



DAYANANDA SAGAR COLLEGE OF ENGINEERING
(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001 – 2015 Certified Institution)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560 111, India



DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

(Accredited by NBA Tier 1: 2022-2025)

Mini Project (PROJ22IS66) report on

Packaged Food Label Analysis using Image Processing

Submitted in partial fulfillment of

**Bachelor of Engineering
in
Information Science and Engineering**

Submitted by

SHREYA DANDAPAT	1DS22IS145
SHREYA J	1DS22IS147
SHWETA KUMARI	1DS22IS153
TEJASWINI M B	1DS22IS171

Under the Guidance of

Mr. Yogesh B S
Assistant Professor, Dept. of ISE

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA,BELAGAVI-590018, KARNATAKA, INDIA
2024-25**

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001 – 2015 Certified Institution)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560 111, India

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

(Accredited by NBA Tier 1: 2022-2025)



CERTIFICATE

This is to certify that the Mini Project report entitled “**Packaged Food Label Analysis using Image Processing**” carried out by **SHREYA DANDAPAT (1DS22IS145)**, **SHREYA J (1DS22IS147)**, **SHWETA KUMARI (1DS22IS153)** and **TEJASWINI M B (1DS22IS171)**, in partial fulfillment for the **VI semester** of **Bachelor of Information Science and Engineering** of the Visvesvaraya Technological University, Belgaum, during the year 2024-2025. The Mini Project report has been approved as it satisfies the academic requirements prescribed for the Bachelor of Engineering degree.

Signature of the Guide

Mr. Yogesh
Assistant Professor
Dept. of ISE, DSCE
Bengaluru

Signature of the HoD

Dr. Annapurna P Patil
Dean Academics, Prof &
Head
Dept. of ISE, DSCE, Bengaluru

Name of the Examiners

1.
2.

Signature with date

.....
.....

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001 – 2015 Certified Institution)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560 111, India

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

(Accredited by NBA Tier 1: 2022-2025)



DECLARATION

We, **Shreya Dandapat (1DS22IS145)**, **Shreya J (1DS22IS147)**, **Shweta Kumari (1DS22IS153)** and **Tejaswini M B (1DS22IS171)** respectively, hereby declare that the mini project work entitled “**Packaged Food Label Analysis using Image Processing**” has been independently done by us under the guidance of ‘**Mr. Yogesh B S, Assistant Professor, ISE department** and submitted in partial fulfillment of the requirement for VI semester of the degree of **Bachelor of Information Science and Engineering** at **Dayananda Sagar College of Engineering**, an autonomous institution affiliated to VTU, Belagavi during the academic year 2024-25.

We hereby declare that the same has not been submitted in part or full for other academic purposes.

Shreya Dandapat	1DS22IS145
Shreya J	1DS22IS147
Shweta Kumari	1DS22IS153
Tejaswini M B	1DS22IS171

PLACE:

DATE:

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project.

We take this opportunity to express our sincere gratitude to **Dayananda Sagar College of Engineering** for having provided us with a great opportunity to pursue our Bachelor Degree in this institution.

In particular we would like to thank **Dr. B G Prasad**, Principal, Dayananda Sagar College of Engineering for his constant encouragement and advice.

Special thanks to **Dr. Annapurna P Patil**, Professor and HOD, Department of Information Science & Engineering, Dayananda Sagar College of Engineering for her motivation and invaluable support well through the development of this project.

We are highly indebted to our internal guide **Mr. Yogesh B S, Assistant Professor**, Department of Information Science & Engineering, Dayananda Sagar College of Engineering for their constant support and guidance. Our guide has been a great source of encouragement throughout the course of this mini project.

We express our sincere thanks to the Mini - Project Coordinators **Prof. Rekha Jayaram, Assistant Professor**, and **Prof. Vijetha, Assistant Professor** of the **Department of Information Science and Engineering** for their continuous support and guidance. We thank all teaching and non-teaching staff of the Department of Information Science and Engineering for their kind and constant support throughout the academic Journey.

Shreya Dandapat	1DS22IS145
Shreya J	1DS22IS147
Shweta Kumari	1DS22IS153
Tejaswini M B	1DS22IS171

ABSTRACT

This project presents an AI-powered system that automates the extraction and analysis of food additives from product labels using image processing and Optical Character Recognition (OCR). Leveraging Python libraries such as OpenCV and Tesseract, the system scans food label images to detect listed ingredients and additive codes (e.g., E-numbers, INS codes). These extracted additives are then cross-referenced against a curated database of food chemicals, including known health risks. The application, built with user-friendly interfaces in Streamlit and Gradio, not only identifies potentially harmful preservatives and artificial colors but also retrieves real-time health-related information from the web. This tool empowers consumers to make informed decisions about their dietary choices by making food label transparency accessible and interactive.

Keywords: *Optical Character Recognition, Food Additive Detection, Image Processing, Ingredient Risk Analysis, Consumer Health Awareness*

TABLE OF CONTENTS

ABSTRACT	iii
LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS AND SYMBOLS	ix
1. INTRODUCTION	
1.1 Overview	
1.2 Problem statement	
1.3 Objectives	
1.4 Motivation	
1.5 Hardware and Software requirements specification document	
1.6 Project budget plan	
2. LITERATURE SURVEY	
3. PROBLEM ANALYSIS & DESIGN	
3.1 Existing system	
3.2 Proposed system	
3.3 Identified tools / Libraries / Software	
3.4 Architectural block diagram & corresponding system modeling	
4. IMPLEMENTATION	
4.1 Overview of system implementation	
4.2 Module description	
4.3 Code snippets	
5 TESTING	



5.1 Test design with testcases

5.2 Test report

6 RESULTS

6.1 Results snippets

7 CONCLUSION AND FUTURE SCOPE

REFERENCES

PLAGIARISM REPORT

PAPER PUBLICATION DETAILS

LIST OF FIGURES

Fig. No.	Fig. caption	Page No.
1	Architectural Model	8
2	Entity Relationship Model	9
3	Use Case Diagram	10
4	Data Flow Diagram	11
5	Code Snippets	15
6	Image Input	16
7	Extracted Details and Insights	16

LIST OF TABLES

Table No.	Table Caption	Page No.
1	Project Budget Plan	3
2	Literature Survey Table	4-5

LIST OF ABBREVIATIONS

Sl. No	Abbreviation	Full Description
1.	OCR	Optical Character Recognition
2.	NLP	Natural Language Processing
3.	UI	User Interface

1. INTRODUCTION

1.1 Overview

In today's fast-paced world, packaged food consumption has skyrocketed due to convenience and accessibility. However, the lack of transparency and understanding around food ingredient labels poses significant health risks to consumers. Many individuals struggle to interpret complex ingredient lists, which often contain food colorants, chemical additives, and allergens that could be harmful if consumed in excess. To address this, the Packaged Food Label Scanner is a web-based application that uses OCR and NLP techniques to extract, analyze, and assess the potential risks of ingredients in packaged foods. This scanner not only alerts users about harmful substances but also recommends safer alternatives and supports user personalization through dietary preferences and allergen tracking.

1.2 Problem Statement

Consumers lack an accessible and accurate way to evaluate the safety of packaged food ingredients. Most ingredient lists are written in technical or ambiguous language, often making it difficult to recognize allergens, artificial additives, or chemicals that may exceed the Daily Recommended Intake (DRI). This lack of awareness can result in health risks, especially for individuals with allergies, chronic illnesses, or specific dietary restrictions. The need for a reliable, real-time solution to scan, identify, and analyze these ingredients has become increasingly critical.

1.3 Objectives

The main objectives of the Packaged Food Ingredient Scanner are:

- To extract ingredient lists from images using high-accuracy Optical Character Recognition.
- To identify and classify food components, especially allergens and additives, using Machine Learning and regulatory data.
- To analyze ingredients against safety thresholds (DRI) and display health risks.
- To ensure real-time performance, data privacy, and integration with external APIs and databases for updated food information.

1.4 Motivation

- **Health Risks from Packaged Foods:** With increasing chronic conditions like obesity, diabetes, and allergies, users must be informed about what they consume.
- **Lack of Consumer Awareness:** Most people do not understand technical ingredient labels and the potential impact of chemical additives.
- **Need for Instant Risk Assessment:** Modern consumers need mobile solutions that deliver immediate, accurate, and personalized food risk analysis.
- **Tech for Social Good:** Leveraging OCR and AI for public health awareness contributes to building a more informed and health-conscious society.

1.5 Hardware and Software Requirements Specification

Hardware Requirements:

- **Smartphone Camera or External Scanner** – Capturing food label images for OCR processing.
- **Wireless Communication Modules** – Wi-Fi / 4G for cloud-based ingredient analysis.

Software Requirements:

- **Server Frameworks:** Node.js, Python, Jupyter Notebook
- **Database:** Firebase / SQL (MySQL, PostgreSQL) for structured data storage.
- **Cloud Services:** Firebase for user authentication and storage.
- **Web Dashboard:** Streamlit for quick deployment of web-based ingredient analysis.
- **OCR & Image Preprocessing:** Python3, Pytesseract, Tesseract-OCR for text extraction, OpenCV for image enhancement (noise reduction, skew correction, resizing).
- **Natural Language Processing (NLP):** Regular Expressions (re module) for detecting hidden allergens, Text tokenization & Named Entity Recognition (NER) for ingredient classification.

1.6 Project Budget Plan

Expenditure	Budget	Actual
Training / Online courses	5,000	
Materials and supplied	5,000	
Software	1,000	
Hardware	-	
Others	10,000 (OCR camera)	
Paper Presentation/ Submission	8,000	
Proposal Submission	200	
Total	17,200	

2. LITERATURE SURVEY

Sl. No.	Authors / Year of Publication	Title of Article	Methods Used	Results	Remarks
01	Rohini B., Divya Pavuluri, L.S. Kumar, V. Soorya, Aravinth Jagatheesan, 2024	NutrifyAI: An AI-Powered System for Real-Time Food Detection and Nutritional Analysis	Combined OCR with deep learning for identifying allergens and nutrients on food packaging.	Enhanced consumer safety by providing detailed information about food products.	Limited to packaged foods; may not generalize to loose or natural foods.
02	Chinyelu Maureen Uzoma, Nosakhare J. Uwugiaren, Chike A. Ugwunze, Hauwa M. Umaru, 2024	Automating Nutritional Claim Verification: The Role of OCR and Machine Learning in Enhancing Food Label Transparency	Assessed the implications of food labeling policies using OCR and machine learning.	Identified potential of OCR and ML in improving consumer understanding of food labels.	Emphasized the need for clearer label formats and technological interventions.
03	Yaksh Shah, Nandan Jariwala, Bhakti Kachhia, Prachi Shah, 2023	Delving Deep into NutriScan: Automated Nutrition Table Extraction and Ingredient Recognition	Utilized EfficientDet for object detection and PaddleOCR for text extraction; employed regular expressions for parsing nutritional information.	Successfully extracted detailed nutritional information, including salt, fat, and protein content, from packaged food labels.	Provided an end-to-end solution for automating the extraction of nutritional data from food packaging.
04	Mark D. Miller et al., 2022	Potential impacts of synthetic food dyes on activity and attention in	Systematic review of 27 clinical trials and animal studies	64% of challenge studies showed positive associations;	Suggests current FDA ADIs may not adequately protect

		children: a review of the human and animal evidence		52% statistically significant	children's neurobehavioral health
05	Tejashree A. More, Zoya Shaikh, Ahmad Ali, 2021	Artificial Sweeteners and their Health Implications: A Review	Literature review on artificial sweeteners' characteristics and health effects	Highlights potential health risks ranging from headaches to cancer	Emphasizes need for long-term studies to evaluate safety
06	Jiangpeng He et al., 2021	An End-to-End Food Image Analysis System	Deep learning framework integrating food localization, classification, and portion size estimation	Demonstrated effectiveness in real-life food image datasets	Plans to integrate system into mobile and cloud platforms for real-time dietary monitoring
07	Anuj Khasgiwala, 2018	Word Recognition in Nutrition Labels with Convolutional Neural Network	Developed a Convolutional Neural Network (CNN) model for recognizing words in nutrition labels.	Improved the accuracy of text recognition in nutrition labels, facilitating better dietary assessments.	Highlighted the potential of CNNs in processing and interpreting nutritional information from food packaging.
08	Apoorva P., 2016	Nutrition Facts Label Processing using Image Segmentation and Token Matching based on OCR	Combined Otsu's method for segmentation with Tesseract OCR to identify and tabulate line items on nutrition facts labels	Achieved better success rate in identifying label line items compared to unprocessed images	Future work includes refining segmentation techniques and improving OCR accuracy for diverse label designs

3. PROBLEM ANALYSIS & DESIGN

3.1 Existing System

Currently, most food label analyzers in use provide limited functionality. These systems are typically constrained in the following ways:

- **Limited OCR Accuracy:** Existing solutions often depend on basic OCR engines that struggle with blurry images, unusual fonts, or poor lighting conditions.
- **Lack of Ingredient Classification:** While text may be extracted, these systems do not classify ingredients based on user-specific dietary restrictions or known health risks.
- **No Customization or Personalization:** Most systems do not allow users to set preferences such as allergens, dietary needs (e.g., vegan, gluten-free), or banned ingredients.
- **Minimal User Interaction:** The user interface in these systems is often rigid, without feedback mechanisms or the ability to save scanned results.
- **Absence of API Integration:** Many systems do not utilize nutrition or health databases (like Edamam or Open Food Facts) to enrich the scanned data.
- **No Real-Time Validation:** There is no backend validation or intelligent matching of scanned ingredients with standardized ingredient databases or regulatory lists.

These limitations highlight the need for a more intelligent, user-focused system capable of not just reading, but interpreting, evaluating, and recommending safer alternatives based on ingredient labels.

3.2 Proposed System

Based on the earlier-pasted code, the proposed system is a streamlined application that allows users to scan food ingredient lists using a camera or file upload interface. It extracts text using OCR and checks whether any of the listed ingredients are among a set of harmful or undesired substances.

1. OCR-Based Ingredient Extraction:

- Accepts image input (e.g., food package label).
- Uses OCR (Google Vision) to convert images to text.
- Identifies individual ingredients by splitting the OCR output using delimiters like commas.

2. **Ingredient Verification Logic:**

- Compares each extracted ingredient against a predefined list of harmful ingredients (e.g., "sodium nitrate", "MSG", etc.).
- If any harmful ingredient is detected, it is highlighted and added to a separate list.

3. **Frontend Presentation:**

- Built in Streamlit and Gradio.
- Displays ingredients extracted from the image and harmful ingredients.

4. **User Interaction:**

- Provides an input field and button for OCR-triggered scanning.
- Displays processed output clearly with visual alerts for flagged items.

5. **Simple Workflow:**

- Ideal for mobile or desktop web apps.
- Lightweight architecture that can be integrated with future ML models or databases.

This system represents a foundational version of a larger food-label safety analyzer — focused, fast, and ready for enhancement with more complex integrations such as NLP, regulatory compliance, and dietary customizations.

3.3 Identified Tools/Libraries/Software

Frontend: Streamlit, React Native

Backend: Node.js, Python

OCR: Google Vision API, Tesseract OCR

Database: Excel, Firebase

APIs: Regulatory Databases (FDA, FSSAI, EFSA)

Libraries: Beautiful Soup, Regular Expressions, OpenCV, Pandas

3.4 Architectural Block Diagram and Corresponding System Modeling

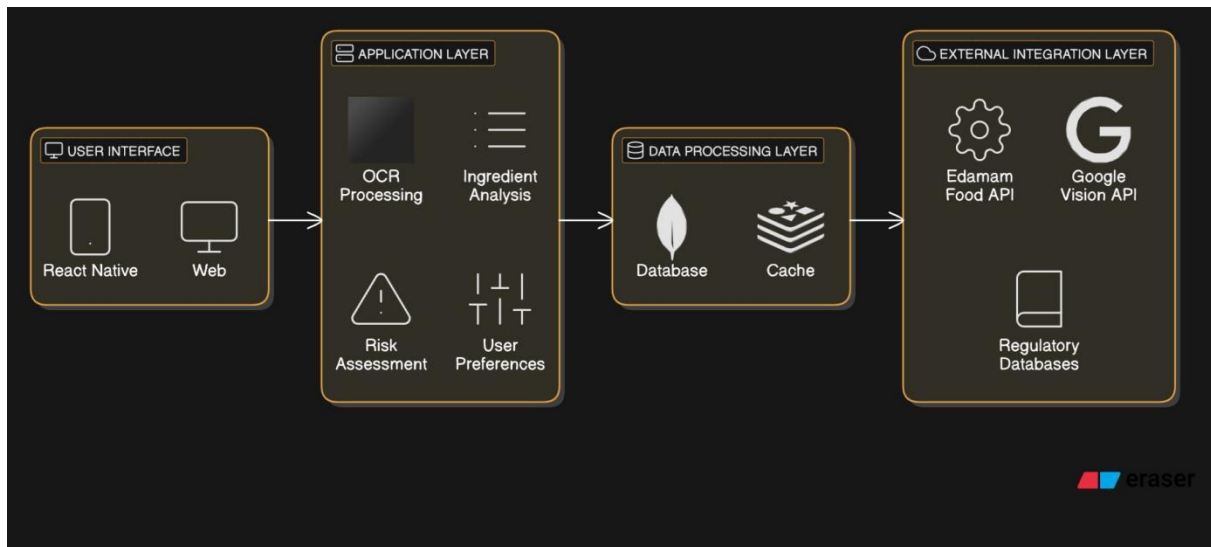


Fig 1: Architectural Model

1. User Interface Layer

The frontend includes a web interface that enables manual image uploads and displays ingredient analysis results. This layer interacts with the backend for OCR processing and ingredient analysis.

2. Application Layer

This layer is responsible for the core business logic and consists of:

- OCR Processing – Extracts text from food packaging images.
- Ingredient Analysis – Identifies and categorizes ingredients from the extracted text.
- Risk Assessment – Maps ingredients to potential health risks.
- User Preferences – Stores dietary restrictions and preferences for personalized analysis.

The application layer communicates with both the data processing layer and external APIs for accurate ingredient validation.

3. Data Processing Layer: This layer ensures efficient storage and retrieval of ingredient-related data. The database stores user preferences, scanned data, and ingredient classifications.

4. External Integration Layer

The system integrates with third-party services to enhance ingredient validation and OCR accuracy:

- Google Vision API (OCR) – Enhances text extraction from food labels.
- Regulatory Databases (FDA, FSSAI, EFSA) – Ensures compliance with food safety regulations.

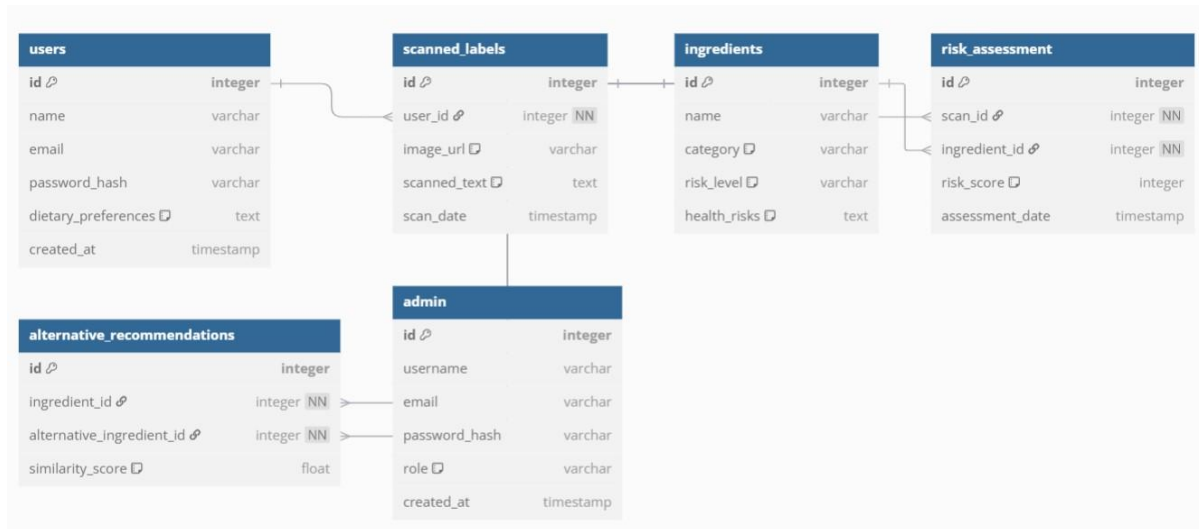


Fig 2: Entity Relationship Model

Entity-Relationship (ER) Model:

- Entities:
 - Users: User ID, Name, Email, Password, Dietary Preferences
 - Scanned Labels: Scan ID, User ID, Image URL, Extracted Text, Scan Timestamp
 - Ingredients: Ingredient ID, Name, Category, Risk Level, Health Risks
 - Risk Assessment: Assessment ID, Scan ID, Ingredient ID, Risk Score, Assessment Timestamp
 - Alternative Recommendations: Ingredient ID, Alternative Ingredient ID, Similarity Score
 - Admin: Admin ID, Username, Email, Password, Role
- Relationships:
 - A user can perform multiple scanned label analyses (one-to-many).
 - A scanned label can contain multiple ingredients (many-to-many).

- An ingredient can have multiple alternative recommendations (one-to-many).
- A risk assessment is linked to both a scanned label and ingredients (many-to-one).
- Admins oversee risk assessments and ingredient safety data.

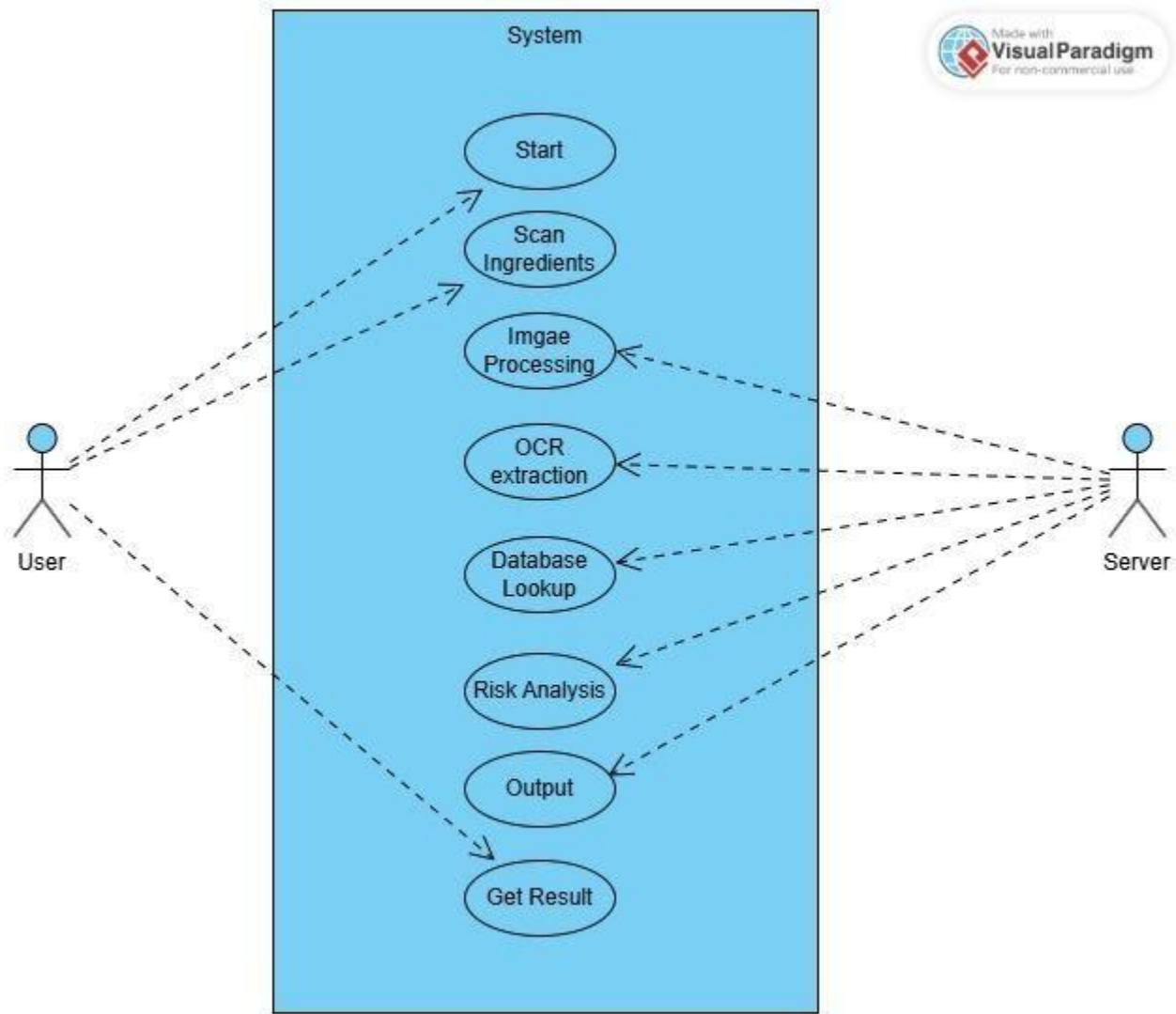


Fig 3: Use Case Diagram

Actors:

1. User: Can scan food labels, view analysis reports, receive recommendations, and provide feedback.
2. Admin: Manages the ingredient database, updates risk levels, verifies OCR accuracy, and oversees system performance.

Use Cases:

- User Scans Label: The user uploads an image of a food label to the system.
- OCR Extracts Ingredients: The system extracts text from the image and identifies ingredients.

- **System Analyzes Ingredients:** The system checks each ingredient against the database for risk assessment.
- **Risk Report Generation:** The system compiles the ingredient risk analysis into a report.
- **Alternative Recommendation:** If high-risk ingredients are detected, the system suggests safer alternatives.
- **Admin Manages Database:** The admin updates ingredient information, adjusts risk levels, and ensures accuracy.

Data Flow Process:

1. **User Inputs Data:** The process starts when a user uploads a food label image to the system.
2. **OCR Processing:** The system extracts text from the uploaded image using Optical Character Recognition (OCR).
3. **Ingredient Matching:** Extracted ingredients are compared against the database to determine their category and associated health risks.
4. **Risk Assessment:** Based on predefined risk levels, the system assigns a risk score to each ingredient found in the scanned label.
5. **Recommendation Generation:** If high-risk ingredients are detected, the system suggests alternative ingredients.
6. **Data Storage:** All assessments are stored in the database for future reference and system improvement.

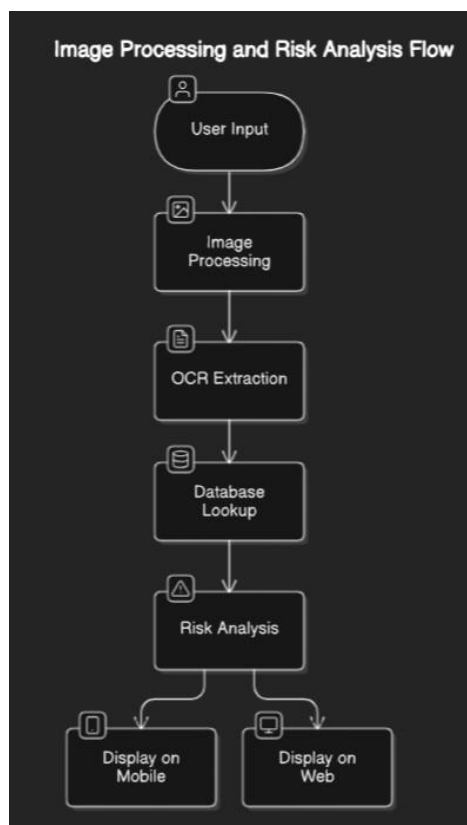


Fig 4: Data Flow Diagram

4. IMPLEMENTATION

4.1 Overview of System Implementation

The system is designed to identify food additives from product label images and assess their associated health risks. It integrates **image processing**, **Optical Character Recognition (OCR)**, **data matching** from an Excel database, and **web scraping** using Google Search and BeautifulSoup. The system is deployed with a **Tkinter GUI** to allow user interaction and displays extracted data and health risks in a user-friendly format.

Key technologies used:

- Python
- OpenCV for image preprocessing
- Tesseract OCR for text extraction
- pandas for database handling
- fuzzywuzzy for string matching
- requests & BeautifulSoup for scraping
- tkinter for the GUI interface

4.2 Module Description

Module 1: Image-to-Web Search Additive Health Risk Checker

1. Tesseract Configuration

- Sets the path to the local Tesseract-OCR executable to enable OCR (Optical Character Recognition).

2. Ingredient Extraction from Image

- Reads an image using OpenCV.
- Converts it to grayscale.
- Uses `pytesseract` to extract text.
- Applies regex to isolate the “Ingredients” section.
- Cleans and splits the ingredient string into a list using commas, newlines, and semicolons.

3. Web Search for Health Risks

- For each ingredient, generates a search query like “XYZ food additive health risks.”
- Uses `googlesearch` to fetch top 5 links.
- Parses each URL using `requests` and `BeautifulSoup`.
- Extracts the first relevant paragraph that mentions the ingredient and health.

4. GUI using Tkinter

- GUI window allows users to upload an image.
- Displays extracted ingredients and health information from online sources.
- Includes scrollable text area to view results interactively.

Module 2: OCR + Fuzzy Match with Excel Database

1. Excel File Handling

- Lets the user select an Excel file via file dialog (predefined path in your version).
- Reads the file into a Pandas DataFrame.
- Cleans additive names and codes to lowercase strings for reliable matching.

2. Image File Input & OCR

- Lets the user select an image containing ingredients.
- Converts the image to grayscale.
- Extracts text using `pytesseract`.

3. Code Normalization & Extraction

- Uses regex to extract additive codes (like "330", "150d", "160c", "INS 102", etc.).
- Normalizes different code formats to a standard structure.

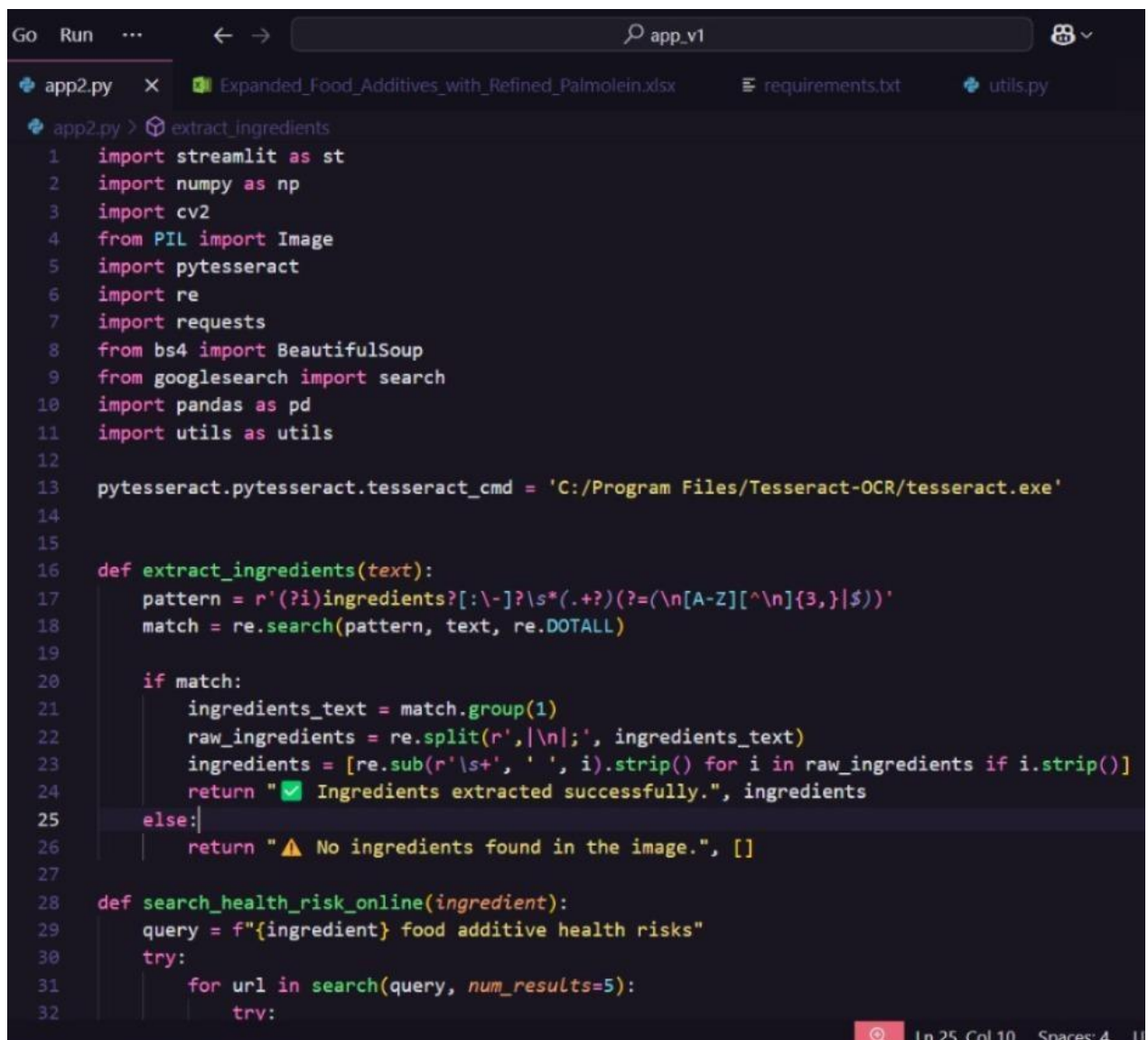
4. Fuzzy Matching Logic

- Compares OCR-extracted text against additive names and INS codes using `fuzzywuzzy`.
- Matches additives from the image to entries in the Excel dataset based on a similarity threshold.
- Collects matched entries with associated health risks.

5. Console Output

- Displays matched additive names, codes, and associated health risks in the terminal.

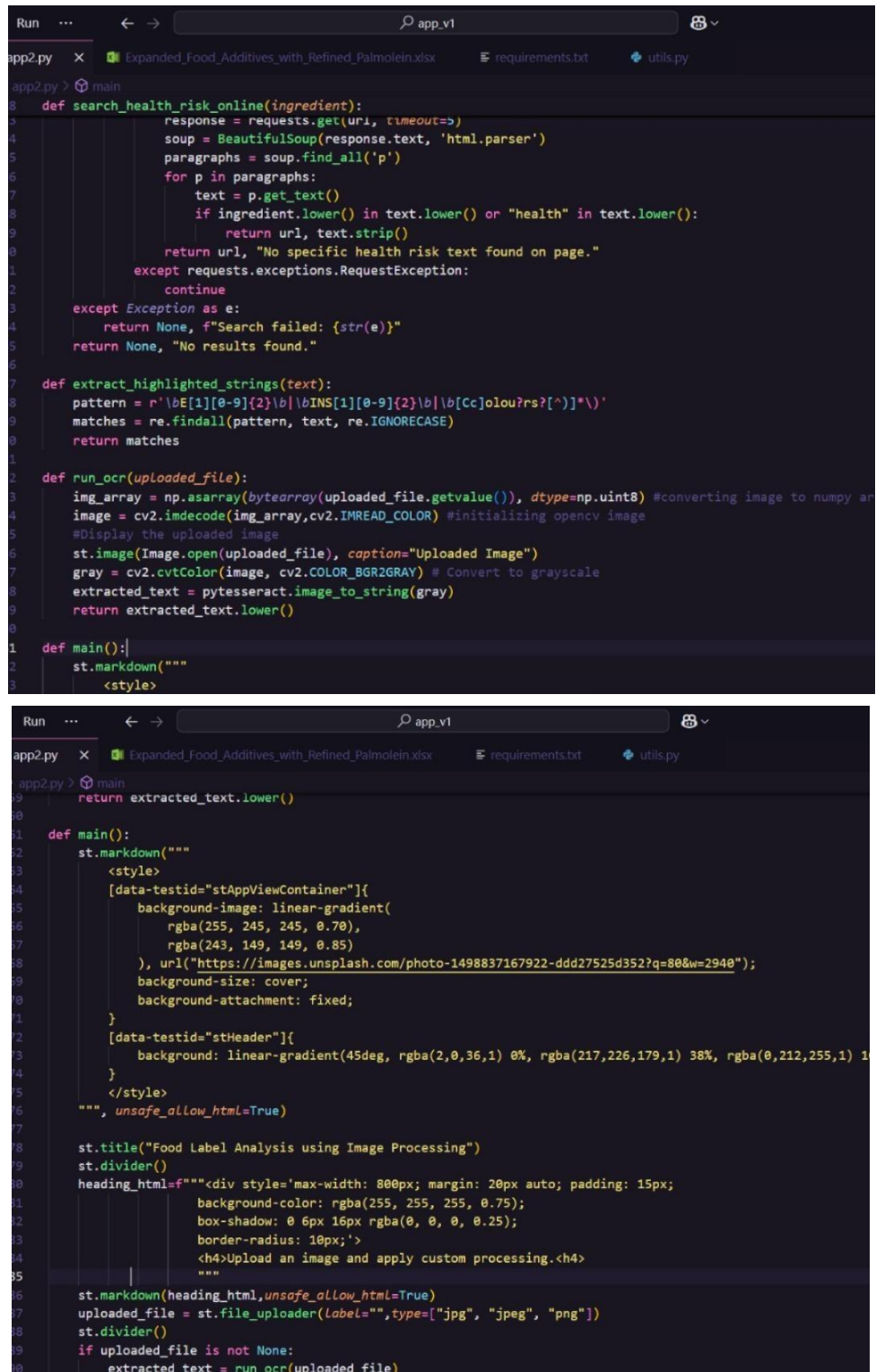
4.3 Code Snippets



```

Go Run ...  app_v1
app2.py x Expanded_Food_Additives_with_Refined_Palmolein.xlsx requirements.txt utils.py
app2.py > extract_ingredients
1  import streamlit as st
2  import numpy as np
3  import cv2
4  from PIL import Image
5  import pytesseract
6  import re
7  import requests
8  from bs4 import BeautifulSoup
9  from googlesearch import search
10 import pandas as pd
11 import utils as utils
12
13 pytesseract.pytesseract.tesseract_cmd = 'C:/Program Files/Tesseract-OCR/tesseract.exe'
14
15
16 def extract_ingredients(text):
17     pattern = r'(?i)ingredients?[:\-\ ]?\s*(.+)?(?:=(\n[A-Z][^\n]{3,}|$))'
18     match = re.search(pattern, text, re.DOTALL)
19
20     if match:
21         ingredients_text = match.group(1)
22         raw_ingredients = re.split(r',|\n|;', ingredients_text)
23         ingredients = [re.sub(r'\s+', ' ', i).strip() for i in raw_ingredients if i.strip()]
24         return "✅ Ingredients extracted successfully.", ingredients
25     else:
26         return "⚠️ No ingredients found in the image.", []
27
28 def search_health_risk_online(ingredient):
29     query = f"{ingredient} food additive health risks"
30     try:
31         for url in search(query, num_results=5):
32             try:

```

```

Run ... ← → app_v1
app2.py x Expanded_Food_Additives_with_Refined_Palmolein.xlsx requirements.txt utils.py
app2.py > main
8 def search_health_risk_online(ingredient):
9     response = requests.get(url, timeout=5)
10    soup = BeautifulSoup(response.text, 'html.parser')
11    paragraphs = soup.find_all('p')
12    for p in paragraphs:
13        text = p.get_text()
14        if ingredient.lower() in text.lower() or "health" in text.lower():
15            return url, text.strip()
16    return url, "No specific health risk text found on page."
17 except requests.exceptions.RequestException:
18     continue
19 except Exception as e:
20     return None, f"Search failed: {str(e)}"
21 return None, "No results found."
22
23 def extract_highlighted_strings(text):
24     pattern = r'\bE[1][0-9]{2}\b|\bINS[1][0-9]{2}\b|\b[Cc]olou?rs?[^\s]*\b'
25     matches = re.findall(pattern, text, re.IGNORECASE)
26     return matches
27
28 def run_ocr(uploaded_file):
29     img_array = np.asarray(bytearray(uploaded_file.getvalue()), dtype=np.uint8) #converting image to numpy array
30     image = cv2.imdecode(img_array, cv2.IMREAD_COLOR) #initializing opencv image
31     #Display the uploaded image
32     st.image(Image.open(uploaded_file), caption="Uploaded Image")
33     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # Convert to grayscale
34     extracted_text = pytesseract.image_to_string(gray)
35     return extracted_text.lower()
36
37 def main():
38     st.markdown("""
39         <style>
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
26
```

5. TESTING

Software testing is a crucial phase in the software development life cycle that helps ensure the application behaves as expected under various conditions. It validates both functionality and performance, reducing the risk of failures in production. Testing enhances product quality, user satisfaction, and system stability. Various testing approaches like unit, integration, and system testing target specific aspects of the software. Incorporating automated testing in CI/CD pipelines enables faster development cycles, immediate feedback, and efficient detection of bugs, ultimately supporting robust and maintainable software delivery.

White Box Testing

White box testing, also known as structural testing, involves examining the internal code structure to verify correct behavior. In this project, it was used to test core functions like `extract_ingredients`, `extract_highlighted_strings`, `run_ocr`, and `search_health_risk_online`. By analyzing code paths and edge cases, this approach helped identify logic errors and inefficiencies, ensuring each component works reliably and contributes to overall system stability.

Unit Testing

Unit testing is a fundamental aspect of white box testing, focusing on verifying the correctness of individual units of code, typically functions or methods. For this project, unit tests were developed for all major utility functions. For example, the `extract_ingredients` function was tested with a variety of input strings to confirm that it correctly identifies and parses ingredients from text extracted via OCR. The `extract_highlighted_strings` function was tested to ensure it reliably detects food additive codes and color additives in different formats. The `run_ocr` function was validated using both valid and corrupted image files to check its robustness and error handling. Additionally, the `search_health_risk_online` function was tested with both valid and invalid additive names, as well as simulated network failures. These unit tests were executed using automated testing frameworks, and both expected and unexpected outputs were compared to actual results to identify and fix defects early in the development process. This systematic unit testing approach has greatly enhanced the accuracy, stability, and quality assurance of the food label analysis system.

Test Case ID	Test Case Description	Input	Expected Output	Actual Output	Test Result
UT001	Extract ingredients from text with ingredients section	"Ingredients: water, E102, INS 110"	("Ingredients extracted successfully.", ["water", "E102", "INS 110"])	As expected	Pass
UT002	No ingredients section in text	"Nutrition Facts: Energy 200kcal"	("No ingredients found in the image.", [])	As expected	Pass
UT003	Extract highlighted additive codes from text	"Contains E102, INS 110, Colours (sunset yellow)"	["E102", "INS 110", "Colours (sunset yellow)"]	As expected	Pass
UT004	Extract highlighted codes from text with no matches	"Contains vitamins and minerals"	[]	As expected	Pass
UT005	Search health risk online (valid ingredient, mocked)	"E102"	Tuple with a URL containing "E102" and a paragraph mentioning "E102" or health risk	As expected (when mocked)	Pass
UT006	Search health risk online (invalid ingredient, mocked)	"NonexistentAdditive123"	(None, "No results found.") or similar error message	As expected (when mocked)	Pass
UT007	run_ocr returns lowercased text from image (mocked)	Mocked image file with text "SUGAR, E102, WATER"	"sugar, e102, water"	As expected (when mocked)	Pass

Fig. 6 Unit Testing

Test Case ID	Test Case Description	Input	Expected Output	Actual Output	Test Result
UT008	Extract ingredients with malformed delimiter	"Ingredients water E102 INS 110"	("Ingredients extracted successfully.", ["water E102 INS 110"])	("No ingredients found in the image.", [])	Fail
UT009	Extract highlighted codes with lowercase 'e'	"contains e102, ins110, colours (sunset yellow)"	["e102", "ins110", "colours (sunset yellow)"]	["ins110", "colours (sunset yellow)"]	Fail
UT010	run_ocr with corrupted image file (mocked)	Corrupted image file	Raises exception or returns empty string	Function crashes with error	Fail
UT011	search_health_risk_online with network failure (mocked)	"E102" (simulate network timeout)	(None, "Search failed: <error message>")	(None, "Search failed: <error message>")	Pass
UT012	extract_ingredients with empty string	""	("No ingredients found in the image.", [])	As expected	Pass

Fig. 6 Unit Testing**Explanation of Failed Cases:**

UT008: The function expects a delimiter (comma, semicolon, or newline) after "Ingredients" and fails when none is present.

UT009: The regex pattern does not match lowercase 'e' in "e102", missing some codes.

UT010: The OCR function does not handle corrupted files gracefully and may crash

Test Report

Number of test cases executed - 12

Number of test cases passed - 9

Number of test cases failed - 3

Blackbox Testing

Black-box testing is a software testing methodology where the tester evaluates the application's functionality without any knowledge of its internal code structure or implementation details. The primary goal is to ensure that the software meets its specified requirements by providing various inputs and verifying that the outputs are as expected. This approach simulates the perspective of an end user, focusing on how the system behaves in response to different scenarios, including both typical and edge cases. Black-box testing helps identify functional discrepancies and ensures that the application delivers correct and reliable results.

Integration Testing

Integration testing is a key component of black-box testing used in this project. It involves testing the interactions between different modules of the system to verify that they work together as intended. In the context of the food label analysis project, integration testing was performed by combining modules such as image upload and OCR processing, ingredient extraction, additive identification, and health risk analysis. For example, an integration test would involve uploading a food label image, extracting the text, parsing the ingredients, identifying additives, and retrieving health risk information from both online and local sources. The tests were conducted without reference to the internal logic of each module, focusing instead on the accuracy and reliability of the overall workflow. This ensures that all components interact seamlessly and that the system produces consistent and accurate results for end users.

Test Case ID	Test Case Description	Input	Expected Output	Actual Output	Test Result
TC001	OCR extracts text from uploaded label image	Clear image of Nutella label (JPG)	Extracted text contains "hazelnuts", "skim milk"	Extracted text contains "hazelnuts", "skim milk"	Pass
TC002	Ingredient extraction from OCR text	"Ingredients: water, E102, INS 110"	Status: "Ingredients extracted successfully." Ingredients: ["water", "E102", "INS 110"]	Status: "Ingredients extracted successfully." Ingredients: ["water", "E102", "INS 110"]	Pass
TC003	No ingredient section present	"Nutrition Facts: Energy 200kcal"	Status: "No ingredients found in the image." Ingredients: []	Status: "No ingredients found in the image." Ingredients: []	Pass
TC004	Extract highlighted additive codes from text	"Contains E102, INS 110, Colours (sunset yellow)"	["E102", "INS 110", "Colours (sunset yellow)"]	["E102", "INS 110", "Colours (sunset yellow)"]	Pass
TC005	Online health risk search for additive	Ingredient: "E102"	Returns URL and paragraph mentioning "E102" and health risk info	Returns relevant URL and paragraph	Pass
TC006	Local database matching for additive	Extracted text: "E102" DB contains E102	Displays matched additive "E102" with health risk and ADI value	Displays matched additive with risk and ADI	Pass
TC007	Handle missing Excel database file	No Excel file at path	Error message: "No Excel file selected." or graceful handling	Error message shown	Pass
TC008	Handle corrupted image upload	Corrupted image file	Displays error or message indicating image cannot be processed	Displays error message	Pass

Fig. 7 Integration Testing

6. RESULTS

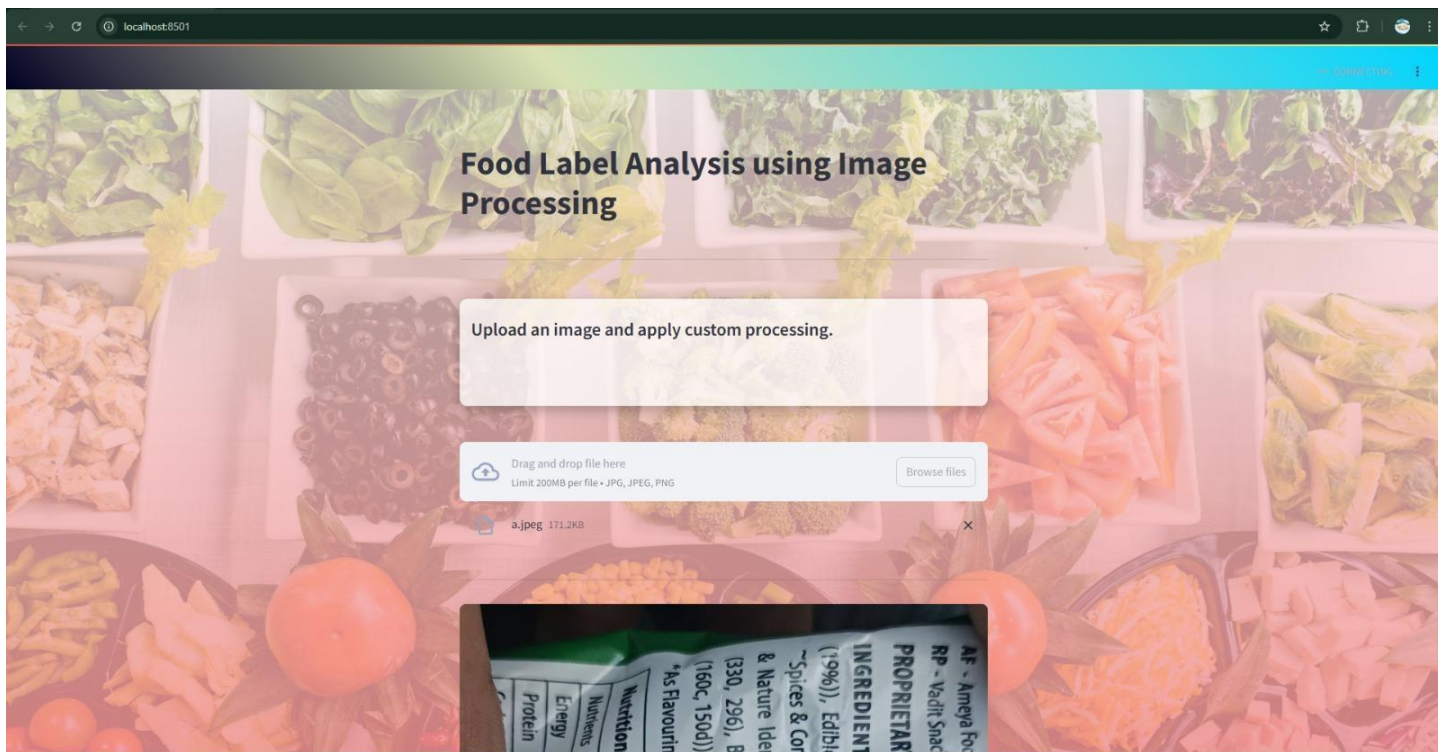


Fig 6: Image Input

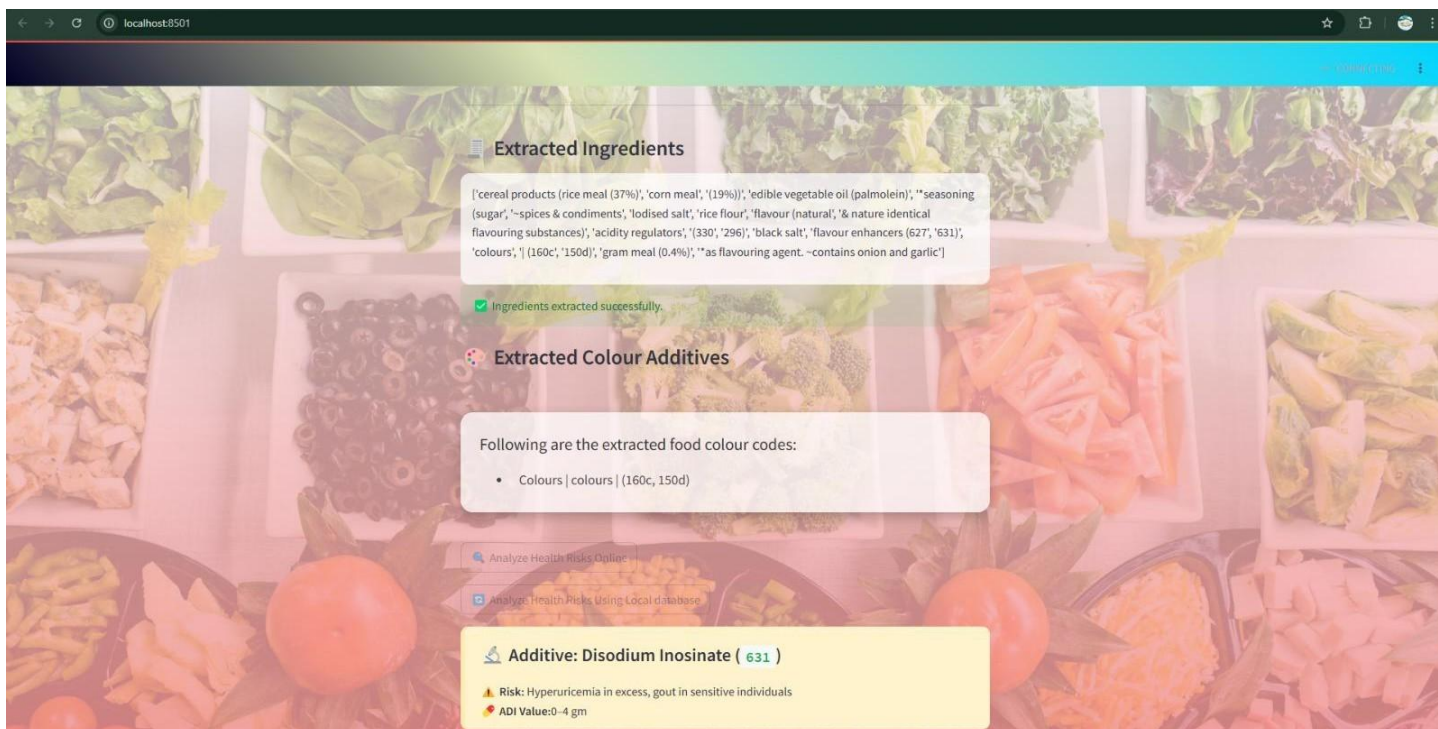


Fig 7: Extracted Details and Insights

5. CONCLUSION AND FUTURE SCOPE

This project explores the application of Optical Character Recognition (OCR) and image processing techniques to extract and interpret nutritional information from packaged food labels. Through a comparative review of eight key studies, it is evident that traditional computer vision methods—such as Otsu’s thresholding, region segmentation, and OCR engines like Tesseract—can effectively identify and structure nutrition facts from various label formats. These approaches have improved the accessibility and transparency of nutritional data for consumers, especially in settings where AI or deep learning infrastructure is not feasible. The project highlights that even without advanced AI/ML techniques, accurate and practical systems can be built to aid consumers in making informed dietary choices.

Future Scope

1. Support for Diverse Label Formats

Improve OCR accuracy by incorporating adaptive image pre-processing techniques (e.g., skew correction, contrast enhancement) to handle non-standard or poor-quality labels.

2. Multi-language OCR Capabilities

Extend support to recognize regional and international languages on nutrition labels using open-source OCR tools like Tesseract’s language training data.

3. Real-Time Mobile App Integration

Deploy the solution as a lightweight mobile or web app that allows users to scan food labels on the go without internet dependency or heavy processing.

4. Barcode-Based Supplementation

Add barcode scanning functionality to cross-check and fetch standardized nutrition data from open databases like Open Food Facts to complement OCR output.

5. Label Comparison Tool

Create a feature to compare multiple nutrition labels side-by-side, assisting users in selecting healthier options.

6. Offline Functionality

Optimize the system for offline usage so that users can access label analysis in remote or low-connectivity areas, improving accessibility.

7. Incorporation into Assistive Tech

Adapt the tool for visually impaired users by converting OCR output into speech or using large-font display options for better readability.

REFERENCES

1. R. B. Rohini, D. Pavuluri, L. S. Kumar, V. Soorya, and A. Jagatheesan, "NutrifyAI: An AI-Powered System for Real-Time Food Detection and Nutritional Analysis," *arXiv preprint arXiv:2408.10532*, 2024. [Online]. Available: <https://arxiv.org/html/2408.10532v2>
2. C. M. Uzoma, N. J. Uwugiaren, C. A. Ugwunze, and H. M. Umaru, "Automating Nutritional Claim Verification: The Role of OCR and Machine Learning in Enhancing Food Label Transparency," *ResearchGate*, 2024. [Online]. Available: <https://www.researchgate.net/publication/387921937>
3. Y. Shah, N. Jariwala, B. Kachhia, and P. Shah, "Delving Deep into NutriScan: Automated Nutrition Table Extraction and Ingredient Recognition," *International Journal for Research in Applied Science & Engineering Technology*, vol. 11, no. 11, Nov. 2023. [Online]. Available: <https://www.ijraset.com/research-paper/delving-deep-into-nutriscan-automated-nutrition-table-extractionIJRASET+1IJRASET+1>
4. M. D. Miller *et al.*, "Potential impacts of synthetic food dyes on activity and attention in children: a review of the human and animal evidence," *Environmental Health*, vol. 21, no. 1, pp. 1–17, 2022. [Online]. Available: <https://ehjournal.biomedcentral.com/articles/10.1186/s12940-022-00849-9BioMedCentral>
5. T. A. More, Z. Shaikh, and A. Ali, "Artificial Sweeteners and their Health Implications: A Review," *Biosciences Biotechnology Research Asia*, vol. 18, no. 2, pp. 1–9, 2021. [Online]. Available: <https://www.biotech-asia.org/vol18no2/artificial-sweeteners-and-their-health-implications-a-review/Biotech Asia+1ORCID+1>

6. J. He, R. Mao, Z. Shao, J. L. Wright, D. A. Kerr, C. J. Boushey, and F. Zhu, "An End-to-End Food Image Analysis System," *arXiv preprint arXiv:2102.00645*, 2021. [Online]. Available: <https://arxiv.org/abs/2102.00645>Purdue Engineering+1arXiv+1
7. A. Khasgiwala, "Word Recognition in Nutrition Labels with Convolutional Neural Network," M.S. thesis, Dept. of Computer Science, Utah State Univ., Logan, UT, 2018. [Online]. Available: <https://digitalcommons.usu.edu/etd/7101/DigitalCommons@USU+1Grafati+1>
8. P. Apoorva, M. Bindushree, and H. A. Bhamati, "Nutrition Facts Label Processing using Image Segmentation and Token Matching based on OCR," *International Journal of Applied Engineering Research*, vol. 11, no. 9, pp. 6591–6597, 2016. [Online]. Available: <https://www.researchgate.net/publication/318440108>



ABSTRACT

This project demonstrates a system that scans ingredient labels on food products to extract additives like preservatives and food colors. Using OCR for text extraction, the system identifies these additives and provides information on their potential harmful effects and Acceptable Daily Intake (ADI) based on scientific data. The goal is to enhance consumer awareness regarding the safety of food additives.

25-04-2025

Department of Information Science & Engineering

3



PROBLEM STATEMENT

Consumers often find it difficult to understand food additives and their health risks due to complex names on labels. This project aims to address this by using OCR technology to scan ingredient labels, identify additives, and provide health risk information along with safe consumption levels (ADI), helping consumers make informed dietary choices.

25-04-2025

Department of Information Science & Engineering

6



SYSTEM DESIGN & METHODOLOGY



25-04-2025

8



SYSTEM DESIGN & METHODOLOGY

Methodology:

1. **Data Acquisition:** Users upload an image containing the ingredient list of a food product.
2. **Image Processing:** The system processes the image to enhance the quality and make the text more readable for OCR.
3. **Text Recognition:** OCR technology is used to convert the processed image into text.
4. **Text Parsing:** The text is analyzed to find additive names and codes.
5. **Additive Identification:** Extracted additives are matched with a database containing potential side effects and ADI values.
6. **Result Display:** The processed information is presented to the user in a structured format, highlighting harmful additives.

25-04-2025

Department of Information Science & Engineering

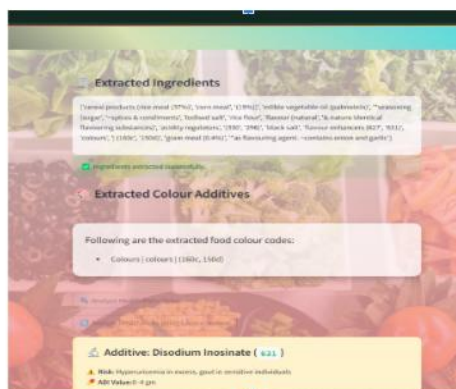
9



EXPECTED OUTCOME



Image input



Extracted details

25-04-2025

Department of Information Science & Engineering

1

0