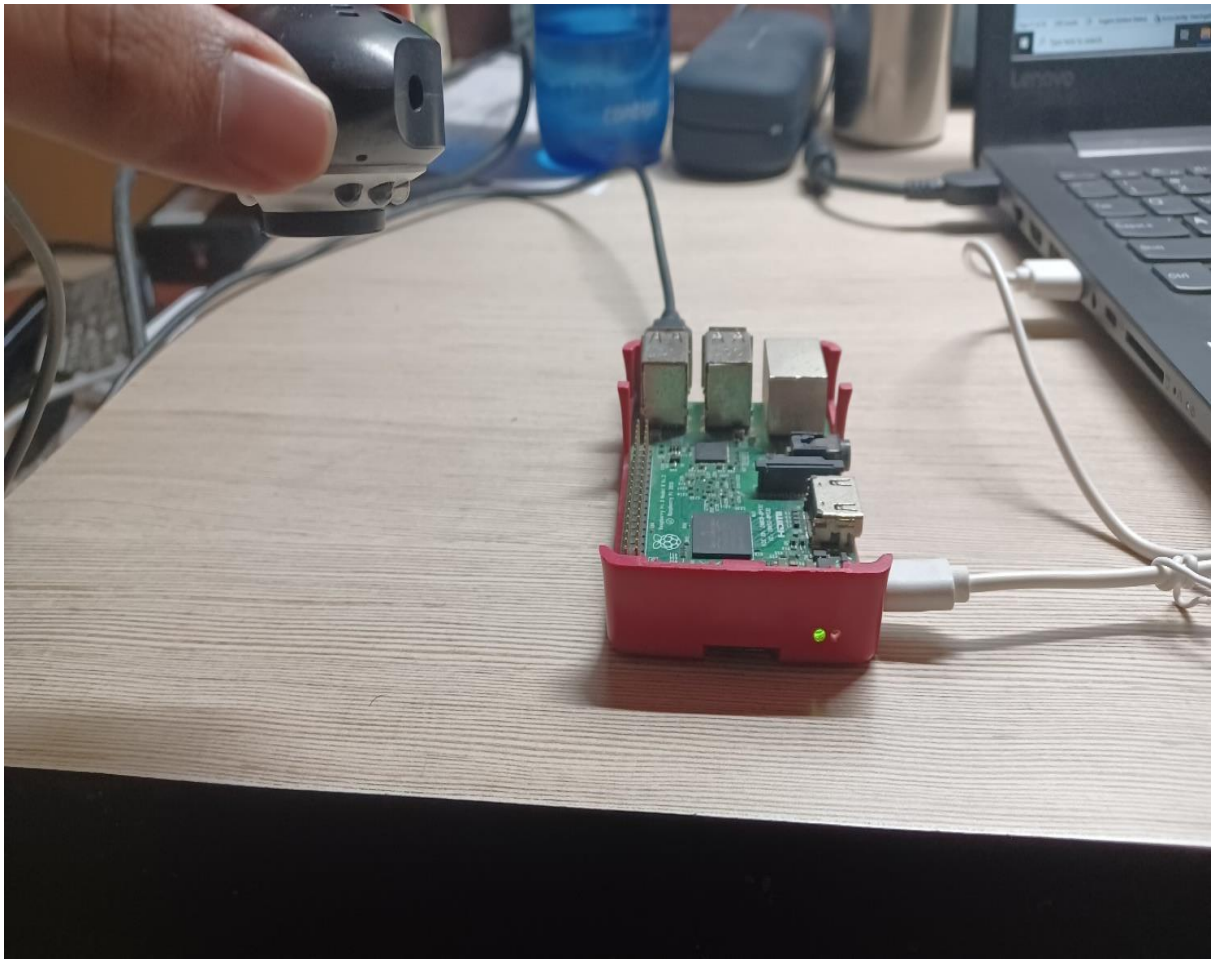


FIGURE 9: PiCam taking picture

The raspberry pi takes the picture of a food item, and sends it to the database. There the prediction is done and the details are relayed to the web page as shown below. A snapshot of the database is also visible for the same.

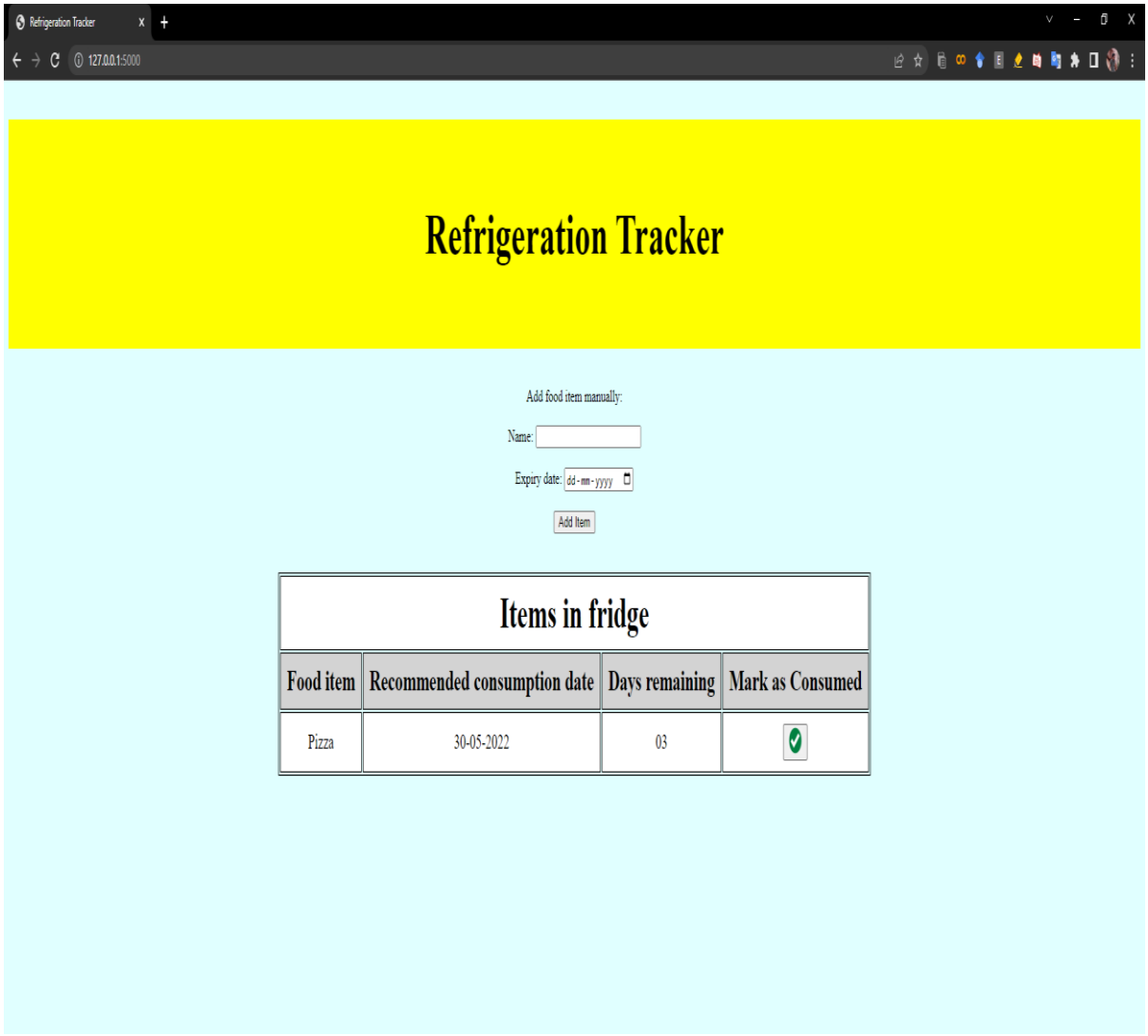


FIGURE 10: Client UI

An option is also given in the web page for the user to add items manually if for some reason the item is not correctly predicted.

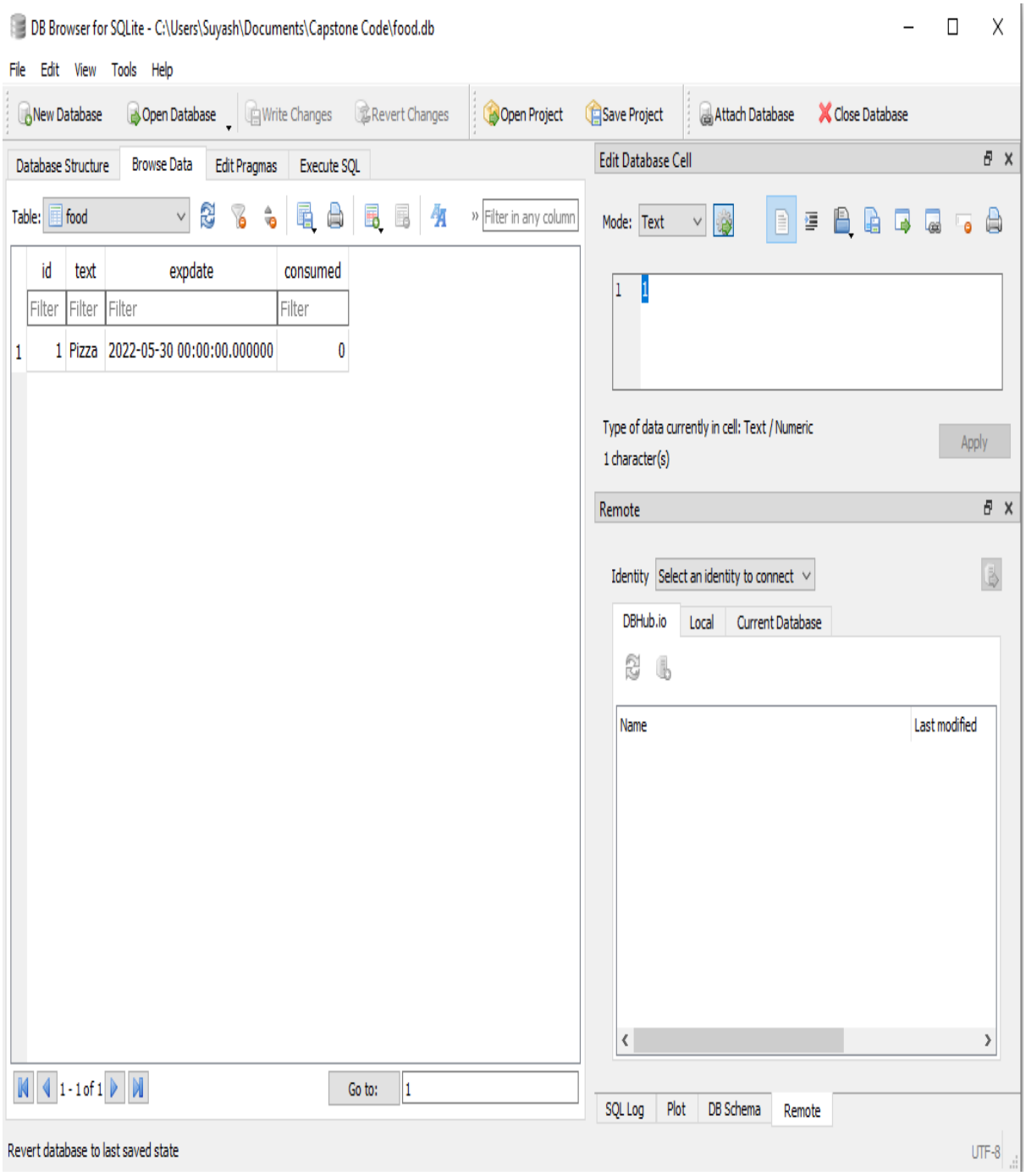


FIGURE 11: Database

7. COST ANALYSIS / RESULTS AND DISCUSSION

After reducing the categories from 101 to 48, we achieved a training accuracy of 79% and validation accuracy of 60.16% as shown in Fig. 6

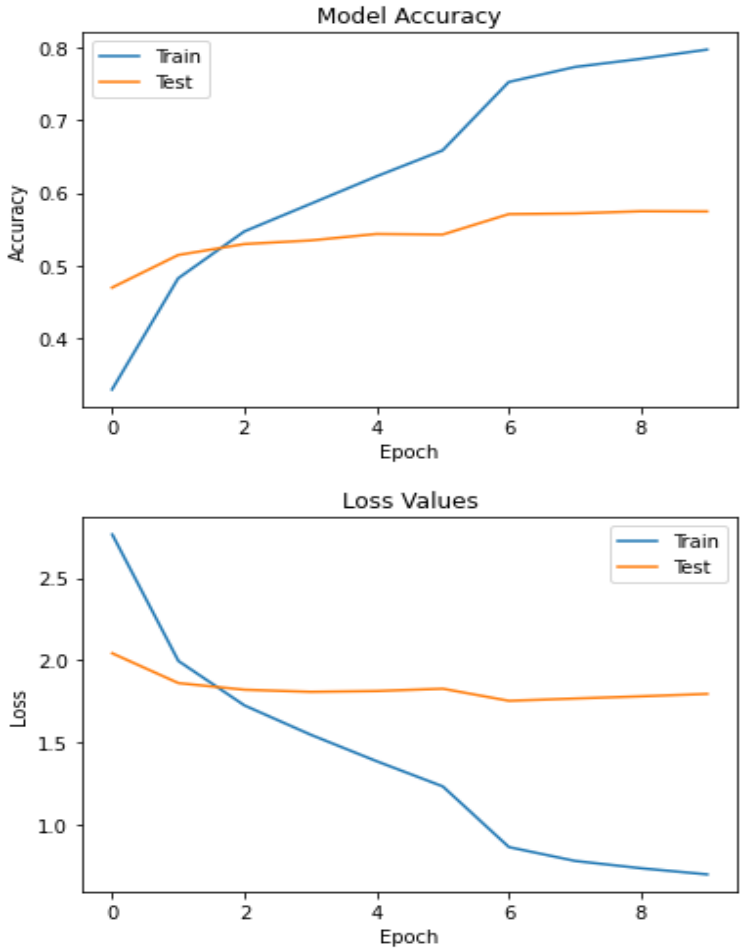


Figure 12: Training Results

Also regarding the cost of the device. The raspberry pi and camera module costed a combined of Rs. 4000. For production level prototype, an additional cost will be required for setting up circuit that powers the raspberry pi at the required voltage.

8. SUMMARY

In this project, we attempted to create a cheap and compact device that would keep track of food items being stored inside the refrigerator in order to remind the user if a particular food item is getting expired and needs to consume soon. The solution created is an IoT device that can be attached to the door of the fridge like a fridge magnet and helps reduce food wastage. We have harnessed the power of machine learning to classify and predict the cuisine category and expiry date and remind the user about the same as the time approaches.

As for this project, the device we created can only keep track of cooked dishes belonging to various cuisines. In the future, increasing the number of such cooked dishes would be a logical next step. Also, we plan on adding a barcode reading functionality so that it can keep track of packaged food items according to their expiry dates as provided by the manufacturer. Along with this, we aim to train our machine learning model on pictures of fruits and vegetables so that it can predict when they could rot and notify the user accordingly.

9. References

[1.] Singla, Ashutosh & Yuan, Lin & Ebrahimi, Touradj. (2016). Food/Non-food Image Classification and Food Categorization using Pre-Trained GoogLeNet Model. 3-11. 10.1145/2986035.2986039.

[2.] Liu, Chang & Cao, Yu & Luo, Yan & Chen, Guanling & Vokkarane, Vinod & Ma, Yunsheng. (2016). DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment.

[3.] Yunus, Raza & Arif, Omar & Afzal, Hammad & Amjad, Muhammad & Abbas, Haider & Bokhari, Hira & Haider, Syeda & Zafar, Nauman & Nawaz, Raheel. (2018). A Framework to Estimate the Nutritional Value of Food in Real Time Using Deep Learning Techniques. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2879117.

[4.] Pouladzadeh, Parisa & Yassine, Abdulsalam & Shirmohammadi, Shervin. (2015). FooDD: Food Detection Dataset for Calorie Measurement Using Food Images. Lecture Notes in Computer Science. 9281. 10.1007/978-3-319-23222-5_54.

[5.] Attokaren, David Joseph & Fernandes, Ian & Sriram, Anirudh & Srinivasa Murthy, Y.V. & Koolagudi, Shashidhar. (2017). Food classification from images using convolutional neural networks. 2801-2806. 10.1109/TENCON.2017.8228338.

[6.] Krithika Jayasankar, Karthika B, Jeyashree T, Deepalakshmi R, Karthika G. (2018). Fruit Freshness Detection Using Raspberry PI. Volume 1, Issue 10, DOI: 10.29027/IJIRASE.v1.i10.2018.202-208

[7.] Valentino, Febrian & Cenggoro, Tjeng Wawan & Pardamean, Bens. (2020). A Design of Deep Learning Experimentation for Fruit Freshness Detection.

[8.] Aniket Harsh, Kishan Kumar Jha, Shashwat Srivastava, Abhinav Raj, Raghav S. (2020). FRUIT FRESHNESS DETECTION USING CNN APPROACH. Volume:02/Issue:06/June-2020. IRJMETS

[9.] Raspberry Pi 3B+ reference manual: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[10.] Flask: https://flask.palletsprojects.com/en/2.1.x/

[11.] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional

neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017

[12.] Geirhos, Robert, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." *arXiv preprint arXiv:1811.12231* (2018).

[13.] Bossard, L., Guillaumin, M., Van Gool, L. (2014). Food-101 – Mining Discriminative Components with Random Forests. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8694. Springer, Cham. https://doi.org/10.1007/978-3-319-10599-4_29

[14.] Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400* (2013).

[15.] M. Mishra and M. Srivastava, "A view of Artificial Neural Network," 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), 2014, pp. 1-3, doi: 10.1109/ICAETR.2014.7012785.

10. APPENDIX A

CODES AND STANDARDS

(a) Machine Learning Part

The following screenshots are of the ML model trained:

Importing the Libraries ¶

In [2]:

```
import tensorflow as tf
import matplotlib.image as img
%matplotlib inline
import numpy as np
from collections import defaultdict
import collections
from shutil import copy
from shutil import copytree, rmtree
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
import os
import random
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPool2D, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.regularizers import l2
from tensorflow import keras
from tensorflow.keras import models
import cv2
```

In [3]:

```
# GPU Check
print(tf.__version__)
print(tf.test.gpu_device_name())
```

```
2.6.2
/device:GPU:0
```

```
2022-04-07 18:13:50.945131: I tensorflow/core/platform/cpu_feature_guard.cc:142] This Tensor
Flow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the followi
ng CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-04-07 18:13:50.997044: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:13:51.097580: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:13:51.098651: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:13:53.244397: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:13:53.245523: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:13:53.246490: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:13:53.247364: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created
device /device:GPU:0 with 15403 MB memory:  -> device: 0, name: Tesla P100-PCIE-16GB, pci bu
s id: 0000:00:04.0, compute capability: 6.0
```

Creating Label list

In [4]:

```
# Creating label list
food_list = ['apple_pie',
 'bread_pudding',
 'breakfast_burrito',
 'caesar_salad',
 'carrot_cake',
 'cheesecake',
 'chicken_curry',
 'chicken_quesadilla',
 'chicken_wings',
 'chocolate_cake',
 'chocolate_mousse',
 'churros',
 'club_sandwich',
 'cup_cakes',
 'donuts',
 'dumplings',
 'fish_and_chips',
 'french_fries',
 'french_toast',
 'fried_rice',
 'frozen_yogurt',
 'garlic_bread',
 'grilled_cheese_sandwich',
 'hamburger',
 'hot_and_sour_soup',
 'hot_dog',
 'ice_cream',
 'lasagna',
 'macaroni_and_cheese',
 'nachos',
```

In [6]:

```
def prepare_data(filepath, src, dest):
    class_images = defaultdict(list)
    with open(filepath, 'r') as txt:
        paths = [read.strip() for read in txt.readlines()]
        for p in paths:
            food = p.split('/')
            if food[0] in food_list:
                class_images[food[0]].append(food[1]+'.jpg')
        for food in class_images.keys():
            print("\nCopying images into ",food)
            if not os.path.exists(os.path.join(dest, food)):
                os.makedirs(os.path.join(dest, food))
            for i in class_images[food]:
                copy(os.path.join(src, food, i), os.path.join(dest, food, i))
```

In [7]:

```
# Creating train and test folder
%cd /
print("Creating train data...")
prepare_data('/kaggle/input/food-101/food-101/food-101/meta/train.txt', '/kaggle/input/food-101/
food-101/food-101/images', 'train')

print("Creating test data...")
prepare_data('/kaggle/input/food-101/food-101/food-101/meta/test.txt', '/kaggle/input/food-101/f
ood-101/food-101/images', 'test')
```

```
/
Creating train data...

Copying images into  apple_pie

Copying images into  bread_pudding
```

In [8]:

```
# File check
print("Total number of samples in train folder")
!find train -type d -or -type f -printf '.' | wc -c
print("Total number of samples in test folder")
!find test -type d -or -type f -printf '.' | wc -c
```

```
Total number of samples in train folder
36000
Total number of samples in test folder
12000
```

Preparing training and test set

In [9]:

```
K.clear_session()
n_classes = 101
img_width, img_height = 256, 256
train_data_dir = 'train'
validation_data_dir = 'test'
nb_train_samples = 36000
nb_validation_samples = 12000
batch_size = 16

train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
```

```
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')
```

```
Found 36000 images belonging to 48 classes.
Found 12000 images belonging to 48 classes.
```

Main Model

In [10]:

```
conv_base = MobileNetV2(weights='imagenet', include_top=False)
x = conv_base.output
x = GlobalAveragePooling2D()(x)
x = Dense(128,activation='relu')(x)
x = Dense(64,activation='relu')(x)
x = Dropout(0.2)(x)

predictions = Dense(48, activation='softmax')(x)

model = Model(inputs=conv_base.input, outputs=predictions)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
checkpointer = ModelCheckpoint(filepath='best_model.h5', save_best_only=True)
earlystop = EarlyStopping(monitor="val_loss", patience=3)

history = model.fit(train_generator,
```

```
history = model.fit(train_generator,
                    steps_per_epoch = nb_train_samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    epochs=20,
                    verbose=1,
                    callbacks=[checkpointer, earlystop])

model.save('model_trained.h5')
```

```
2022-04-07 18:18:51.430428: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:18:51.431650: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:18:51.432632: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:18:51.433868: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:18:51.434869: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:18:51.435900: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:18:51.436880: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
A node, so returning NUMA node zero
2022-04-07 18:18:51.437839: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] succ
essful NUMA node read from SysFS had negative value (-1), but there must be at least one NUM
```

```
/opt/conda/lib/python3.7/site-packages/keras/utils/generic_utils.py:497: CustomMaskWarning:
Custom mask layers require a config and must override get_config. When loading, the custom m
ask layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)


Epoch 2/20
2250/2250 [==============================] - 353s 157ms/step - loss: 2.3351 - accuracy: 0.36
20 - val_loss: 4.3727 - val_accuracy: 0.2272
Epoch 3/20
2250/2250 [==============================] - 357s 158ms/step - loss: 2.0185 - accuracy: 0.44
60 - val_loss: 2.2886 - val_accuracy: 0.4444
Epoch 4/20
2250/2250 [==============================] - 362s 161ms/step - loss: 1.8187 - accuracy: 0.50
15 - val_loss: 3.0223 - val_accuracy: 0.3749
Epoch 5/20
2250/2250 [==============================] - 359s 160ms/step - loss: 1.6784 - accuracy: 0.54
12 - val_loss: 2.1149 - val_accuracy: 0.4797
Epoch 6/20
2250/2250 [==============================] - 360s 160ms/step - loss: 1.5638 - accuracy: 0.56
77 - val_loss: 1.6699 - val_accuracy: 0.5714
Epoch 7/20
2250/2250 [==============================] - 364s 162ms/step - loss: 1.4556 - accuracy: 0.60
00 - val_loss: 1.6841 - val_accuracy: 0.5559
Epoch 8/20
2250/2250 [==============================] - 360s 160ms/step - loss: 1.3659 - accuracy: 0.62
30 - val_loss: 2.4371 - val_accuracy: 0.4672
Epoch 9/20
2250/2250 [==============================] - 354s 157ms/step - loss: 1.2879 - accuracy: 0.64
24 - val_loss: 1.5459 - val_accuracy: 0.6000
Epoch 10/20
2250/2250 [==============================] - 364s 162ms/step - loss: 1.2344 - accuracy: 0.66
04 - val_loss: 1.6517 - val_accuracy: 0.6156
```

Custom prediction

In [12]:

```
img = image.load_img('/test/pizza/3248869.jpg', target_size=(256, 256))
img = image.img_to_array(img)
img = np.expand_dims(img, axis=0)
img /= 255
pred = model.predict(img)
index = np.argmax(pred)
food_list.sort()
pred_value = food_list[index]
plt.imshow(img[0])
plt.axis('off')
plt.title(pred_value)
plt.show()
```

(b) Model_exec.py

This code runs the ML model and makes the prediction:

index.html | model_exec.py | app.py

```
from tensorflow.keras.models import load_model
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image

def model_predict(img_path, model):
    img = image.load_img(img_path, target_size=(256, 256))

    # Preprocessing the image
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    preds = model.predict(x)
    result = np.argmax(preds)
    return result

#def main():
MODEL_PATH = 'model.h5'
file_path = 'sample_model4.jpg'
model = tf.keras.models.load_model(MODEL_PATH)
result = model_predict(file_path, model)
print(result)
```

(c) Web App

The following two screenshots are of the code used to run the web application:

index.html app.py

```python
from flask import Flask, render_template, request, redirect, url_for
from datetime import datetime
from flask_sqlalchemy import SQLAlchemy
from time import strftime
import time

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///C:/Users/Suyash/Documents/Capstone Code/food.db'

db = SQLAlchemy(app)

class Food(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.String(200))
    expdate = db.Column(db.DateTime, default=datetime.now)
    consumed = db.Column(db.Boolean)

@app.route('/')
def index():
    incomplete = Food.query.filter_by(consumed=False).all()
    consumed = Food.query.filter_by(consumed=True).all()
    today = datetime.now()

    return render_template('index.html', incomplete=incomplete, today=today, time=time)

@app.route('/add', methods=['POST'])
def add():
    food = Food(text=request.form['fooditem'], expdate=datetime.strptime(request.form['expdate'], '%Y-%m-%d'), consumed=False)
    db.session.add(food)
    db.session.commit()

    return redirect(url_for('index'))

@app.route('/consumed/<id>')
def consumed(id):

    food = Food.query.filter_by(id=int(id)).first()
    food.consumed = True
    db.session.commit()

    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
        h1 {text-align: center; padding: 75px; background-color: yellow;}
        .displaydiv {text-align: center;}
        .itemadder {text-align: center;}
        th {padding: 10px;}
        td {padding: 10px;}
        tr:nth-child(even) {background-color: lightgray;}
        tr:nth-child(odd) {background-color: white;}
    </style>
    <title>Refrigeration Tracker</title>
</head>
<body style="background-color: lightcyan;">
    <h1 style="font-size: 40pt;">Refrigeration Tracker</h1>
    <div class="itemadder">Add food item manually:
        <form action="{{ url_for('add') }}" method="POST">
            <br/>Name: <input type="text" name="fooditem"><br/><br/>
            Expiry date: <input type="date" name="expdate"><br/><br/>
            <input type="submit" value="Add Item">
        </form>
    </div>
    <br/>
    <br/>
    <div class="displaydiv">
        <table border="1" align="center" bordercolor="black">
            <tr>
                <th colspan="4" style="font-size: 30pt;">
                    Items in fridge
                </th>
            </tr>
            <tr>
                <th colspan="1" style="font-size: 20pt;">
                    Food item
                </th>
                <th colspan="1" style="font-size: 20pt;">
                    Recommended consumption date
                </th>
                <th colspan="1" style="font-size: 20pt;">
                    Days remaining
                </th>
                <th colspan="1" style="font-size: 20pt;">
                    Mark as Consumed
                </th>
            </tr>
            {% for food in incomplete %}
                <tr>
                    {% set diff = time.strftime("%d", time.gmtime((food.expdate - today).total_seconds())) %}
                    <td style="font-size: 15pt;">{{ food.text }}</td>
                    <td style="font-size: 15pt;">{{ food.expdate.strftime("%d-%m-%Y") }}</td>
                    <td style="font-size: 15pt;">{{ diff }}</td>
                    <td><a href="{{ url_for('consumed', id=food.id) }}"><button><img src="{{url_for('static', filename='tick logo.png') }}" width="25" height="25" /></button></a></td>
                </tr>
            {% endfor %}
        </table>
    </div>
</body>
</html>
```