FreshiFy

INTELLIGENT FOOD SHELF–LIFE MONITORING SYSTEM

R25 - 073

SLIIT FACULTY OF COMPUTING

INTRODUCTION

PERERA P.A.E.U.K
IT21829710

PERERA B.C.V
IT20196110

FERNANDO M.G.S.S.A
IT19970882

KULARATHNA E.M.S.D
IT21828720

Why Freshify?

1.3 billion tons of food wasted globally each year → $1 trillion loss + 8–10% GHG emissions.

- **Sri Lanka's Challenge**

  Households waste 20–30% of purchased food mainly due to poor storage and lack of real-time monitoring.

A smart, affordable system using AI + IoT to monitor freshness, predict shelf-life, and reduce waste.

Research Problem

How can an integrated AI-IoT system be developed to address the inefficiencies in current household food management , such as : lack of real-time spoilage detection, environmental monitoring, storage compatibility, and dynamic recipe suggestions to effectively reduce food waste by 25-30% in tropical climates like Sri Lanka?

Objective

- **Main Objective**

Develop an Intelligent Food Shelf-Life Monitoring System using AI, IoT, and ML to significantly reduce household food waste, enhance storage compatibility, and improve management efficiency, promoting sustainable consumption.

Sub Objectives

- **Real-Time Food Identification and Spoilage**

  Implement a subsystem using CNNs for image analysis and MQ135 sensors for ethylene detection, achieving 92% accuracy across diverse foods (fruits, vegetables, curries) to provide timely health risk alerts via mobile app.

- **Shelf-Life Estimation and Prediction**

  Develop models that blend data from sensors (like temperature, humidity, and ethylene levels) with MobileNetV2 tech and the Q10 equation to predict shelf life accurately within about 1.2 days error, helping users plan better and eat food before it goes bad.

Sub Objectives

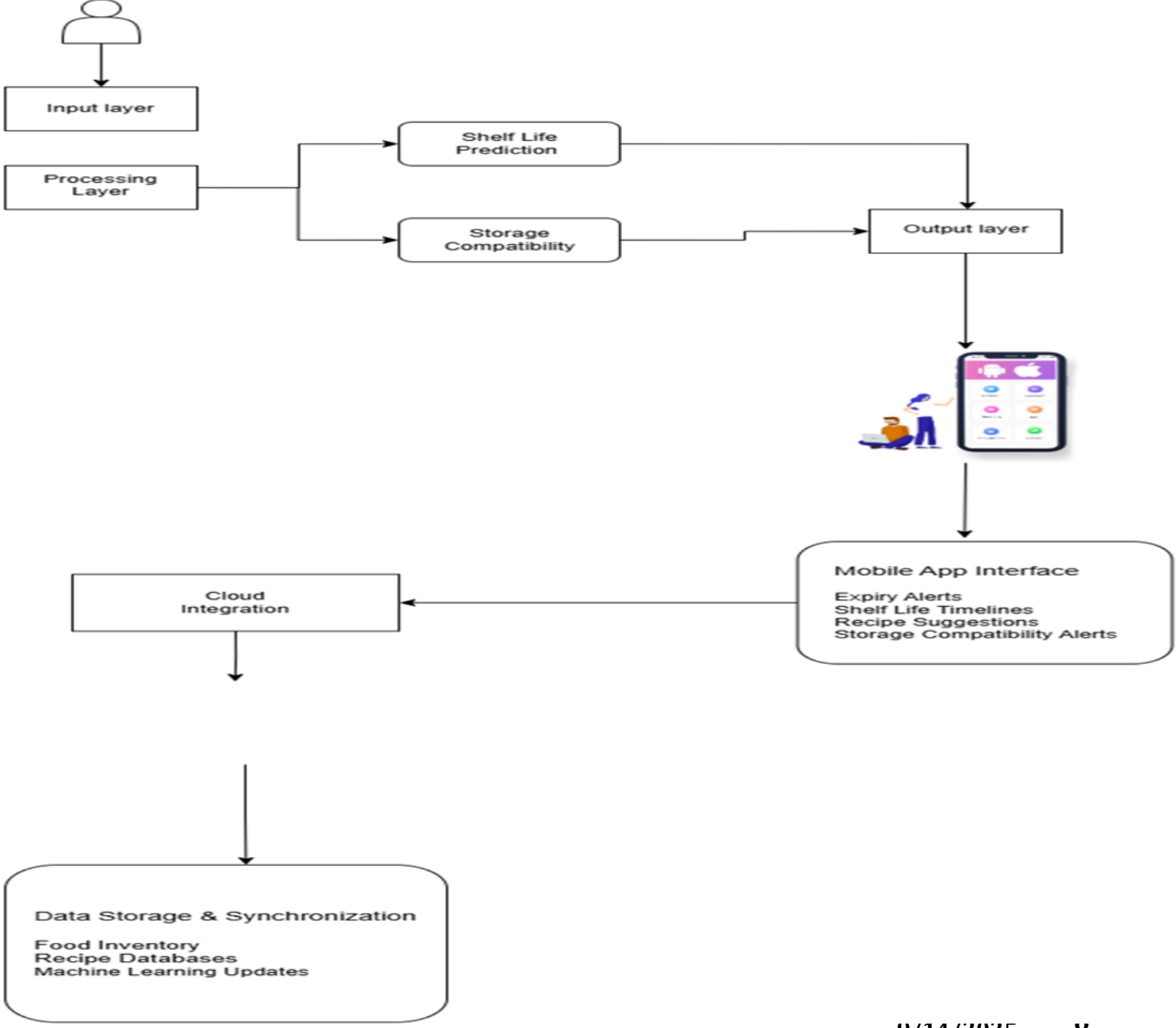
- **Storage Compatibility and Real-Time Monitoring**

  Apply simple AI rules to spot bad food combos that speed up spoiling (with 85% success rate), then offer tips to fix storage setups and stop waste before it happens.

- **Recipe Suggestion and Expiration Management**

  Suggesting Recipes and Handling Expiring Items

System Diagram



FreshiFy

Estimating Remaining Shelf Life of Food Items

IT21828720 | Kularathna E.M.S.D

SPECIALIZING IN INFORMATION TECHNOLOGY

COMPONENT -1

Introduction

FreshiFy

This component is a key part of our innovative food management system, leveraging AI, sensor data, and custom logic to predict and manage food expiry. It helps users reduce waste and maintain fresh inventory through smart predictions and alerts.

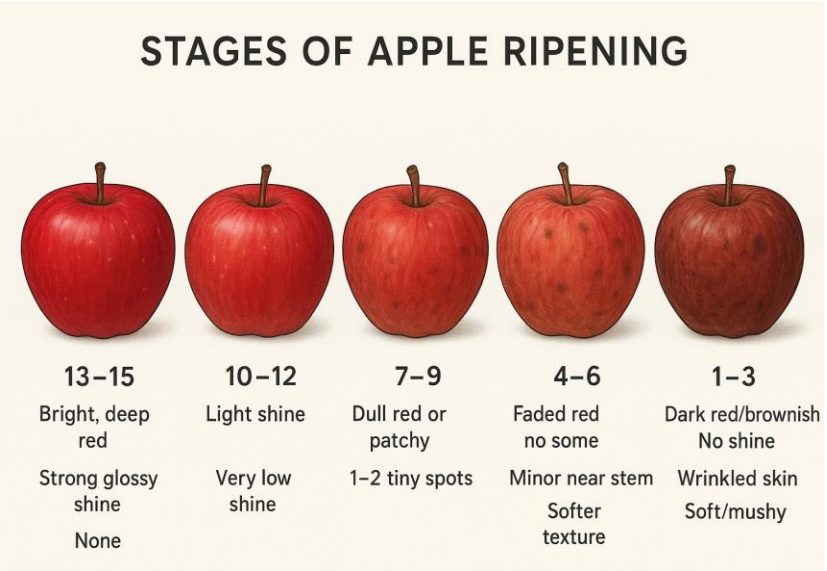
What i did?

- Trained 6 AI models with custom datasets for precise food expiry prediction
- Built custom expiry logic from user surveys, science facts, and smart rules
- OCR scanning for track expiry dates on labels foods
- Created inventory tracking with automated expiry alerts

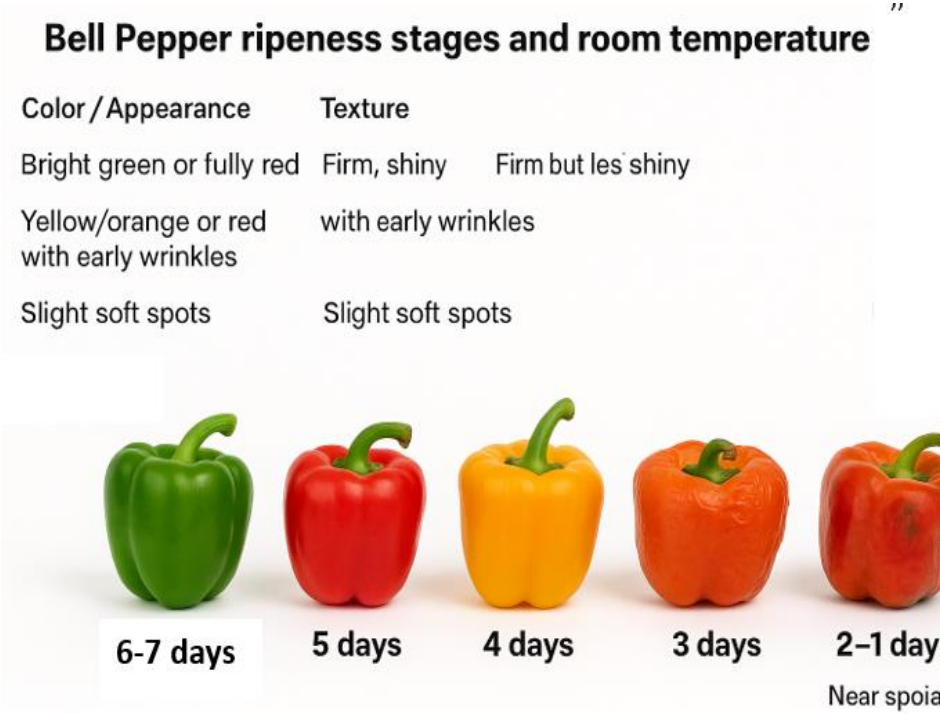
1. Dataset creation & Models training

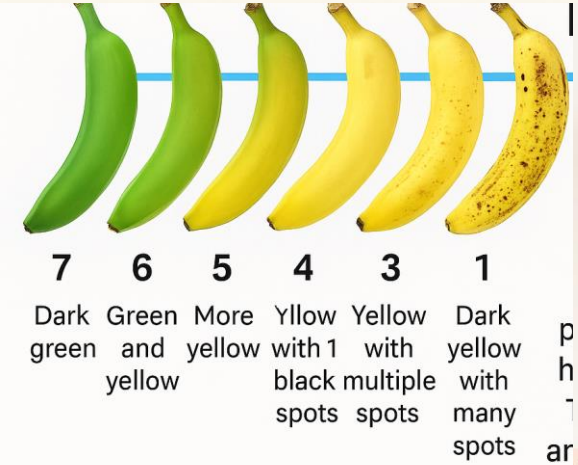
FreshiFy

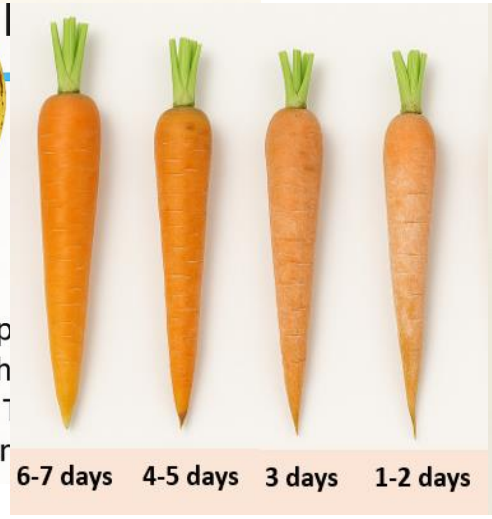
Dataset creation

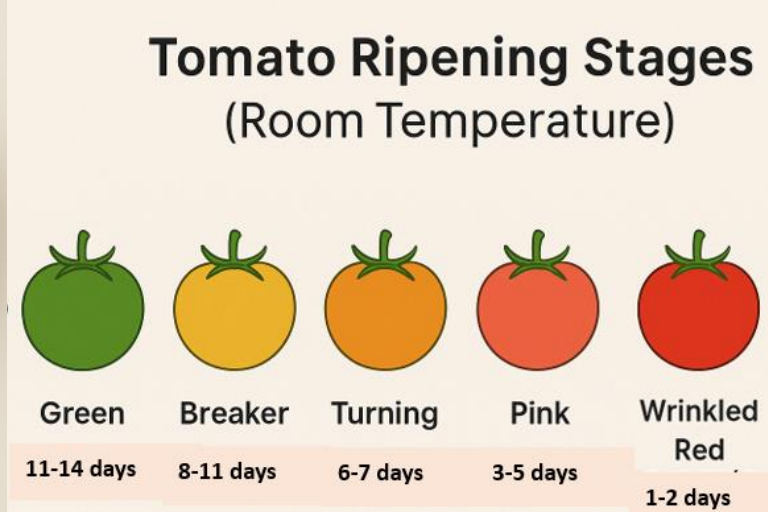


Color: colour changes for timeline.
Shine/Gloss: Monitored loss of surface gloss as a spoilage indicator.
Spots/Bruises: Included images with visible blemishes or damage.
Wrinkles: Captured texture changes like wrinkling or shriveling.







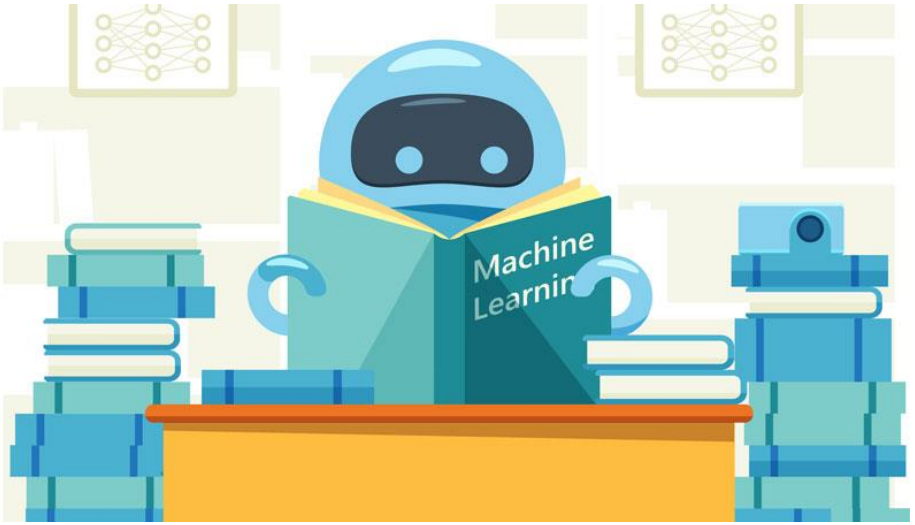


Models creation

- **ML Models Training:**

Used transfer learning with EfficientNet and MobileNet architectures . Implemented data augmentation (rotation, flip, zoom), early stopping, and learning rate scheduling in TensorFlow/Keras. Trained separately for each fruit to handle unique spoilage patterns.**Accuracies**:

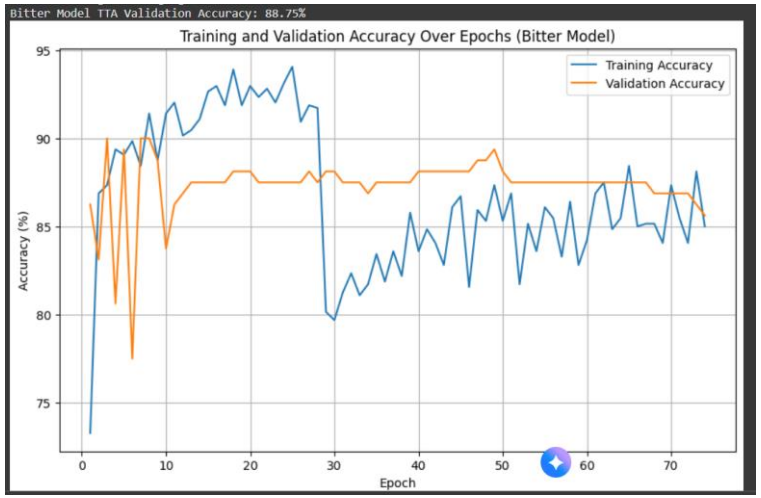
- Apple: 72% (EfficientNetB1)
- Bitter Gourd: 89% (MobileNetV2)
- Carrots: 72% (EfficientNetB0)
- Tomatoes: 85% (EfficientNetB3)
- Bell Pepper: 94% (EfficientNetB0)
- Banana: 72% (MobileNetV2)



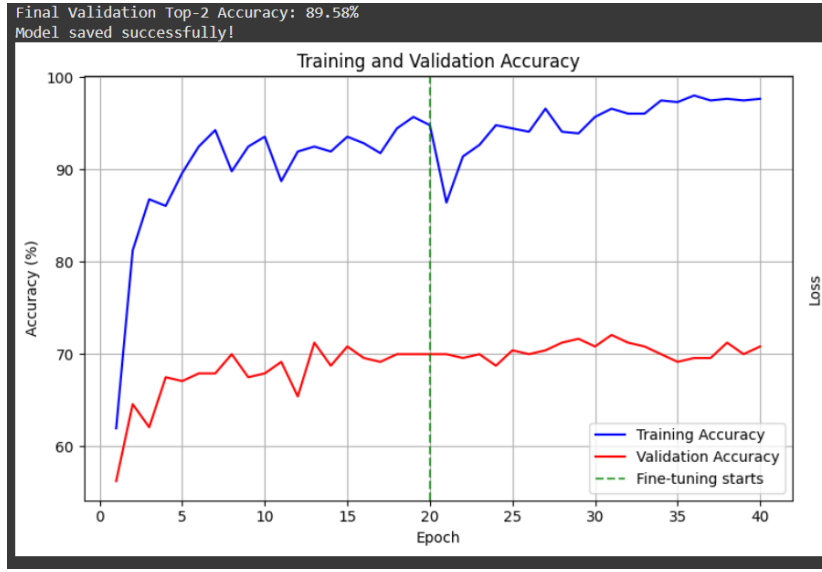
- **Why six Models Instead of one ?**

Each fruit exhibits unique spoilage signs (bananas brown, tomatoes soften), requiring tailored feature detection. Separate models for each fruit optimize accuracy by focusing on specific visual cues, while a single model would struggle to adapt to these diverse patterns, leading to lower overall precision and inconsistent predictions.

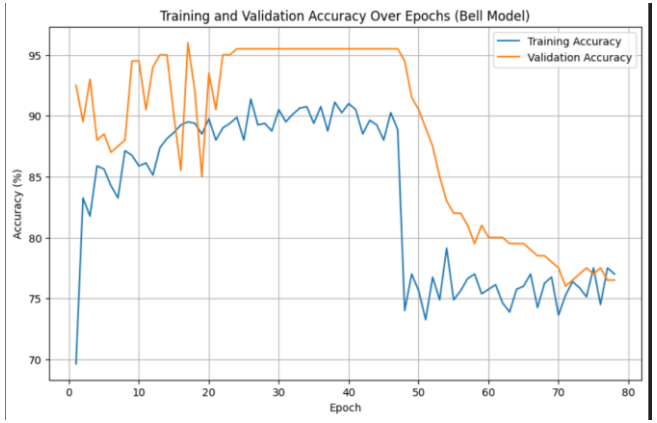
```
Best Validation Accuracy Achieved: 72.08%

Detailed Performance Analysis:
----------------------------------------
15/15 ━━━━━━━━━━━━━━━━━━━━ 19s 1s/step

Classification Report:
                  precision    recall  f1-score   support

carrots_1-2 days       0.59      0.73      0.65        60
  carrots_3 days       0.58      0.37      0.45        60
carrots_4-5 days       0.79      0.97      0.87        60
carrots_6-7 days       0.91      0.82      0.86        60

        accuracy                           0.72       240
       macro avg       0.72      0.72      0.71       240
    weighted avg       0.72      0.72      0.71       240
```

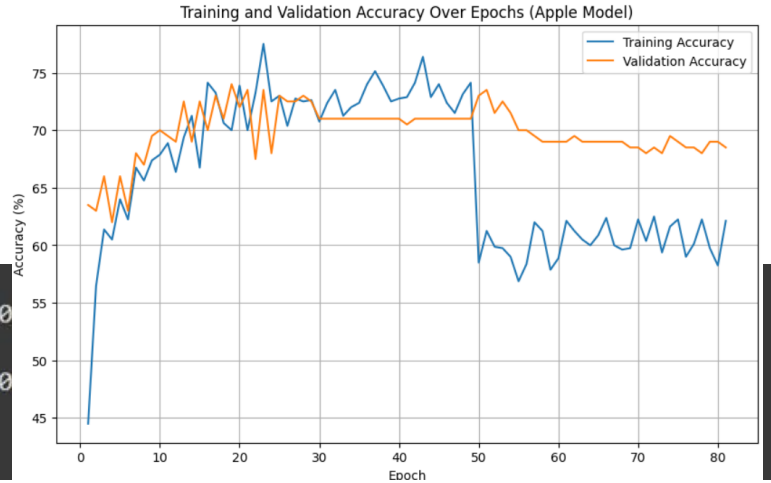


FreshiFy




```
# Load and evaluate the model
model = tf.keras.models.load_model('best_efficientnet_apple_model.keras')
val_loss, val_acc = model.evaluate(val_generator)
print(f"Apple Model Validation Accuracy: {val_acc * 100:.2f}%")

7/7 ━━━━━━━━━━━━━━━━━━━━ 24s 2s/step - accuracy: 0.7805 - lo
Apple Model Validation Accuracy: 72.00%
```






```
Epoch 30/50
25/25 ━━━━━━━━━━━━━━━━━━━━ 188s 6s/step - accuracy: 0.7605 - loss: 0.7736 - val_accuracy: 0.7650
Epoch 31/50
25/25 ━━━━━━━━━━━━━━━━━━━━ 200s 6s/step - accuracy: 0.7702 - loss: 0.7084 - val_accuracy: 0.7650
7/7 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.9706 - loss: 0.1397
Bell Model Validation Accuracy: 94.50%
```

- **Real - time estimation with sensor inputs :**

1. Fetch Latest Sensor Data

Pulls current temperature,humidity, and ethylene from MongoDB's SensorReadings table.

2. Apply Food-Specific Science

Uses calculate_enhanced_factors() with simple science rules:

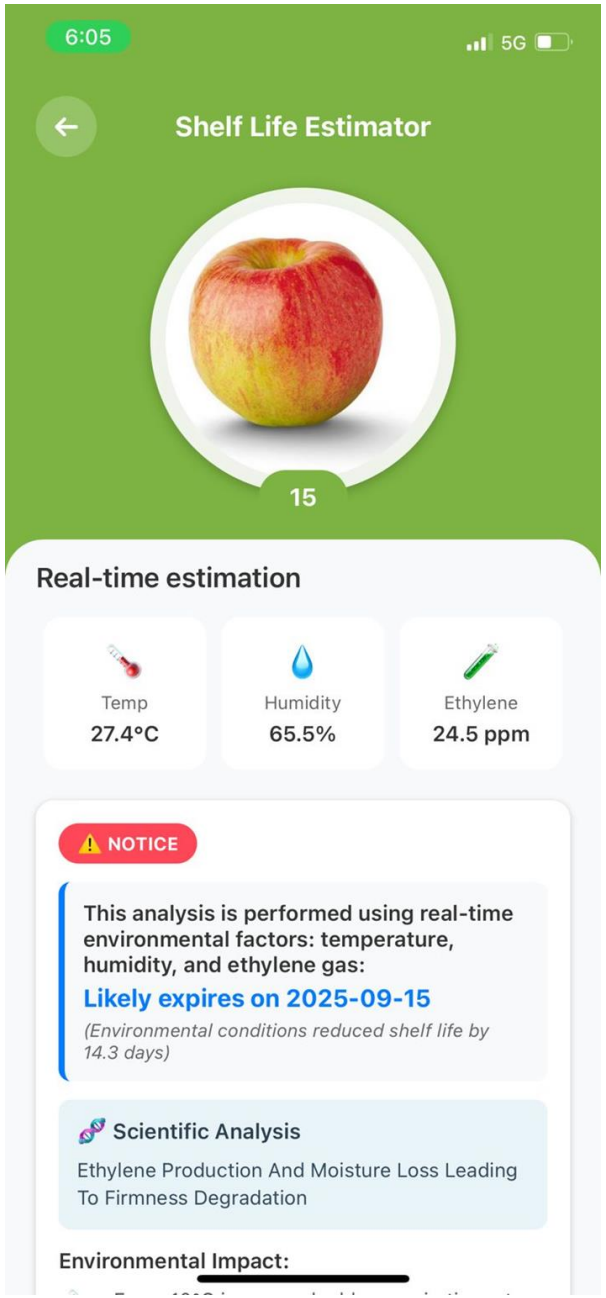
- Temperature: Checks if temp (e.g., 20°C) is above optimal (e.g., 4°C for apples). Every 10°C above doubles spoilage (16°C → ~2x faster, Factor ~0.5).

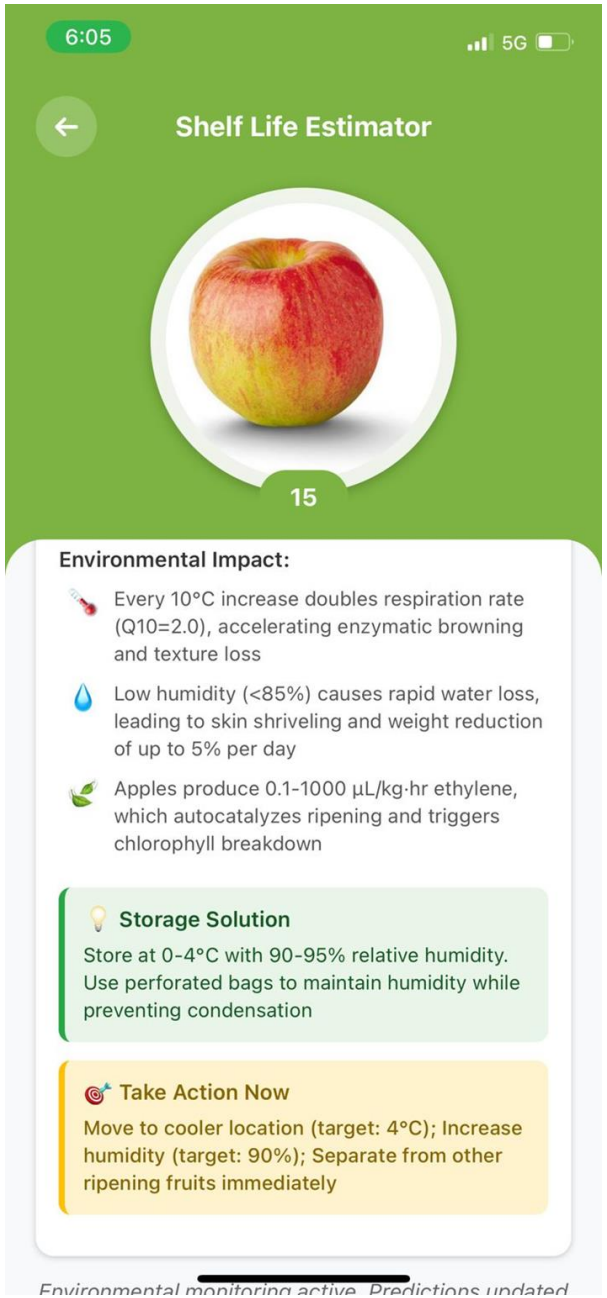
- Humidity: Compares to optimal (e.g., 90% for apples). A 30% drop (e.g., 60%) shortens life by sensitivity (0.4 × 30% = 12%, Factor ~0.7).

- Ethylene: If above threshold (e.g., 0.1 ppm for apples), extra gas (e.g., 0.4 ppm) speeds ripening by sensitivity (0.9 × 0.4 = 36%, Factor ~0.6).

2. Adjust Shelf Life

$$\text{Adjusted Days} = \text{Initial Days} \times (\text{Temp Factor} \times \text{Humidity Factor} \times \text{Ethylene Factor})$$

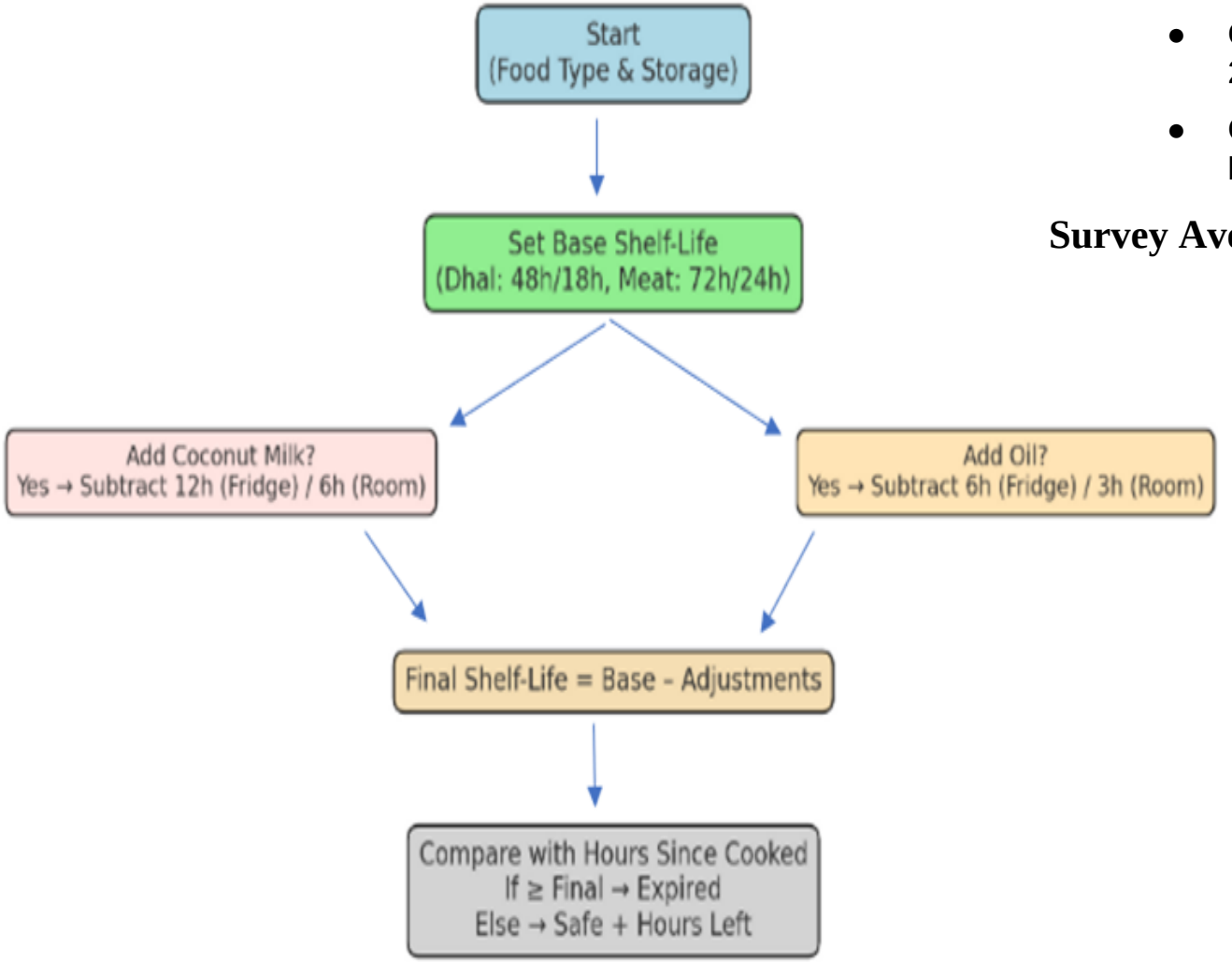




FreshiFy

- **Warning Note Purpose**: Provides users with a guide, not a final decision, showing why environmental factors affect food (e.g., high temp speeds spoilage) and offers storage solutions (e.g., "Store at 4°C").
- **Limitation**: Environmental factors (temp, humidity, ethylene) change minute by minute, so the prediction is a snapshot—trust it as a helpful hint, not a lock-in for the next minute.
- **Scientific Basis**: Based on food science (e.g., apples spoil 2x faster above 14°C) with practical fixes like optimal storage conditions.

2. Custom expiry logic for HomeMade from user surveys, science facts, and smart rules



USDA / FDA + WHO food safety guidelines:

- Cooked Meat Dishes : safe for 3–4 days in fridge (≤4°C), but only 2 hours at room temperature (>25°C).
- Cooked Legume Dishes : safe for 2–3 days in fridge, but 4–6 hours at room temperature.

Survey Averages

- Most respondents refrigerated curries for 2–3 days (around 48–72 hours).
- Room temperature storage: majority kept dhal <12 hrs, meat <24 hrs.
- Coconut milk curries: respondents noticed faster spoilage, often within 1 day in fridge.

Base Shelf-Life

- Dhal Curry → 48 hours (Fridge), 18 hours (Room Temp)
- Meat Curry → 72 hours (Fridge), 24 hours (Room Temp)

Ingredient Adjustments:

- Coconut Milk: Decreases freshness (spoils faster).
  → Subtract 12h (Fridge) or 6h (Room Temp).
- Oil: Also speeds spoilage.
  → Subtract 6h (Fridge) or 3h (Room Temp).

Questions Responses 110 Settings

Homemade Food Shelf-Life Estimation

Target group: People aged 30+ with home cooking experience.
Purpose: To understand how ingredients and storage methods affect the safe shelf life of homemade dhal and meat curry.
ඉලක්ක කණ්ඩායම: වයස අවුරුදු 30ට වැඩි, නිවසේ ආහාර පිසීමේ පළපුරුද්දක් ඇති පුද්ගලයන්
අරමුණ: පරිප්පු සහ මස් ව්‍යාංජනවල උණුසුම් කිරීමේ ක්‍රම සහ සංග්‍රහණය කිරීම ආහාරයේ ආයු කාලය කෙසේ බලපානද තහවුරු කිරීම

SECTION A: Participant Background *

What is your age group?
(ඔබේ වයස් කාණ්ඩය කුමක්ද?)

- 30–39
- 40–49
- 50–59

FreshiFy

Purpose of the Survey

- To understand real household practices about storing and consuming homemade food.
- To compare scientific guidelines with local experience.

Why Age 30+ Group?

- People above 30 usually have more cooking experience.
- They rely on both scientific awareness and traditional knowledge passed down from families.
- Their responses reflect practical, real-life food safety decisions

3. OCR scanning for labels products

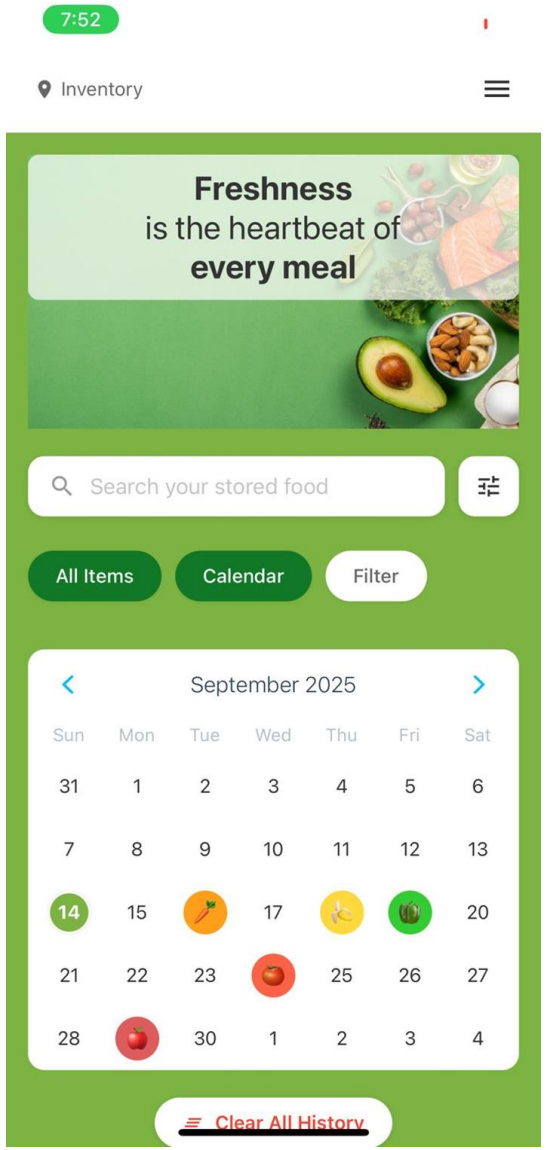
Integrated Tesseract OCR library with image preprocessing (contrast enhancement, sharpening, multiple PSM configurations) to extract dates from labels in formats like DD/MM/YYYY or "Best by DD/MM/YYYY".

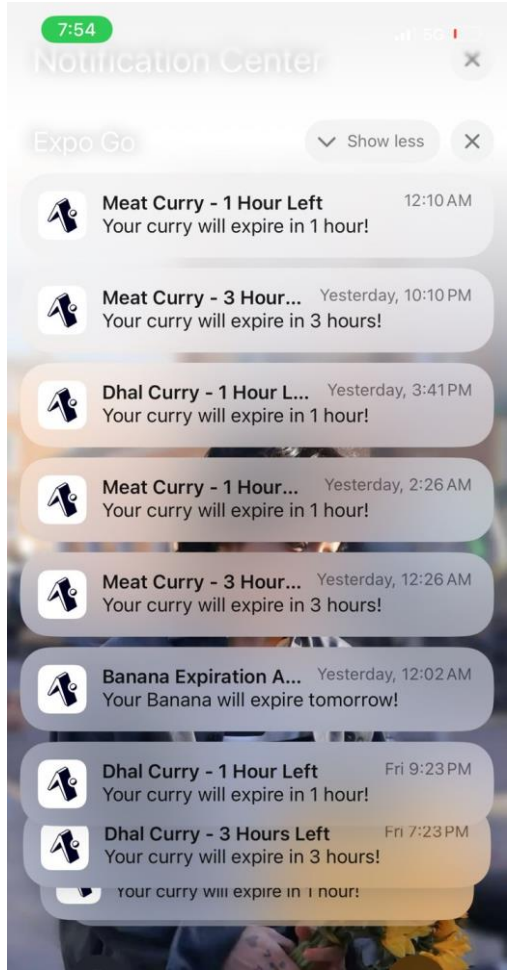
- **Process**: Crops image regions, applies adaptive thresholding, runs OCR with whitelists for numbers/dates, validates extracted dates (e.g., reasonable future dates), and adjusts expiry based on sensor data for real-time accuracy.

- **Purpose**: This feature helps notify users about expiration dates to use food before it spoils or replace it, aligning with our main goal to minimize food waste by preventing unnoticed spoilage.

4. Inventory tracking with automated expiry alerts

FreshiFy





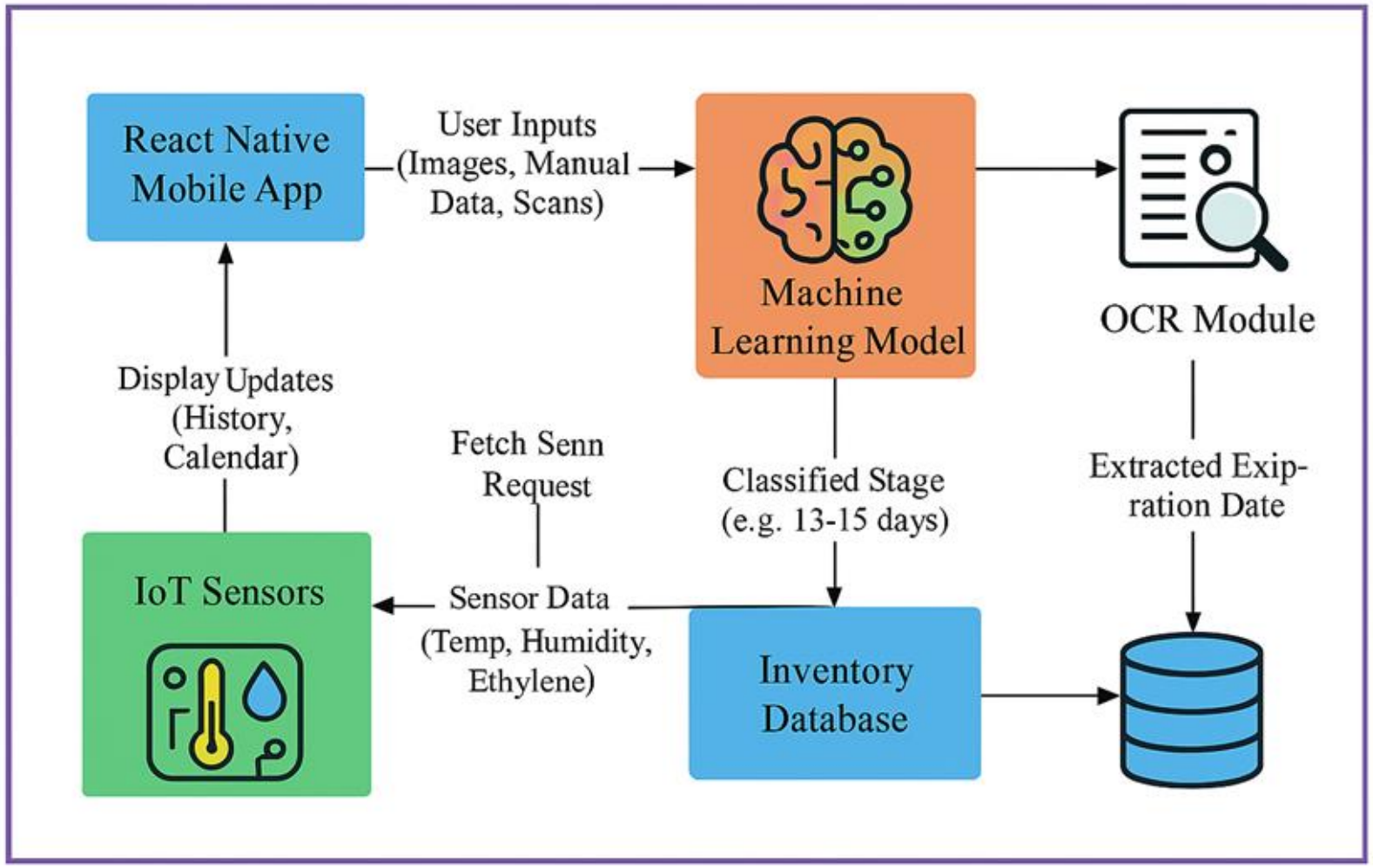
Implementation: Developed a system to store predictions in a MongoDB predictions collection, tracking expiry dates and stages for all scanned items.

Process: Uses /history endpoint to fetch and display real-time inventory, calculating remaining days/hours, and triggers notifications via notifyOneDayBefore or notifyThreeHoursBefore based on proximity to expiry.

Benefits: Provides a centralized view of all food items, with alerts (e.g., "Expires tomorrow!") to prompt action, enhancing user awareness.

Purpose: Designed to minimize food waste by reminding users to consume or replace items before they spoil, ensuring timely decisions to reduce waste.

System diagram



References

- Saltveit, M. E. (1999). Effect of Ethylene on Quality of Fresh Fruits and Vegetables. Postharvest Biology and Technology, 15(3), 279-292.
- Watkins, C. B. (2003). Principles and Practices of Apple Storage Management. New York Fruit Quarterly, 11(2), 15-20.
- Turner, D. W., & Fortescue, J. A. (2007). Banana Physiology: Postharvest Handling and Storage. Acta Horticulturae, 740, 499-506.
- USDA “Cold Food Storage Chart” & FDA refrigerator safety guides.
- USDA “Leftovers and Food Safety” guidance.
- Study: “Evaluation of the changes in physiochemical properties and fatty acid profile of industrially pasteurized coconut (Cocos nucifera) milk during storage”
- Research: “Physico-chemical and shelf life evaluation of pasteurized coconut milk” and similar studies.

FreshiFy

Managing Food Storage Compatibility

IT19970882| Fernando M.G.S.S.A

SPECIALIZING IN INFORMATION TECHNOLOGY

Introduction

FreshiFy

This **component of our system**, designed to **optimize household food storage**.

It combines **IoT sensors, AI models, and rule-based logic** to monitor temperature, humidity, and ethylene gas in real time.

By detecting **incompatible food pairings** and predicting spoilage risks, the system delivers **smart alerts and recommendations** through a mobile app.

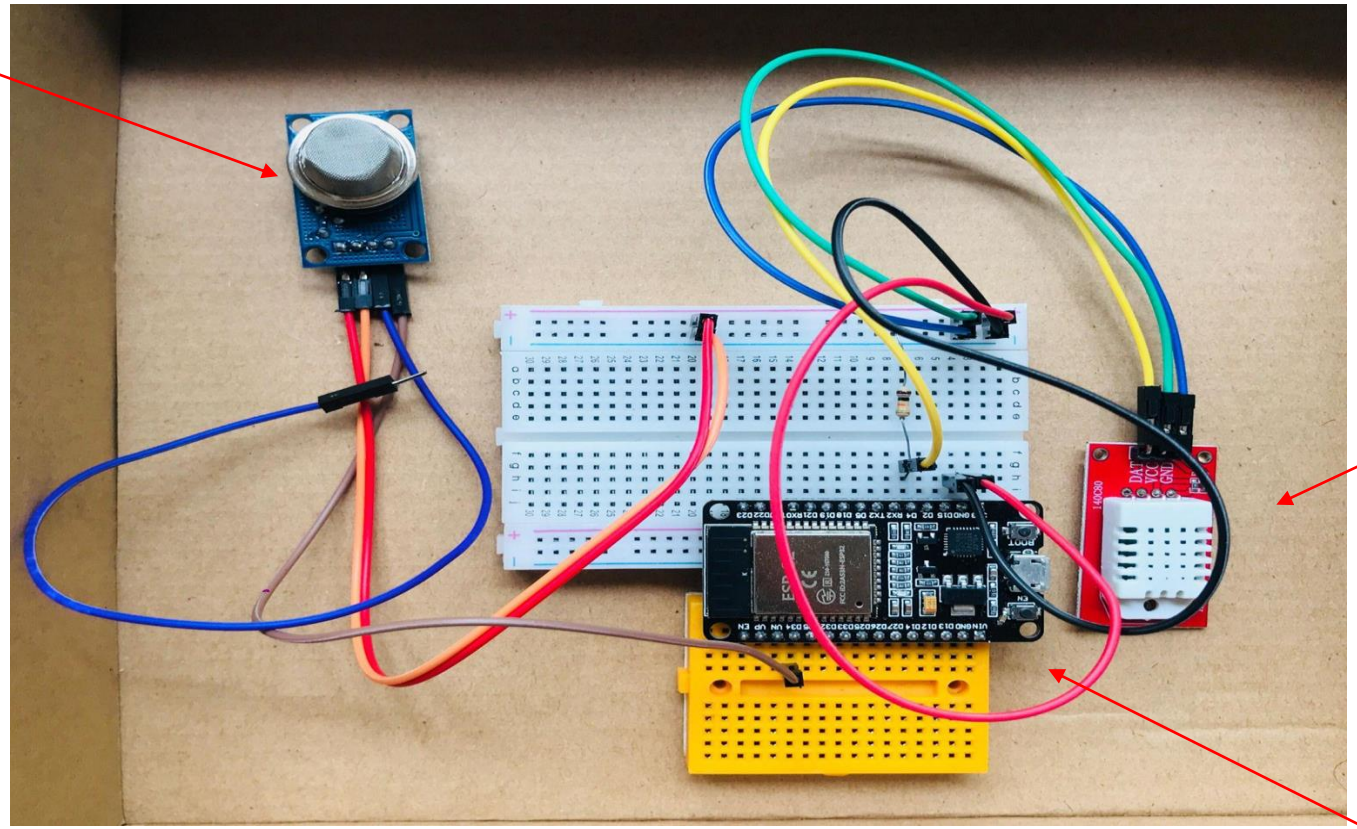
The goal is to **reduce food waste, extend freshness, and empower households** with data-driven storage decisions.

What I did so far?

- Built an **IoT sensor network** (ESP32 + DHT22 + MQ135) for real-time data collection
- Developed a **Flask backend + MongoDB database** to process and store sensor data.
- Trained and tested **ML models (Random Forest, SVM, RNN)** for spoilage prediction
- Integrated a **two-tier alert system** (Critical & Advisory) for user-friendly notifications
- Created a **React Native mobile app** for dashboards, alerts, and user input

1. Built an IoT sensor network (ESP32 + DHT22 + MQ135) for real-time data collection

MQ-135 (gas sensor)



DHT22 (temp & humidity sensor)

ESP32 Board

2. Developed a **Flask backend + MongoDB database** to process and store sensor data.

Purpose

- Processes IoT readings (temperature, humidity, ethylene), stores them reliably, and serves real-time predictions, history, and alerts to the mobile app.

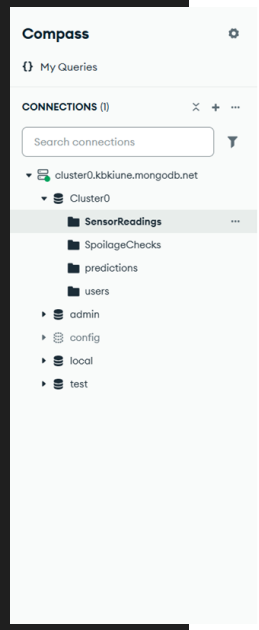
Framework & Database

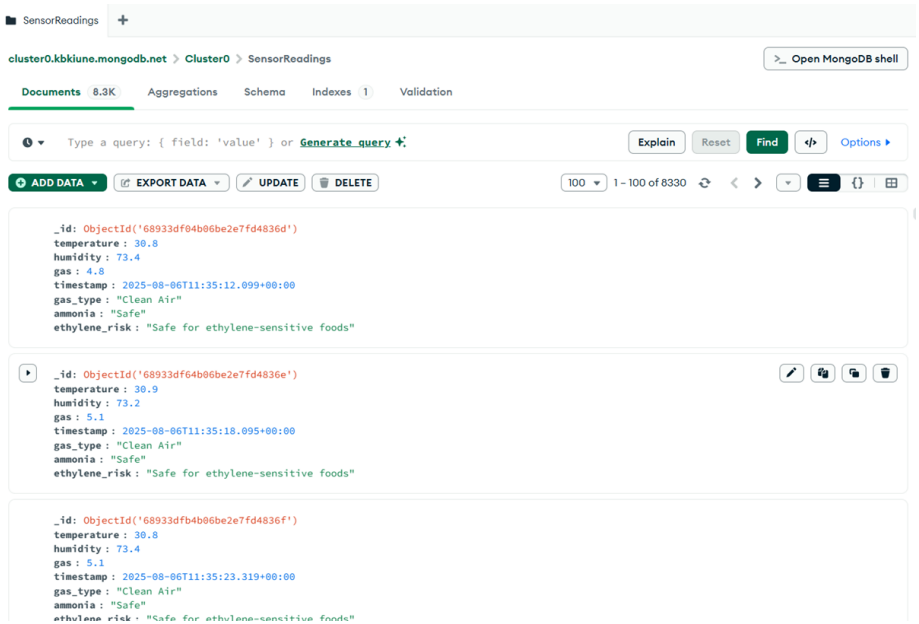
- **Backend:** Flask REST API
- **Database:** MongoDB Atlas (cloud-based storage for sensor logs + predictions)

app.py > combined_risk_from_latest

```python
from flask import Flask, request, jsonify
from pymongo import MongoClient
from datetime import datetime
import os
import pickle
import numpy as np
from keras.models import load_model
from dotenv import load_dotenv
from flask_cors import CORS
from difflib import get_close_matches
from datetime import datetime, timezone
from dateutil import parser as dtparser

# === Load environment variables from .env ===
load_dotenv()

# === Load ML Models ===
print("\U0001F4E6 Loading models...")
try:
    rf_model = pickle.load(open("rf_model.pkl", "rb"))
    svm_model = pickle.load(open("svm_model.pkl", "rb"))
    scaler = pickle.load(open("scaler.pkl", "rb"))
    rnn_model = load_model("rnn_model.h5")
    print("✅ All models loaded.")
except Exception as e:
    print("❌ Error loading models:", e)
    exit()

# === Flask App ===
app = Flask(__name__)
CORS(app)

# === MongoDB Setup ===
MONGO_URI = os.environ.get("MONGO_URI")
try:
    client = MongoClient(MONGO_URI)
    db = client["Cluster0"]
```

```python
@app.route("/")
def home():
    return "✅ FreshiFy API Running!"

@app.route("/upload", methods=["POST"])
def upload():
    try:
        data = request.get_json()
        data["timestamp"] = datetime.now(timezone.utc)
        interpretation = interpret_air_quality(data.get("gas", 0))
        data.update(interpretation)

        if collection is not None:
            collection.insert_one(data)

        return jsonify({"status": "saved"}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route("/latest", methods=["GET"])
def get_latest_data():
    try:
        latest = collection.find_one(sort=[("timestamp", -1)])
        if not latest:
            return {"error": "No sensor data found"}, 404

        temp = float(latest.get("temperature", 0))
        hum  = float(latest.get("humidity", 0))
        eth  = float(latest.get("gas", 0))

        combo = combined_risk_from_latest(temp, hum, eth)

        return {
            "temperature": temp,
            "humidity": hum,
            "ethylene": eth,
            "gas_type": latest.get("gas_type", "-"),
            "ammonia": latest.get("ammonia", "-"),
            "ethylene_risk": latest.get("ethylene_risk", "-"),
            "timestamp": ts_ms(latest.get("timestamp")),
            "combined_risk": combo["combined_risk"],
            "combined_score": combo["combined_score"],
            "combined_notes": combo["combined_notes"],
        }
    except Exception as e:
        return {"error": str(e)}, 500


@app.route("/alerts", methods=["GET"])
def get_alerts():
    try:
        latest = collection.find_one(sort=[("timestamp", -1)])
```





3. Trained and tested ML models (Random Forest, SVM, RNN) for spoilage prediction

FreshiFy

```python
rf = RandomForestClassifier(random_state=42)

# Train the Random Forest classifier
rf.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred = rf.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Random Forest Accuracy:", accuracy)
print("\nRandom Forest Classification Report:")
print(classification_report(y_test, y_pred))
```

[16]

```
Random Forest Accuracy: 1.0

Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        31
           1       1.00      1.00      1.00        26

    accuracy                           1.00        57
   macro avg       1.00      1.00      1.00        57
weighted avg       1.00      1.00      1.00        57
```

```python
import pickle
from sklearn.preprocessing import StandardScaler

# Save the rf model as a pickle file
with open('rf_model.pkl', 'wb') as f:
    pickle.dump(rf, f)

print("rf model saved as 'rf_model.pkl'.")
```

[17]

```
rf model saved as 'rf_model.pkl'.
```

```python
import pickle

# Load the rf model
with open('rf_model.pkl', 'rb') as f:
    rf_loaded = pickle.load(f)

print("rf model loaded successfully.")
```

[18]

```
rf model loaded successfully.
```

SVM

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
import joblib
import pickle
import h5py
import numpy as np


# Use the balanced DataFrame from the SMOTE step
X_bal = df[['Temperature', 'Humidity', 'Ethylene Gas']]
y_bal = df['Label']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_bal, y_bal, test_size=0.3, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize the SVM classifier
svm = SVC(random_state=42)

# Train the SVM classifier
svm.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred = svm.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Serialize the SVM model using pickle
model_bytes = pickle.dumps(svm)
```

[11]

```
Accuracy: 0.9649122807017544

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97        31
           1       0.93      1.00      0.96        26
```

RNN

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping

# Assume 'df' is your balanced DataFrame from the SMOTE step
X_bal = df[['Temperature', 'Humidity', 'Ethylene Gas']]
y_bal = df['Label']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_bal, y_bal, test_size=0.3, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# For RNNs we need a 3D input: (samples, timesteps, features)
# Here, we treat each sample as a sequence with 1 timestep.
X_train_scaled = X_train_scaled.reshape((X_train_scaled.shape[0], 1, X_train_scaled.shape[1]))
X_test_scaled = X_test_scaled.reshape((X_test_scaled.shape[0], 1, X_test_scaled.shape[1]))
```

[20]

```python
# Build the RNN model with an LSTM layer
rnn_model = Sequential([
    LSTM(50, activation='tanh', input_shape=(X_train_scaled.shape[1], X_train_scaled.shape[2])),
    Dense(1, activation='sigmoid')  # Binary classification
])

# Compile the model
rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Set an early stopping callback to avoid overfitting
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model
history = rnn_model.fit(X_train_scaled, y_train,
                        validation_split=0.2,
                        epochs=50,
                        batch_size=16,
                        callbacks=[early_stop],
                        verbose=1)
```

[21]

FreshiFy

4. Integrated a **two-tier alert system** (Critical & Advisory) for user-friendly notifications

```python
# === Helper Logic ===
def interpret_air_quality(ethylene_ppm: float) -> dict:
    e = float(ethylene_ppm or 0)

    if e < 0.10:
        level = "Safe"
        note  = "Background ethylene — OK for all items."
    elif e < 0.50:
        level = "Low"
        note  = "Sensitive leafy greens may age a bit faster over days."
    elif e < 1.00:
        level = "Moderate"
        note  = "Shorter shelf life for ethylene-sensitive produce."
    elif e < 5.00:
        level = "High"
        note  = "Rapid deterioration for sensitive veg; speeds fruit ripening."
    else:
        level = "Very High"
        note  = "Unsafe for storage; separate fruits and greens immediately."

    return {
        "gas_type": "Ethylene (estimated)",
        "ethylene_risk": level,      # <- show this in your UI pill
        "ethylene_note": note,       # <- optional helper text
    }

def combined_risk_from_latest(temp: float, hum: float, eth: float) -> dict:
    # Use the generic path in spoilage_score_for (no specific food),
    # which uses ACTIVE_PROFILE temp band, 60-80% humidity, and neutral ethylene.
    rb = spoilage_score_for("__generic__", temp, hum, eth)
    return {"combined_risk": rb["band"], "combined_score": rb["score"], "combined_not
```

```python
# Temperature vs active profile
tset = PROFILE_SETPOINT.get(ACTIVE_PROFILE, 4.0)
tmin, tmax = tset - 2, tset + 2
if temp < tmin:
    alerts.append({
        "message": f"Temperature {temp:.1f}°C is below {tmin}°C — risk of chilling injury fo
        "priority": "low", "action": "Raise setpoint slightly or move sensitive items to pro
    })
elif temp > tmax and temp <= tmax + 4:
    alerts.append({
        "message": f"Temperature {temp:.1f}°C is above {tmax}°C — food safety window narrowi
        "priority": "high", "action": "Check door seals, reduce door-open time, set a colder
    })
elif temp > tmax + 4:
    alerts.append({
        "message": f"Temperature {temp:.1f}°C is dangerously high — perishable items may spo
        "priority": "high", "action": "Move perishables to colder storage immediately and in
    })

# Humidity band for general fridge (advisory if way off)
if hum < 60:
    alerts.append({
        "message": f"Humidity {hum:.0f}% is low — leafy greens may wilt.",
        "priority": "low", "action": "Use high-humidity drawer or add covered container to raise RH.", "time": "Now"
    })
elif hum > 90:
    alerts.append({
        "message": f"Humidity {hum:.0f}% is very high — mold risk increases.",
        "priority": "low", "action": "Improve airflow; keep items dry and unstacked.", "time": "Now"
    })
```

```python
FOOD_PROFILES = {
    # Climacteric (ethylene-producing) fruits
    "apple":   {"tmin": 0,  "tmax": 4,  "hmin": 90, "hmax": 95, "ethylene": "producer", "baseline_days": 30, "zone": "produce_drawer"},
    "banana":  {"tmin": 12, "tmax": 14, "hmin": 85, "hmax": 95, "ethylene": "producer", "baseline_days": 5,  "zone": "cool_room"},
    "mango":   {"tmin": 10, "tmax": 13, "hmin": 85, "hmax": 90, "ethylene": "producer", "baseline_days": 5,  "zone": "cool_room"},
    "avocado": {"tmin": 5,  "tmax": 7,  "hmin": 85, "hmax": 90, "ethylene": "producer", "baseline_days": 5,  "zone": "fridge"},
    "pear":    {"tmin": -1, "tmax": 1,  "hmin": 90, "hmax": 95, "ethylene": "producer", "baseline_days": 20, "zone": "produce_drawer"},
    "orange":  {"tmin": 3,  "tmax": 7,  "hmin": 85, "hmax": 90, "ethylene": "neutral",  "baseline_days": 14, "zone": "fridge"},

    # Ethylene-sensitive veg
    "lettuce": {"tmin": 0,  "tmax": 2,  "hmin": 95, "hmax": 100,"ethylene": "sensitive","baseline_days": 7,  "zone": "produce_drawer"},
    "spinach": {"tmin": 0,  "tmax": 2,  "hmin": 95, "hmax": 100,"ethylene": "sensitive","baseline_days": 5,  "zone": "produce_drawer"},
    "broccoli":{"tmin": 0,  "tmax": 2,  "hmin": 95, "hmax": 100,"ethylene": "sensitive","baseline_days": 7,  "zone": "produce_drawer"},
    "kale":    {"tmin": 0,  "tmax": 2,  "hmin": 95, "hmax": 100,"ethylene": "sensitive","baseline_days": 7,  "zone": "produce_drawer"},
    "carrot":  {"tmin": 0,  "tmax": 2,  "hmin": 95, "hmax": 100,"ethylene": "sensitive","baseline_days": 20, "zone": "produce_drawer"},

    # Tomatoes (special case—prefer cool room once ripened)
    "tomato":  {"tmin": 12, "tmax": 20, "hmin": 85, "hmax": 95, "ethylene": "producer", "baseline_days": 5,  "zone": "cool_room"},

    # Cooked foods
    "dhal curry": {"tmin": 0, "tmax": 4, "hmin": 0, "hmax": 100, "ethylene": "neutral", "baseline_days": 3, "zone": "fridge"},
    "meat curry": {"tmin": 0, "tmax": 4, "hmin": 0, "hmax": 100, "ethylene": "neutral", "baseline_days": 3, "zone": "fridge"},
}

KNOWN_FOODS = list(FOOD_PROFILES.keys())

# Extra human-friendly guidance per item
FOOD_TIPS = {
    "apple": {
        "aka": ["apples"],
        "shelf_life": "1–2 months in the fridge drawer; 5–7 days at room temp once fully ripe.",
        "dos": [
            "Store in the high-humidity produce drawer (2–4°C).",
            "Keep in a breathable bag; rinse before eating, not before storage.",
        ],
        "donts": [
            "Don't store next to leafy greens (apples emit ethylene).",
            "Don't wash before storage (adds surface moisture → mold).",
        ],
        "use_soon": "Use within 1–2 days if kept warm to slow further ripening."
    },
```

4. Created a **React Native mobile app** for dashboards, alerts, and user input

Framework

- Built with **React Native (Expo)** for cross-platform support (Android & iOS).

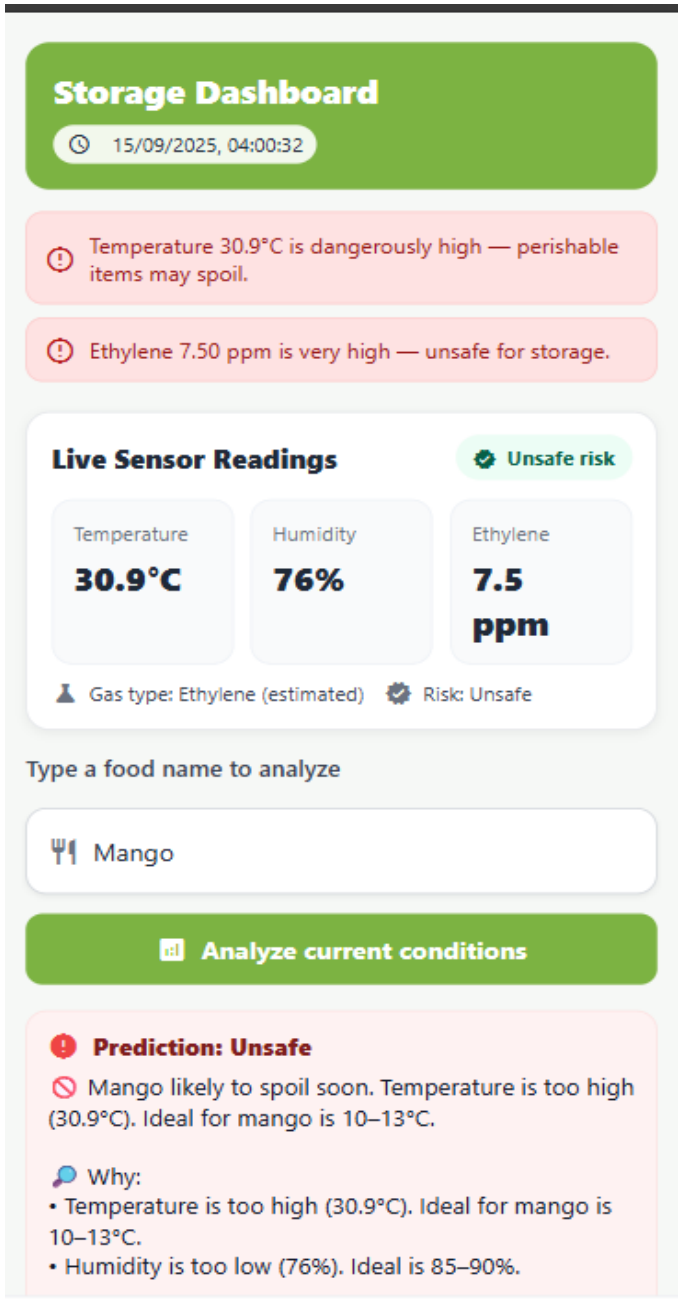
Key Features

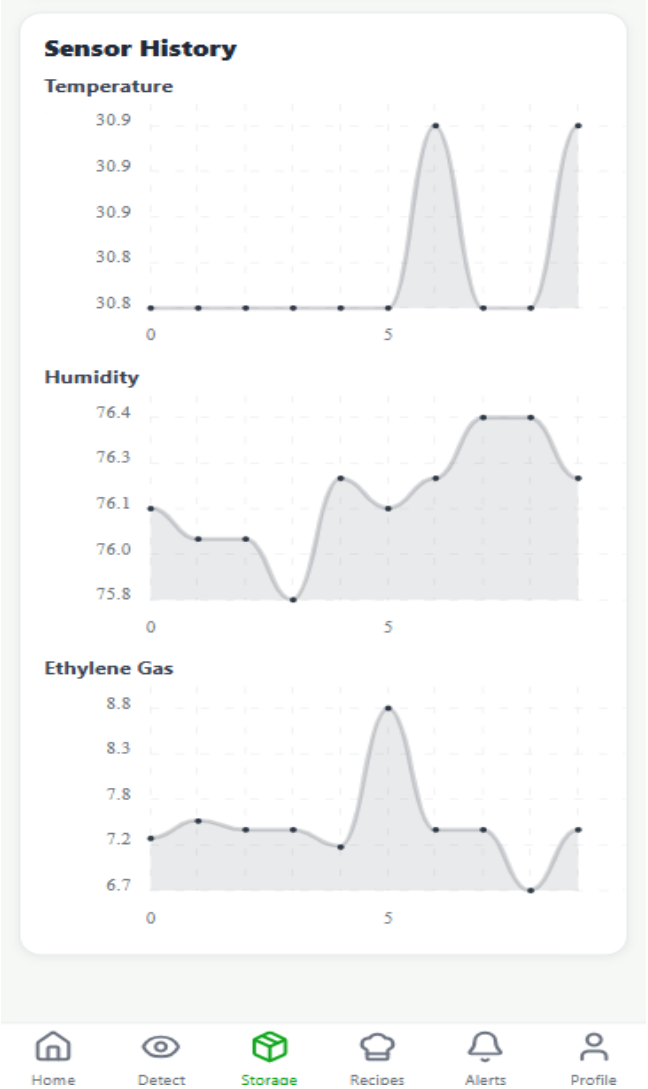
- **Real-time Dashboard** → Displays live sensor readings (Temperature, Humidity, Ethylene).
- **Spoilage Prediction Panel** → User enters/selects food → system shows prediction + recommendation.
- **History View** → Charts of past readings & spoilage predictions for trend analysis.
- **Alerts & Notifications** →
  - **Critical:** Ethylene > 0.5 ppm OR Temp deviation > ±2°C (shown in Red).
  - **Advisory:** Humidity outside 60–80% OR nearing expiry (shown in Yellow).

FreshiFy

Usability Testing Results

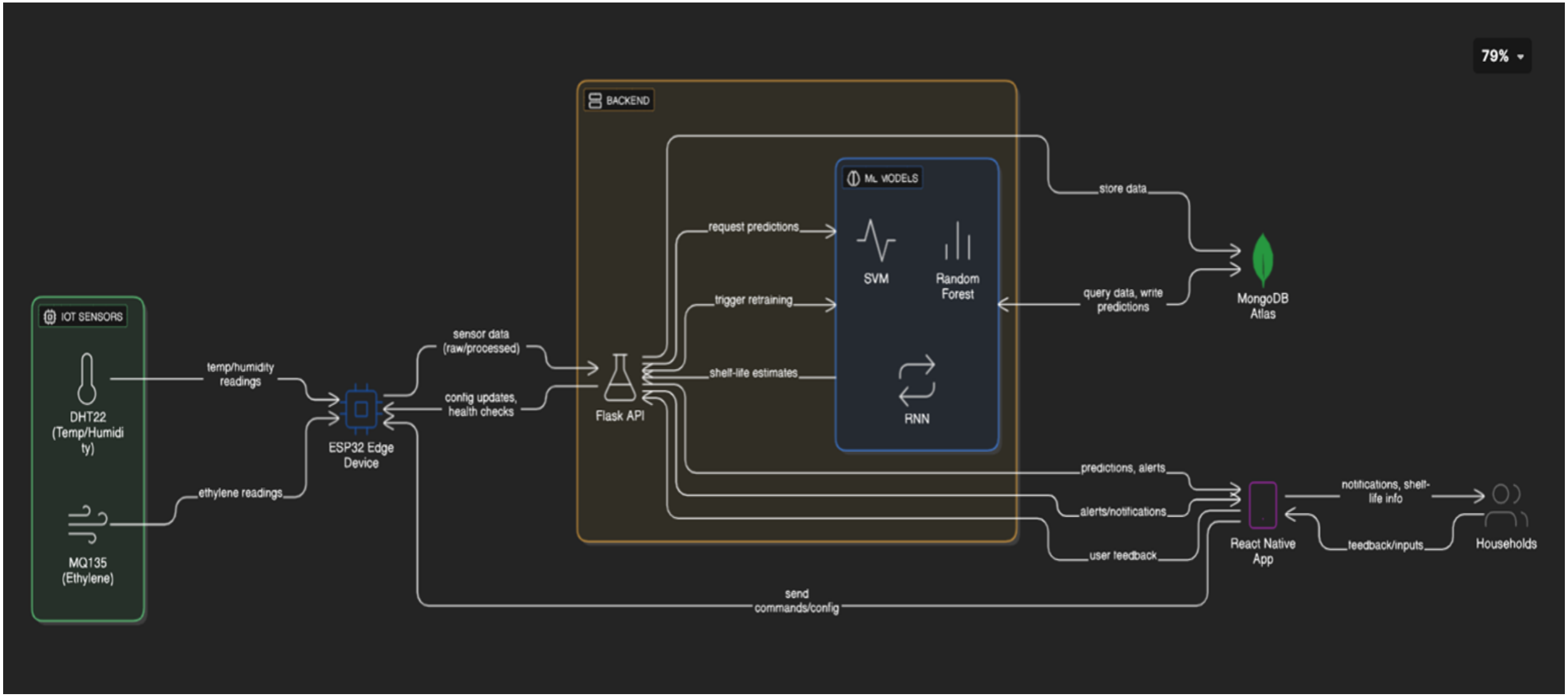
- Dashboard updates every **3–5 seconds**.
- Prediction panel responds within **2 seconds**.
- Alerts clear & color-coded (Red = Critical, Yellow = Advisory, Green = Safe).
- History view effective for long-term monitoring.





System architecture diagram

FreshiFy



Reference

- M. Z. K. S. A. A. M. M. A. A. S. R. &. M. F. A. Ebrahimi, "Novel strategies to control ethylene in fruit and vegetables for extending their shelf life: A review," *International Journal of Environmental Science and Technology,* vol. 19, no. 2022, p. 4599–4610, 2021.
- K. S. N. Bazaru Venkata Sai Nithin, "Non-Destructive Approach for Detection of Food Freshness using IoT," in *2023 International Conference on Inventive Computation Technologies (ICICT)*, Lalitpur, Nepal, 26-28 April 2023.
- C. T. a. M. Formentini, "Digital Technologies for Food Loss and Waste Prevention and Reduction in Agri-Food Supply Chains: A Systematic Literature Review and Research Agenda," *IEEE Transactions on Engineering Management,* vol. 71, no. 10, pp. 12326-12345, 2024.
- M. E. Saltveit, "Effect of ethylene on quality of fresh fruits and vegetables," *Postharvest Biology and Technology,* vol. 15, no. 3, pp. 279-292, March 1999.

FreshiFy

Real - Time Food Identification and Spoilage detection System

IT20196110 – PERERA B.C.V

SPECIALIZING IN INFORMATION TECHNOLOGY

COMPONENT -3

Introduction

Helps users identify and track food types to manage their meals more effectively.

Identifying spoiled food is crucial for preventing health risks.

Eliminates manual spoilage checks by automating the identification and notification process.

Research Problem

FreshiFy

•Current systems lack reliability in accurately identifying homemade food types.

•There are limited solutions for real-time analysis of food spoilage, combining gas sensors and image recognition to notify users proactively.

•Untimely alerts prevent users from recognizing spoilage, causing waste and risks.

Objectives

FreshiFy

Specific Objective

To develop a system that identifies food types and detects spoilage using gas sensors and image recognition, providing timely notifications to prevent health risks and minimize waste.

Sub Objective

•Identify and classify different food types.

•

•Analyze and detect the spoilage of homemade foods.

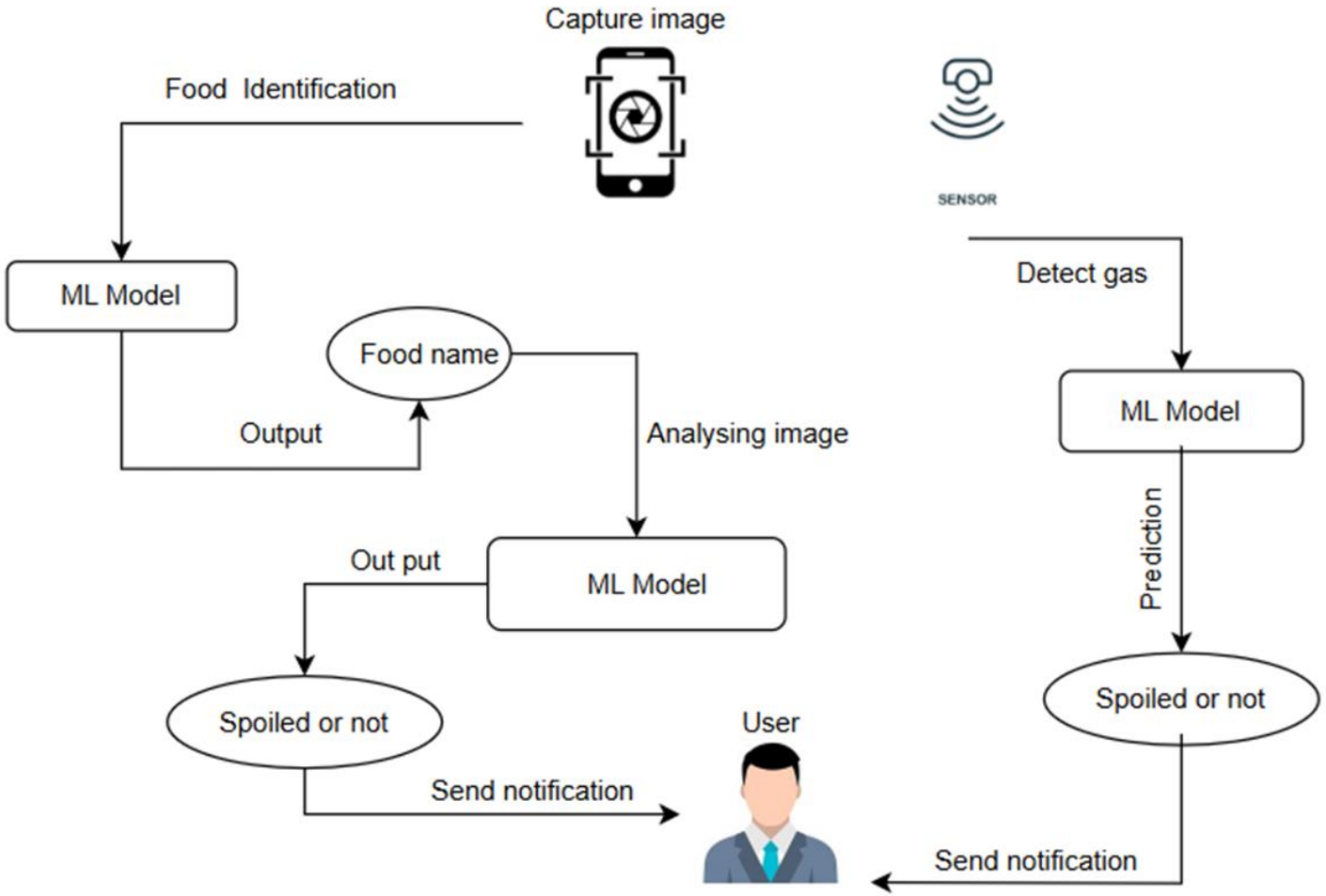
•

•Implement a notification system to inform users of spoiled food immediately.

•

•Enhance system reliability for accurate spoilage detection and timely alerts.

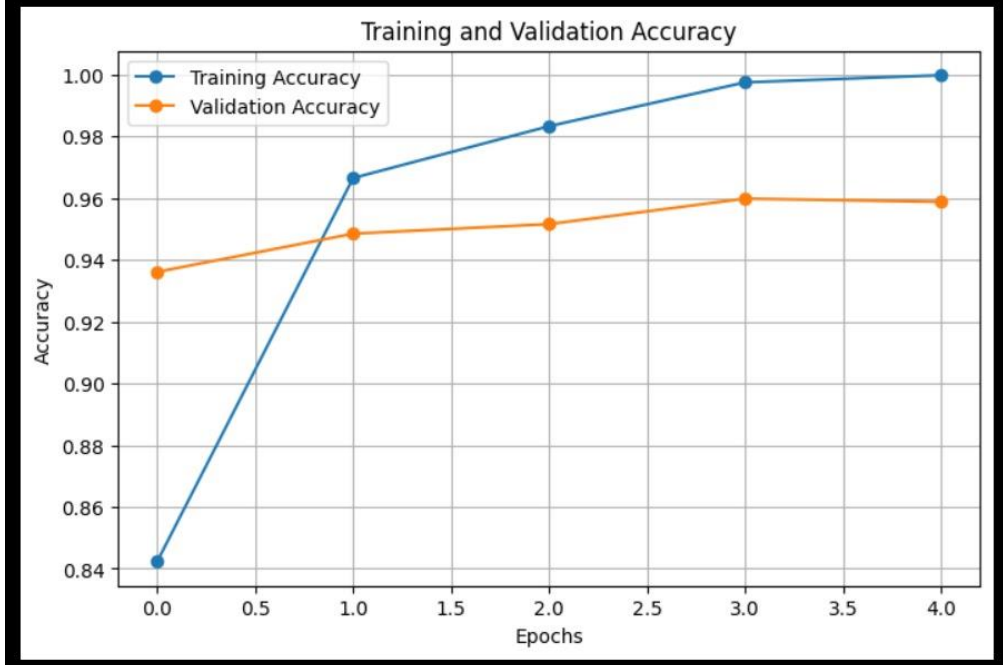
System Diagram



Methodology

Fruit classification model

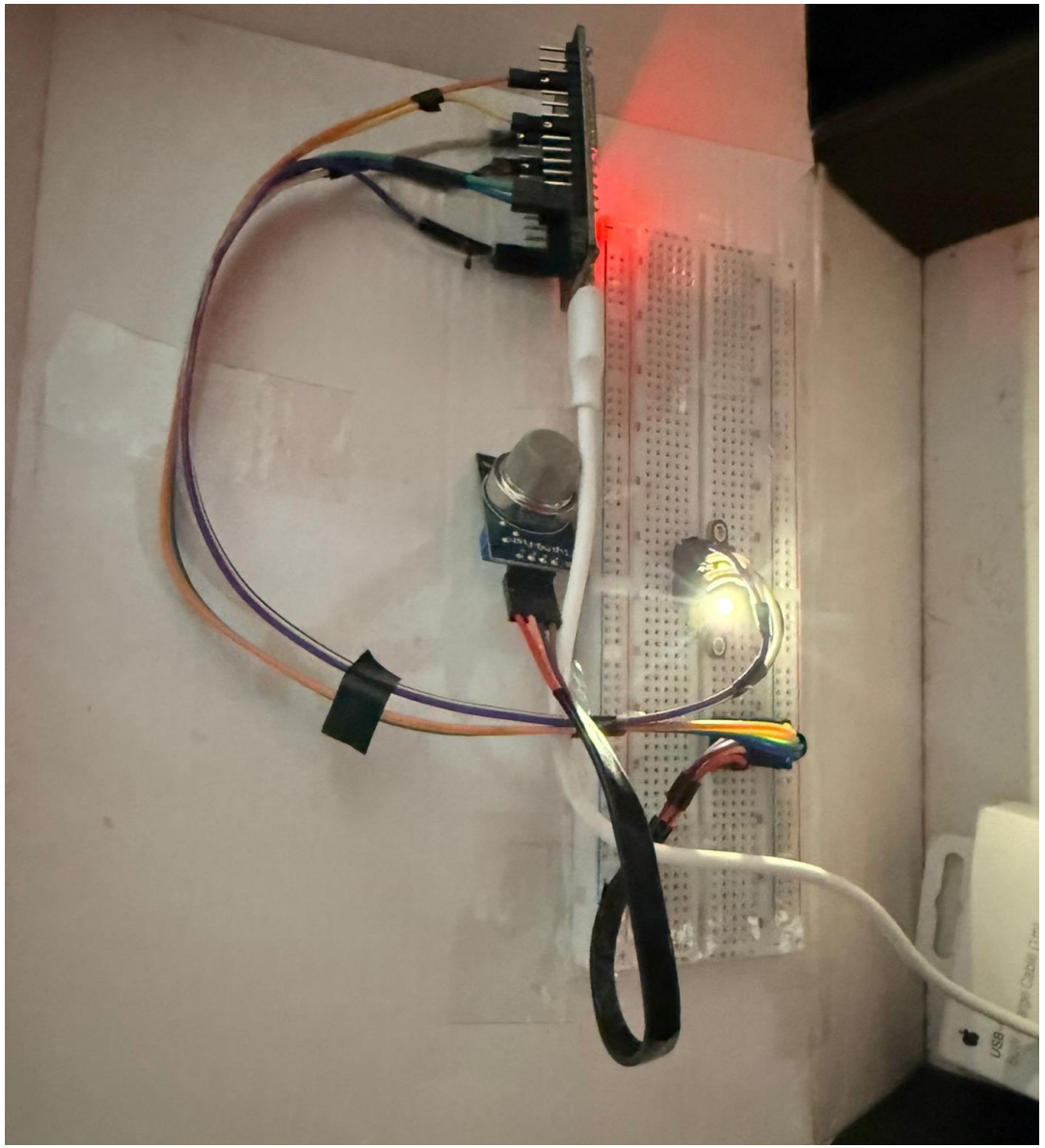
```
Model Accuracy on Validation Set: 0.9588
```




```python
# -------------------------------
# 6. Evaluate Model & Plot Accuracy Curves
# -------------------------------

# Print final accuracy
test_loss, test_acc = model.evaluate(val_dataset)
print(f"\nModel Accuracy on Validation Set: {test_acc:.4f}")

# Plot training & validation accuracy curves
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.plot(history.history['accuracy'], label='Training Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', marker='o')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.grid()
plt.show()
```

[24] Python

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 14s 432ms/step - accuracy: 0.9573 - loss: 0.1255

Model Accuracy on Validation Set: 0.9588
```



Sensor model

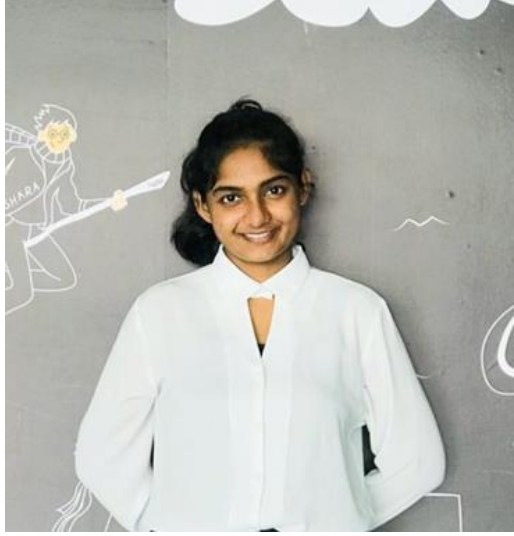
References

[1] G. C. Green, A. D. C. Chan and R. A. Goubran, "Monitoring of food spoilage with electronic nose: potential applications for smart homes," (2009) 3rd International Conference on Pervasive Computing Technologies for Healthcare, London, 2009, pp. 1-7, doi: 10.1109/PCTHEALTH.2009.5291419.

[2] P. P. Jose, N. D. Bobby, V. Ragul, P. L. Babu and P. J. J. Dinesh, "Wireless detection of cooked food decay," (2014) IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, 2014, pp. 1 5, doi: 10.1109/ICCIC.2014.7238309.

[3] S. Fang, "Design of Intelligent Detection System for Food Spoilage," (2018) 11th International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, 2018, pp. 190-194, doi: 10.1109/ICICTA.2018.00050.

[4] P. Chongthanaphisut, T. Seesaard and T. Kerdcharoen, "Monitoring of microbial canned food spoilage and contamination based on e-nose for smart home," (2015) 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Hua Hin, 2015, pp. 1-5, doi: 10.1109/ECTICon.2015.7206960.



Suggest a Recipe from near Expiring Food

IT21829710 | Perera P.A.E.U.K.

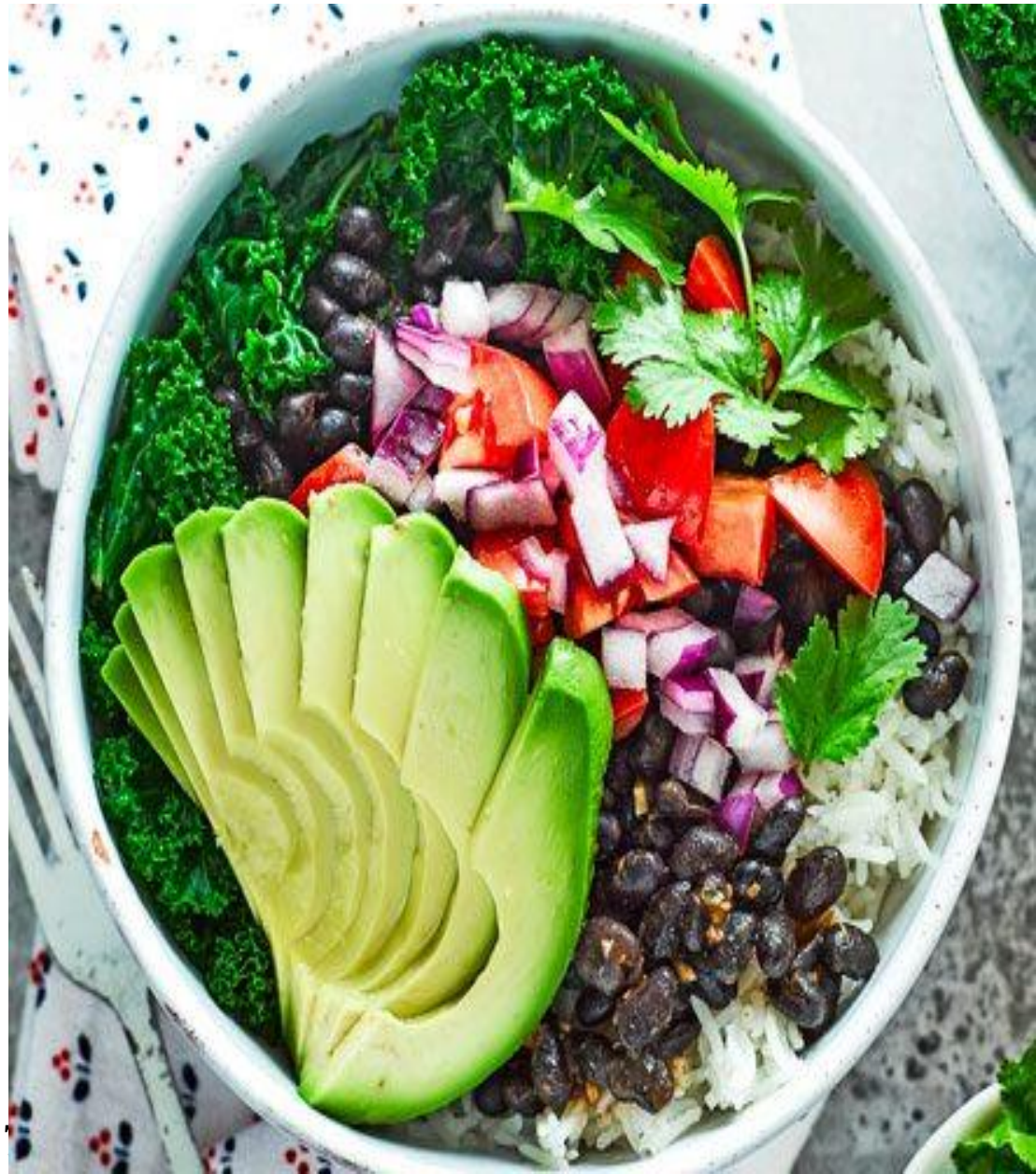
SPECIALIZING IN INFORMATION TECHNOLOGY

COMPONENT - 4

Introduction

FreshiFy

Focuses on personalized recommendations to reduce food waste by tracking expirations and suggesting recipes based on user profiles (health status, preferences, allergies and age).



What I did?

- I trained neural network-based ML model to predict risk food for users
- Trained AI algorithms to generate the recipes based on the near expiring food

Data set and Model Training

FreshiFy

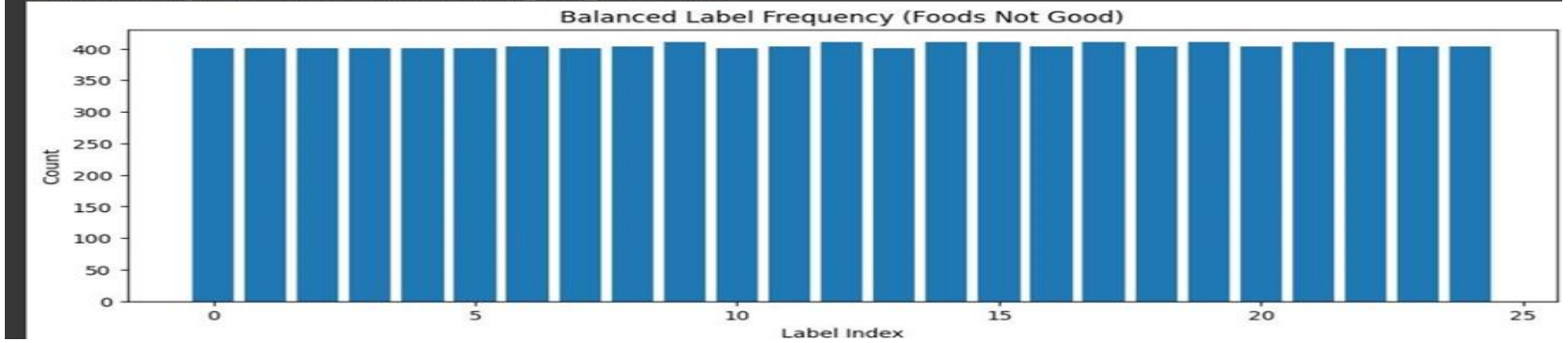
To fetch the user details – I get the user input for their preferences, health status, BMI and age.

Based on that data, I will train the model to predict the allergies food and risky food for the user.

By using all these conditions then available recipes will generate.

Patient_ID	Age	Gender	Height_cm	Weight_kg	BMI	Chronic_Disease	Blood_Pressure_Systolic	Blood_Pressure_Diastolic	Cholesterol_Level	Blood_Sugar_Level	Genetic_Risk_Factor	Allergies
P00001	56	Other	163	66	24.84	None	175	75	219	124	No	None
P00002	69	Female	171	114	38.99	None	155	72	208	72	No	None
P00003	46	Female	172	119	40.22	None	137	101	171	145	No	Gluten Intoler
P00004	32	Female	197	118	30.41	None	148	91	258	235	No	Nut Allergy
P00005	60	Female	156	109	44.79	Hypertension	160	109	260	248	Yes	None
P00006	25	Female	189	99	27.71	Diabetes	146	100	237	214	No	None
P00007	78	Female	199	61	15.4	Hypertension	165	103	189	118	Yes	None
P00008	38	Female	174	96	31.71	Hypertension	92	83	234	218	No	None
P00009	56	Female	162	116	44.2	Heart Disease	170	72	243	157	Yes	None
P00010	75	Male	170	114	39.45	Heart Disease	121	70	160	221	Yes	None
P00011	36	Other	190	76	21.05	None	97	74	266	109	No	None
P00012	47	Male	153	75	32.04	Diabetes	160	72	256	169	No	Nut Allergy
P00013	46	Male	165	71	26.08	None	180	99	150	224	Yes	Nut Allergy
P00014	45	Other	162	111	42.3	None	150	102	276	236	No	Lactose Intole
P00015	43	Male	161	88	33.95	Heart Disease	133	70	211	189	Yes	None
P00016	59	Female	169	93	32.56	None	127	105	193	160	Yes	None

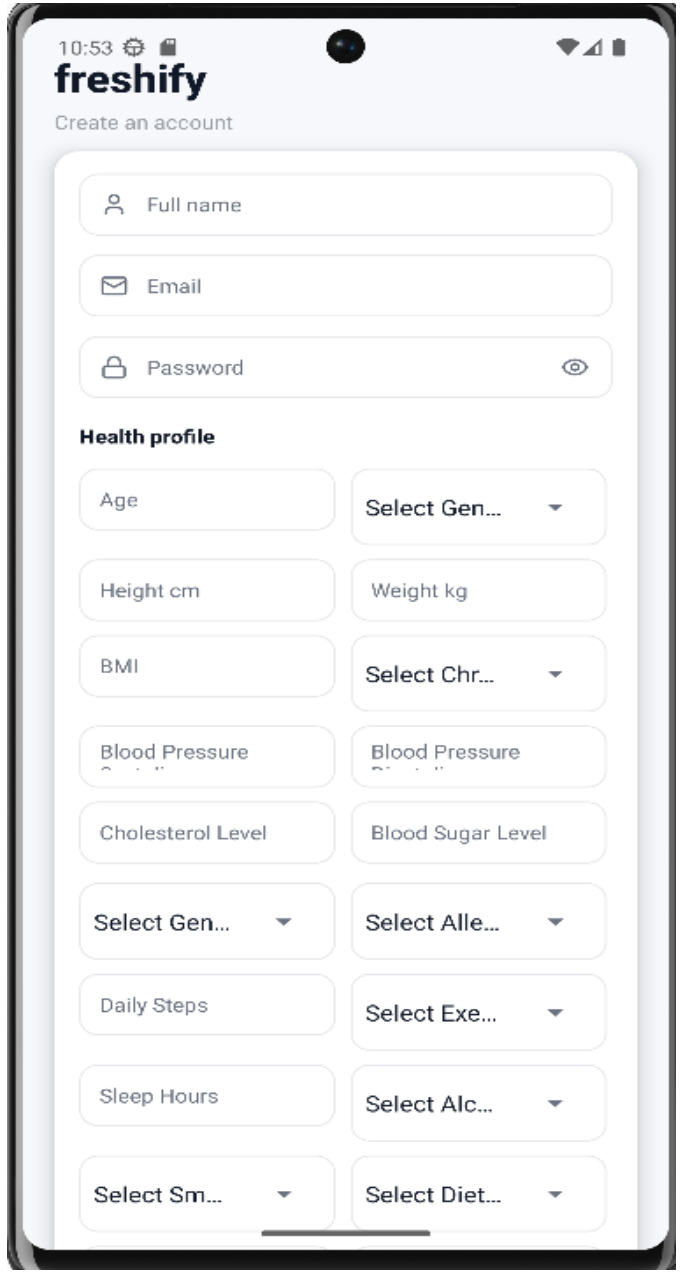
```
Epoch 45/50
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - binary_accuracy: 0.9993 - loss: 0.0023 - val_binary_accuracy: 0.9364 - val_loss: 0.1
Epoch 46/50
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - binary_accuracy: 0.9999 - loss: 9.6683e-04 - val_binary_accuracy: 0.9359 - val_loss:
Epoch 47/50
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - binary_accuracy: 0.9997 - loss: 9.7252e-04 - val_binary_accuracy: 0.9357 - val_loss:
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Final Accuracy: 0.8560661411081968
Model training complete and saved to food_risk_model.keras
```

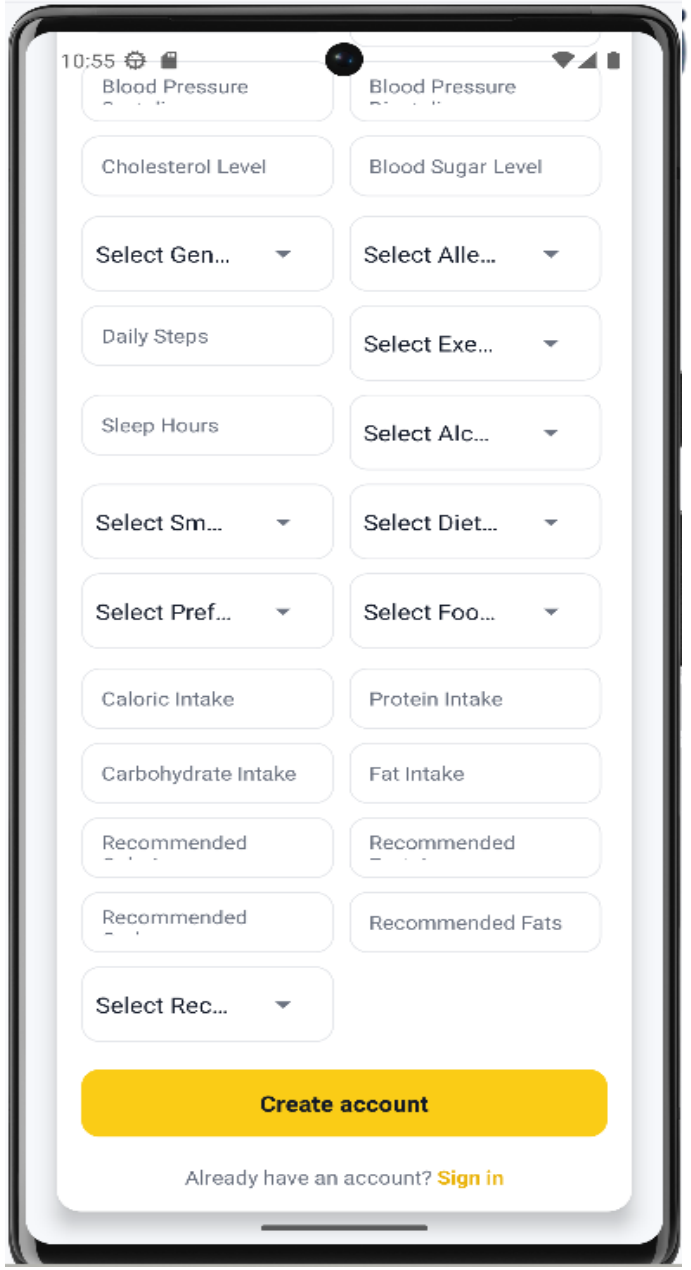


To get user inputs

User enter their details to this interface and register with the system.

These data will save in the mongo DB Database.

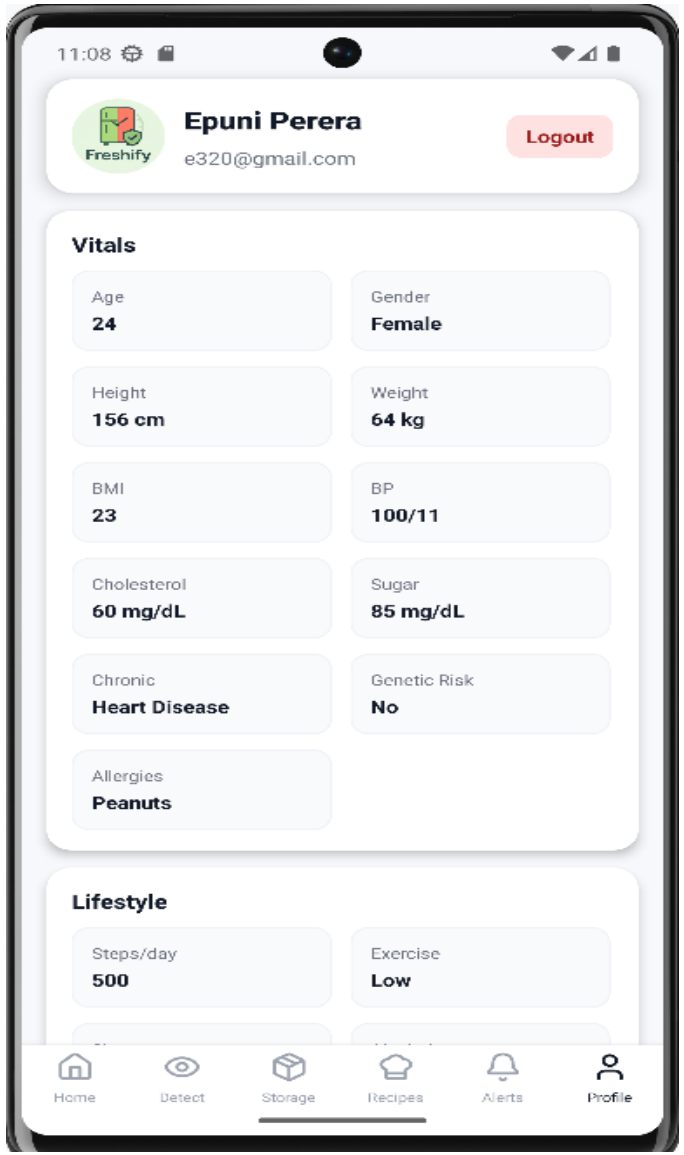


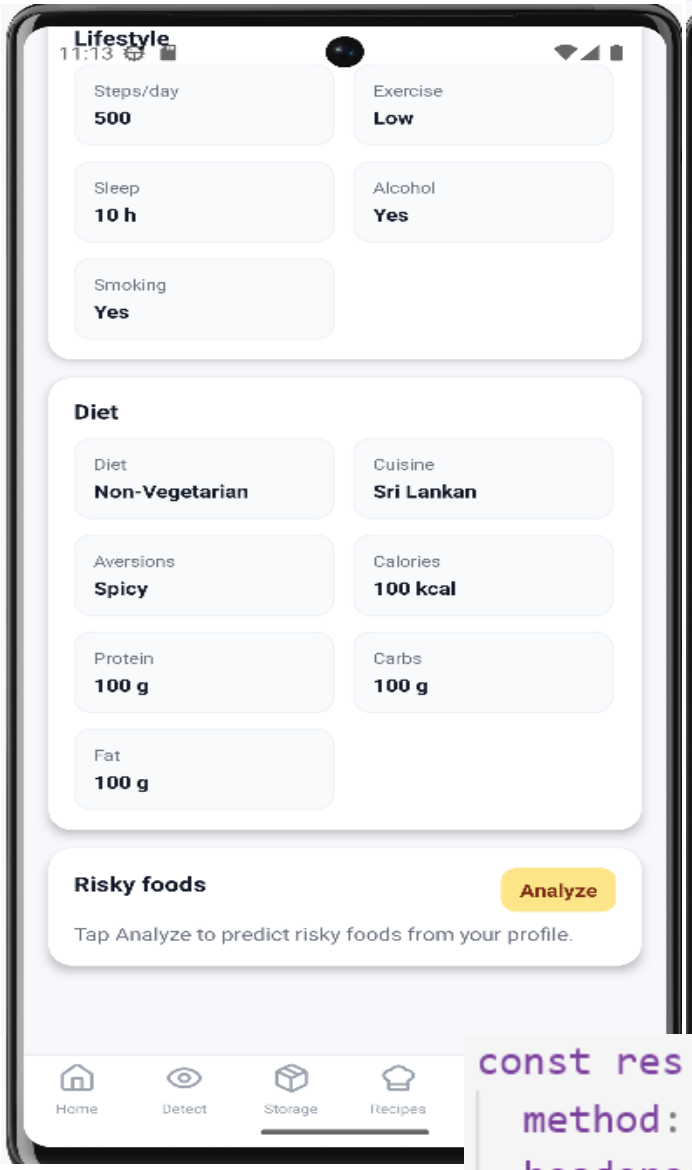


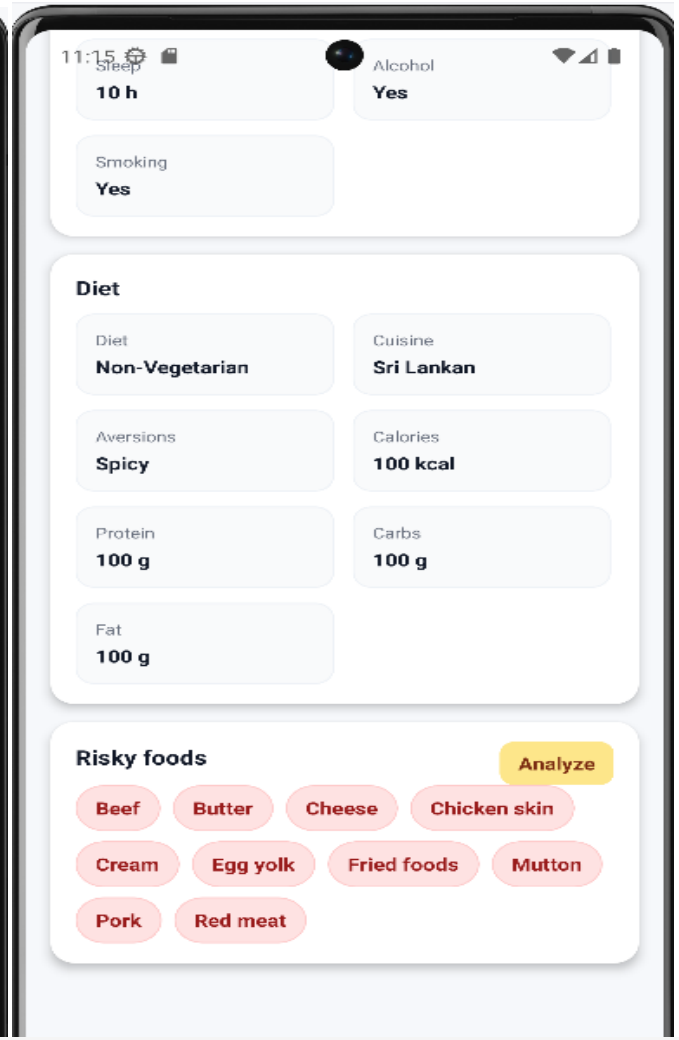
FreshiFy

User Profile

FreshiFy








```
const res = await fetch(`${FLASK_BASE}/predict-food-risk`, {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(body),
});
```

```python
@app.route( rule: '/predict-food-risk', methods=['POST'])
def predict_food_risk():
    try:
        new_patient = request.get_json()
        df_input = pd.DataFrame([new_patient])

        # Binary mapping
        binary_cols = ['Genetic_Risk_Factor', 'Alcohol_Consumption', 'Smoking_Habit']
        for col in binary_cols:
            if col in df_input.columns:
                df_input[col] = df_input[col].map({'Yes': 1, 'No': 0}).fillna(0)

        # Categorical to string + one-hot
        cat_cols = ['Gender', 'Allergies', 'Dietary_Habits', 'Preferred_Cuisine', 'Food_A
        for col in cat_cols:
            if col in df_input.columns:
                df_input[col] = df_input[col].astype(str)

        df_input = pd.get_dummies(df_input)

        # Feature engineering
        if 'BMI' in new_patient:
            df_input['BMI_Category'] = pd.cut( x: [new_patient['BMI']],  bins: [0, 18.5, 25,
        if 'Cholesterol_Level' in new_patient:
```

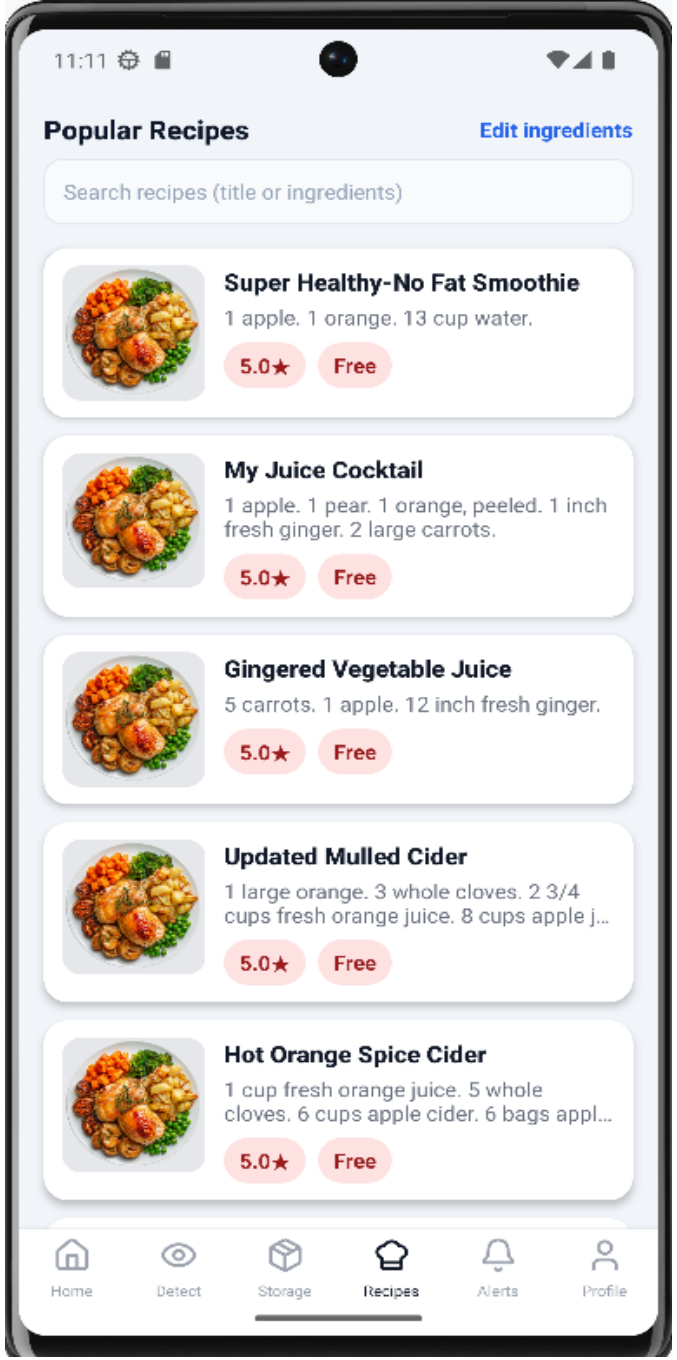
```python
        if 'BMI' in new_patient:
            df_input['BMI_Category'] = pd.cut( x: [new_patient['BMI']],  bins: [0, 18.5, 25
        if 'Cholesterol_Level' in new_patient:
            df_input['High_Cholesterol'] = int(new_patient['Cholesterol_Level'] > 240)
        if 'Blood_Pressure_Systolic' in new_patient:
            df_input['High_BP'] = int(new_patient['Blood_Pressure_Systolic'] > 140)

        df_input.drop(columns=['BMI', 'Cholesterol_Level', 'Blood_Pressure_Systolic'], e

        # Align with expected model input
        for col in feature_columns:
            if col not in df_input.columns:
                df_input[col] = 0
        df_input = df_input[feature_columns]

        # Predict
        X_scaled = food_scaler.transform(df_input)
        pred = food_model.predict(X_scaled)
        predicted_labels = food_label_binarizer.inverse_transform((pred > 0.5).astype(in

        return jsonify({"risky_foods": predicted_labels[0]})
    except Exception as e:
        return jsonify({"error": str(e)})
```

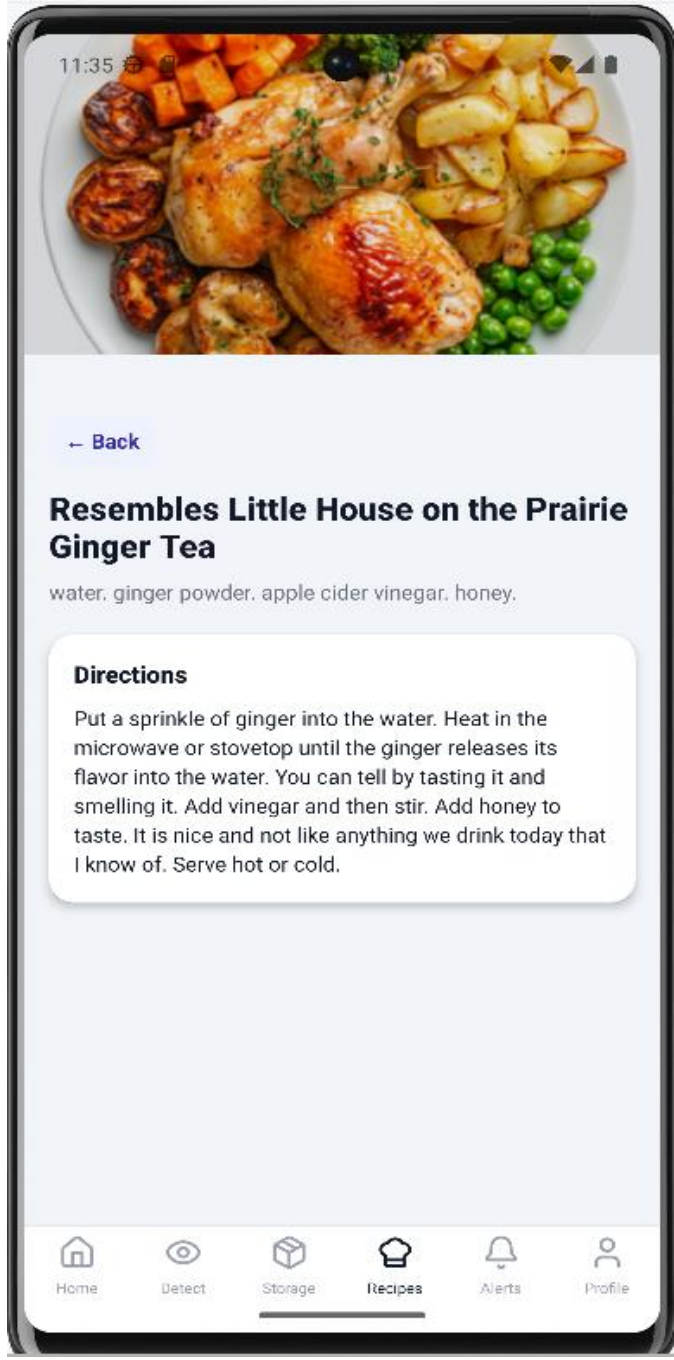
Recipe Generation Page

Based on the user details, risky food for user and age generate the user friendly and healthy recipes.

We can generate up to 50 recipes.

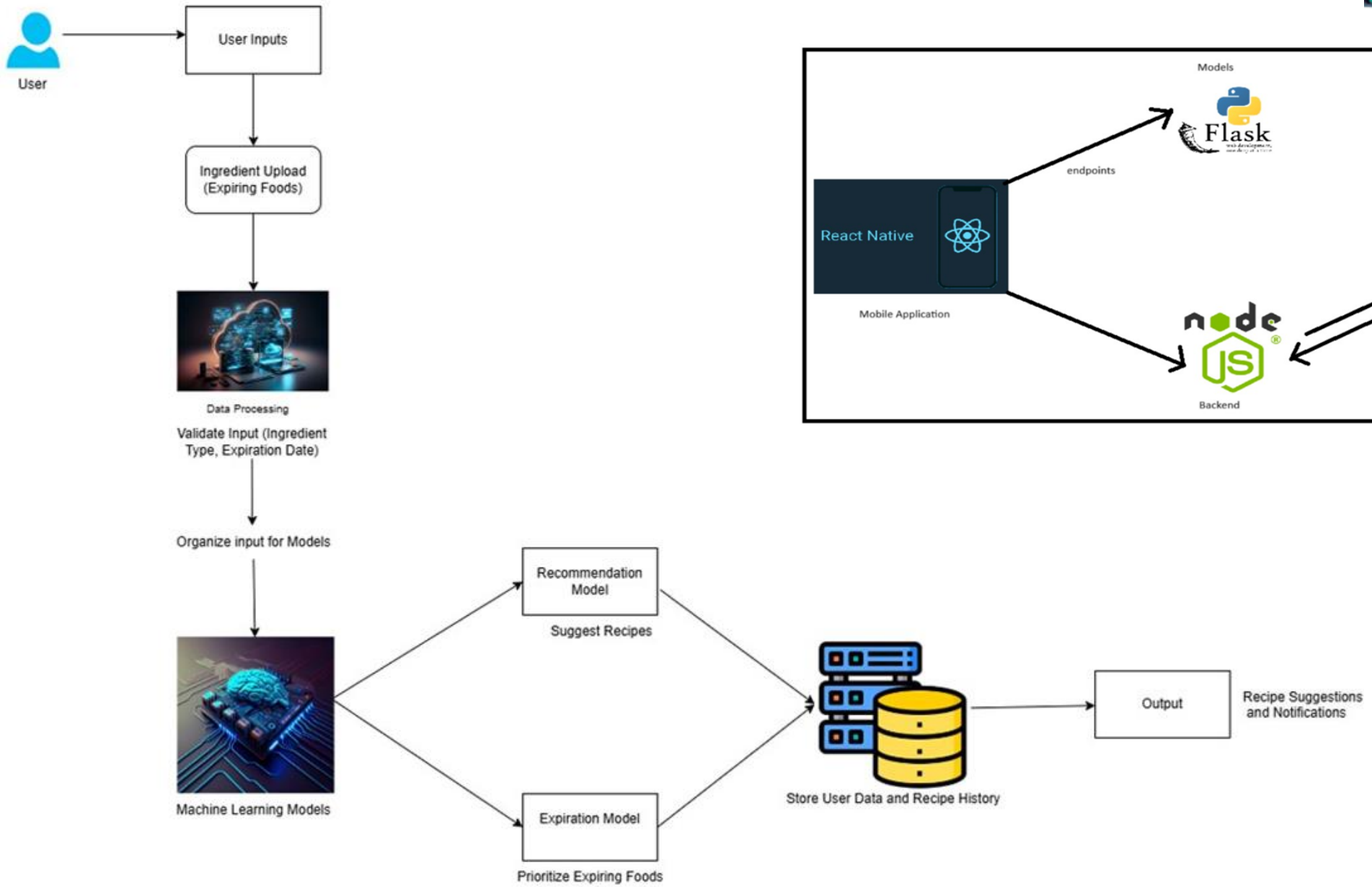
```
const [records] = useState(50);
```

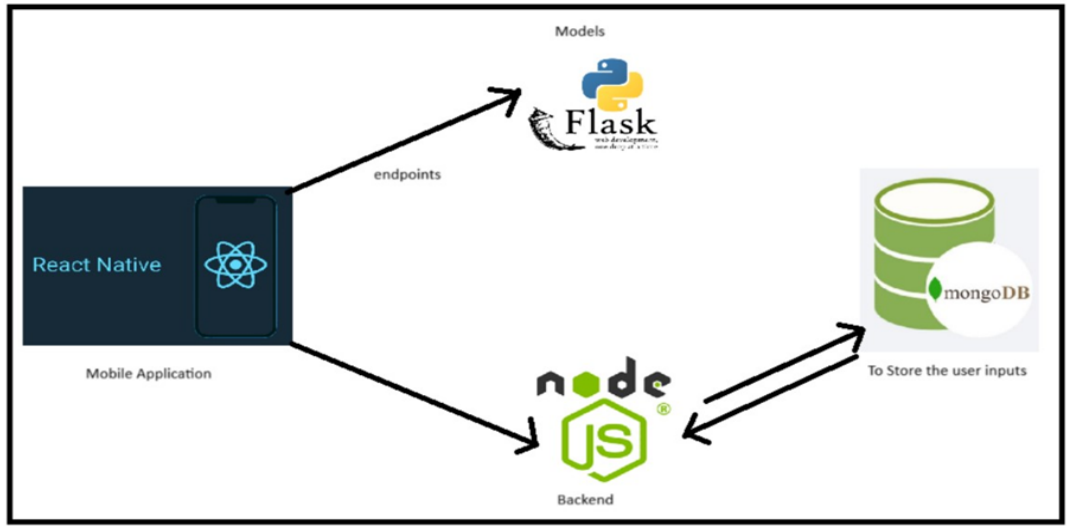




System diagram

FreshiFy





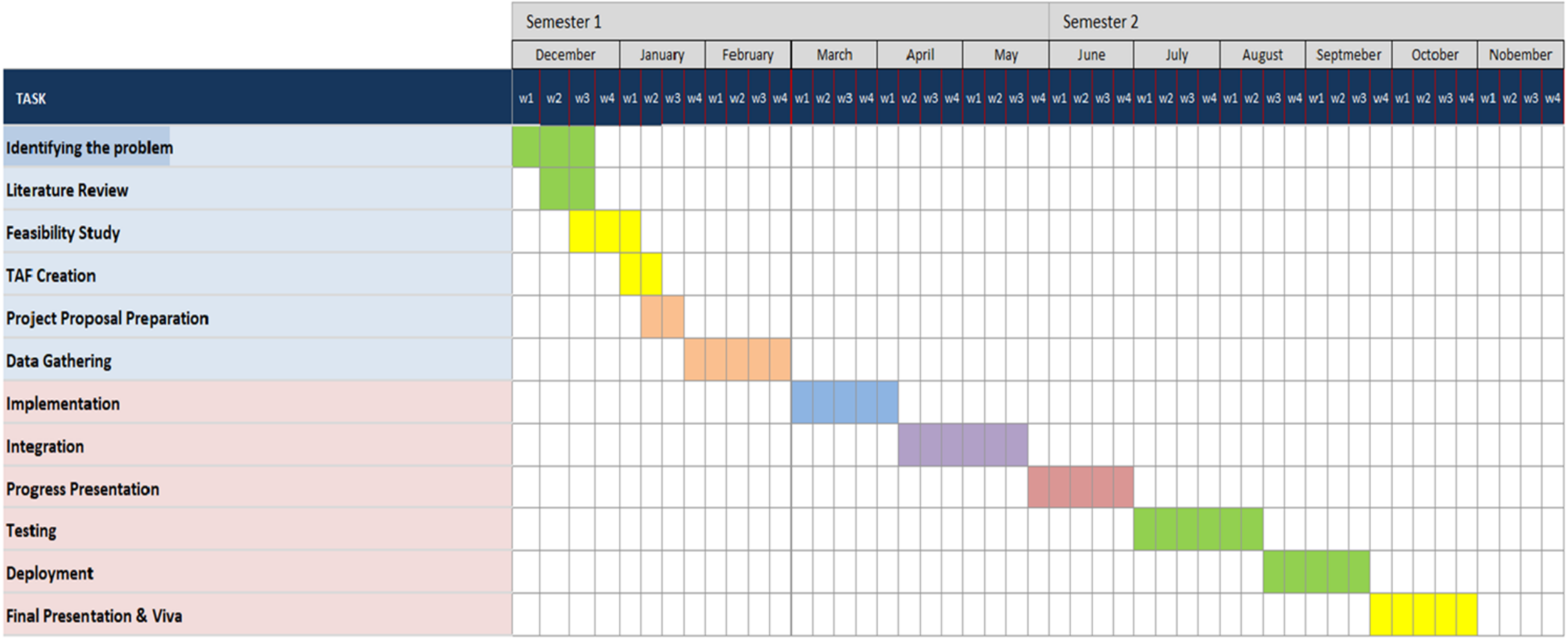
References

- Wang, X., & Li, Z. (2019). *Personalized recipe recommendations using user profiles and preferences.* Journal of Culinary Studies, 12(3), 54-67.

- Patel, S., & Thomas, R. (2020). *Collaborative filtering-based recipe recommendation systems: An analysis of algorithms.* Journal of AI Applications, 9(1), 34-45.

- Anderson, M., & Green, J. (2021). *Reducing food waste through meal planning: A study on the effectiveness of digital tools.* Journal of Sustainable Living, 14(4), 88-97.

Gannt Chart


Semester 1
Semester 2
December
January
February
March
April
May
June
July
August
Septmeber
October
Nobember
TASK
w1 w2 w3 w4
Identifying the problem
Literature Review
Feasibility Study
TAF Creation
Project Proposal Preparation
Data Gathering
Implementation
Integration
Progress Presentation
Testing
Deployment
Final Presentation & Viva

FreshiFy