# 1.INTRODUCTION

The increasing awareness of healthy eating habits has highlighted the importance of understanding the nutritional content of our food. With the advancements in technology, especially in data science and machine learning, it has become feasible to create automated systems that can analyze and detect the nutritional values of various food items. This document introduces a nutrition value detection system using Python, focusing on its significance, fundamental components, and potential applications. Users are required to log in to access the system. Once authenticated, they can upload images of food and beverages. The system uses deep learning models to analyze the images and estimate their caloric content. This information is then presented in a detailed nutrition report, enabling users to make better dietary decisions.

## 1.1 Project Scope

The Nutrition Value Detection System aims to simplify the process of tracking nutritional intake by allowing users to upload images of their food. By leveraging deep learning and image recognition technologies, the system will analyze the images and estimate the calorie content and other nutritional values. This system is designed to cater to health-conscious individuals, dieticians, and fitness enthusiasts who need a reliable and easy-to-use tool for monitoring their diet.

## 2.1 Project Purpose

The primary purpose of this project is to provide an automated and efficient way for users to track their nutritional intake without the need for manual entry. Traditional methods of logging food intake can be tedious and prone to errors. By automating this process, the system aims to reduce the burden on users and provide more accurate and consistent nutritional data.

## 1.3 Project Features

**User Authentication**: Secure login system to ensure user data privacy.

**Image Upload:** Users can upload pictures of their meals.

**Calorie Estimation:** The system analyzes the images to estimate calorie content.

**Nutritional Report:** Detailed report including calories, macronutrients, and micronutrients.

**User-Friendly Interface**: Simple and intuitive design for easy navigation and use.

# 2.LITERATURE SURVEY

The literature on nutrition value detection systems using Python highlights the significant advancements in utilizing machine learning, computer vision, and natural language processing for nutritional analysis. The integration of comprehensive databases, sophisticated models, and userfriendly applications underscores the potential of these systems to enhance dietary management and promote healthier eating habits. Continued research and development in this field are expected to yield even more accurate and accessible tools for nutritional analysis in the future.

## 2.1 Touch-Based Interface

Touch-based interfaces are widely used in modern technology due to their intuitive nature. They provide a direct way for users to interact with digital systems, making them ideal for applications like the Nutrition Value Detection System. These interfaces are crucial for ensuring ease of use, especially for users who may not be tech-savvy.

## 2.2 Accessibility Solutions for the Visually Impaired

Ensuring accessibility for visually impaired users is an important aspect of modern software development. Techniques such as screen readers, voice commands, and high-contrast interfaces can help make the Nutrition Value Detection System usable by a broader audience, promoting inclusivity.

## 2.3 Image-Based Nutritional Analysis

**2.3.1 Food Recognition and Classification**:

Advanced computer vision techniques have been applied to recognize and classify food items from images. These methods often use pre-trained models like VGGNet, ResNet, and InceptionNet for feature extraction.

**2.3.2 Portion Size Estimation**:

Accurately estimating the portion size from images is crucial for determining nutritional content. Various techniques, including depth estimation and volume calculation, have been explored.

# 3.ANALYSIS AND DESIGN

## 3.1 System Analysis

### 3.1.1 Requirement Analysis

The requirement analysis phase involves gathering and analyzing the needs and expectations of the users and stakeholders. This includes functional requirements such as image upload and calorie estimation, as well as non-functional requirements like system reliability and user privacy.

### 3.1.2 Functional Requirements

**User Login and Authentication**: Secure login mechanism to protect user data.

**Image Upload Capability**: Allow users to upload images of their food.

**Image Processing and Analysis**: Utilize deep learning to analyze images and estimate nutritional values.

**Nutritional Report Generation**: Provide detailed nutritional information based on the analysis.

### 3.1.3 Non-Functional Requirements

**Performance**: The system should provide quick and accurate results.

**Scalability:** The system should be able to handle a large number of users and image uploads.

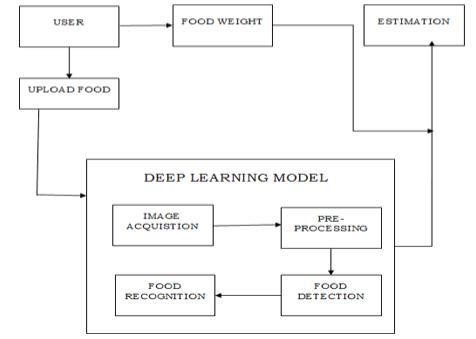
**Usability:** The interface should be user-friendly and accessible.

**Security:** Ensure data privacy and protect against unauthorized access.

## 3.2 System Design

### 3.2.1 Architecture Overview

The system architecture is designed to support efficient image processing and user interaction. It includes a client-server model where the client interface interacts with the server to upload images and receive nutritional reports. The server utilizes deep learning models to analyze the images and generate the reports.



### 3.2.2 Component Design

**Client Interface**: Handles user interactions, including login, image upload, and displaying nutritional reports.

**Server**: Manages user authentication, processes uploaded images, and runs the deep learning models for analysis.

**Database**: Stores user data, including login credentials and nutritional history.

## 3.3 Problem Definition

Current methods of tracking nutritional intake are often manual and time-consuming. Users have to input food details and calculate nutritional values themselves, which can lead to inaccuracies and reduced adherence to dietary plans. An automated system that can estimate nutritional values from images can significantly alleviate these issues.

## 3.4 Existing System

### 3.4.1 Limitations of Existing System

Existing nutritional tracking systems often rely on manual data entry, which is prone to human error. Additionally, they may not provide real-time feedback or be user-friendly, making them less effective for regular use.

## 3.5 Proposed System

### 3.5.1 Advantages of Proposed System

The proposed Nutrition Value Detection System offers several advantages:

**Automation**: Reduces the need for manual data entry.

**Accuracy**: Provides more reliable nutritional estimates using advanced image recognition.

**Convenience:** Simplifies the process of tracking dietary intake.

**Accessibility:** Designed to be user-friendly and inclusive.

## 3.6 Hardware and Software Requirements

### 3.6.1 Hardware Requirements

**Server:** A robust server to handle image processing and data management.

**User Devices**: Smartphones or computers for users to access the system.

### 3.6.2 Software Requirements

**Operating System:** Compatible with major operating systems like Windows, macOS, and Linux.

**Programming Languages:** Python for backend development.

Libraries and Frameworks:

Numpy==1.20.3

Pillow==8.4.0

Keras==2.9.0

TensorFlow==2.9.0

Flask==2.1.2

Pandas==1.4.3

Matplotlib==3.4.3

# 4.EXPERIMENTAL INVESTIGATIONS

## 4.1 Data Collection

The project relies on a dataset of food images with annotated nutritional information. This dataset is used to train the deep learning models. Images are preprocessed to standardize their size and format, ensuring consistency in model training and evaluation.

## 4.2 Model Training

Convolutional neural networks (CNNs) are employed for image recognition. The models are trained using the collected dataset, with techniques such as data augmentation applied to improve model generalization. Hyperparameters such as learning rate, batch size, and the number of epochs are tuned to optimize performance.

## 4.3 Performance Evaluation

The trained models are evaluated using metrics such as accuracy, precision, recall, and F1score. Cross-validation is performed to ensure the robustness of the models. The results are analyzed to identify areas for improvement and further optimization.

**4.4 Data Preprocessing**:

Annotate the images with labels indicating the type of food and its calorie content. Perform data augmentation techniques such as rotation, scaling, and flipping to increase the dataset size and variability.

**4.5 Model Development**:

Choose a suitable deep learning architecture for image recognition and calorie estimation, such as Convolutional Neural Networks (CNNs).

Implement the model using frameworks like TensorFlow or PyTorch.Split the dataset into training, validation, and test sets.

**4.6 Model Training**:

Train the model on the training set, optimizing it using techniques like transfer learning and fine-tuning.

Monitor the model's performance using metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

**4.7 Model Evaluation**:

Evaluate the model's accuracy and generalization ability on the validation and test sets.

Conduct cross-validation to ensure the robustness of the model.

**4.8 Implementation**:

Develop a user-friendly application or system that utilizes the trained model for calorie estimation.

Integrate the system into a food waste management workflow, such as in restaurants or households.

**4.9 Testing in Real-world Setting**:

Deploy the system in a real-world environment to estimate the calories of food waste.

Collect feedback and refine the system based on user experience and accuracy.

Analyze the estimated calorie data to identify patterns in food waste.

Provide insights and recommendations for reducing food waste based on the analysis.

## Expected outcomes

1. **Accurate Calorie Estimation**: A reliable deep learning model capable of accurately estimating the calorie content of food items from images.
2. **Data-Driven Insights**: Detailed analysis of food waste patterns and recommendations for minimizing waste.
3. **Improved Food Waste Management**: A practical tool for restaurants, households, and other entities to manage and reduce food waste.
4. **Sustainability Promotion**: Enhanced awareness and practices for sustainable food consumption and waste management.

# 5.IMPLEMENTATION

## Environment Setup

The project environment is set up using the following dependencies:

**NumPy:** For numerical computations (pip install numpy==1.20.3)

**Pillow:** For image processing (pip install Pillow==8.4.0)

**Keras:** For building deep learning models (pip install Keras==2.9.0)

**TensorFlow:** Backend for Keras (pip install tensorflow==2.9.0) **Flask:** Web application framework (pip install Flask==2.1.2) **pandas:** For data manipulation (pip install pandas==1.4.3) **matplotlib:** For data visualization (pip install matplotlib==3.4.3)

## Code Implementation

The implementation involves developing the backend server using Flask, which handles user authentication, image uploads, and model inference. The frontend is designed to be intuitive, allowing users to easily upload images and view the nutrition reports.

The deep learning models are implemented using Keras and TensorFlow. The models are trained on a high-performance computing setup to handle the computational demands. Once trained, the models are integrated into the Flask application for real-time inference.

**app.py:**

import tensorflow from flask import Flask,flash, request, render\_template import csv import math

import os

import numpy as np

from tensorflow.keras.preprocessing import image from tensorflow.python.keras.models import load\_model from werkzeug.utils import secure\_filename import tensorflow as tf from tensorflow.keras.layers import BatchNormalization tmpl\_dir = os.path.join(os.path.dirname(os.path.abspath(\_file\_)), 'templates') app = Flask(\_name\_, template\_folder=tmpl\_dir)

UPLOAD\_FOLDER = 'static/uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# define label meaning label = ['apple pie:Estimate Calories 237 For 100 Grams Quantity',

'baby back ribs:Estimate Calories 276 For 100 Grams Quantity',

'baklava:Estimate Calories 403 For 100 Grams Quantity',

'beef carpaccio:Estimate Calories 231 For 100 Grams Quantity',

'beef tartare:Estimate Calories 246 For 100 Grams Quantity',

'beet salad:Estimate Calories 231 For 100 Grams Quantity',

'beignets:Estimate Calories 291 For 100 Grams Quantity',

'bibimbap:Estimate Calories 113 For 100 Grams Quantity',

'bread pudding:Estimate Calories 188 For 100 Grams Quantity',

'breakfast burrito:Estimate Calories 169 For 100 Grams Quantity',

'bruschetta:Estimate Calories 206 For 100 Grams Quantity',

'caesar salad:Estimate Calories 158 For 100 Grams Quantity',

'cannoli:Estimate Calories 293 For 100 Grams Quantity',

'caprese salad:Estimate Calories 136 For 100 Grams Quantity',

'carrot cake:Estimate Calories 333 For 100 Grams Quantity', 'ceviche:Estimate Calories 68 For 100 Grams Quantity',

'cheese plate:Estimate Calories 389 For 100 Grams Quantity',

'cheesecake:Estimate Calories 321 For 100 Grams Quantity',

'chicken curry:Estimate Calories 104 For 100 Grams Quantity',

'chicken quesadilla:Estimate Calories 216 For 100 Grams Quantity',

'chicken wings:Estimate Calories 328 For 100 Grams Quantity',

'chocolate cake:Estimate Calories 389 For 100 Grams Quantity',

'chocolate mousse:Estimate Calories 225 For 100 Grams Quantity',

'churros:Estimate Calories 396 For 100 Grams Quantity',

'clam chowder:Estimate Calories 79 For 100 Grams Quantity',

'club sandwich:Estimate Calories 234 For 100 Grams Quantity',

'crab cakes:Estimate Calories 173 For 100 Grams Quantity',

'creme brulee:Estimate Calories 343 For 100 Grams Quantity',

'croque madame:Estimate Calories 199 For 100 Grams Quantity',

'cup cakes:Estimate Calories 389 For 100 Grams Quantity',

'deviled eggs:Estimate Calories 225 For 100 Grams Quantity',

'donuts:Estimate Calories 421 For 100 Grams Quantity',

'dumplings:Estimate Calories 230 For 100 Grams Quantity',

'edamame:Estimate Calories 121 For 100 Grams Quantity',

'eggs benedict:Estimate Calories 260 For 100 Grams Quantity',

'escargots:Estimate Calories 90 For 100 Grams Quantity',

'falafel:Estimate Calories 416 For 100 Grams Quantity',

'filet mignon:Estimate Calories 267 For 100 Grams Quantity',

'fish and\_chips:Estimate Calories 134 For 100 Grams Quantity',

'foie gras:Estimate Calories 462 For 100 Grams Quantity',

'french fries:Estimate Calories 312 For 100 Grams Quantity',

'french onion soup:Estimate Calories 137 For 100 Grams Quantity',

'french toast:Estimate Calories 263 For 100 Grams Quantity',

'fried calamari:Estimate Calories 249 For 100 Grams Quantity',

'fried rice:Estimate Calories 174 For 100 Grams Quantity',

'frozen yogurt:Estimate Calories 127 For 100 Grams Quantity',

'garlic bread:Estimate Calories 350 For 100 Grams Quantity',

'gnocchi:Estimate Calories 201 For 100 Grams Quantity',

'greek salad:Estimate Calories 113 For 100 Grams Quantity',

'grilled cheese sandwich:Estimate Calories 344 For 100 Grams Quantity',

'grilled salmon:Estimate Calories 206 For 100 Grams Quantity',

'guacamole:Estimate Calories 151 For 100 Grams Quantity', 'gyoza:Estimate Calories 211 For 100 Grams Quantity',

'hamburger:Estimate Calories 239 For 100 Grams Quantity',

'hot and sour soup:Estimate Calories 39 For 100 Grams Quantity',

'hot dog:Estimate Calories 322 For 100 Grams Quantity',

'huevos rancheros:Estimate Calories 143 For 100 Grams Quantity',

'hummus:Estimate Calories 166 For 100 Grams Quantity',

'ice cream:Estimate Calories 207 For 100 Grams Quantity',

'lasagna:Estimate Calories 156 For 100 Grams Quantity',

'lobster bisque:Estimate Calories 106 For 100 Grams Quantity',

'lobster roll sandwich:Estimate Calories 199 For 100 Grams Quantity',

'macaroni and cheese:Estimate Calories 190 For 100 Grams Quantity',

'macarons:Estimate Calories 384 For 100 Grams Quantity',

'miso soup:Estimate Calories 24 For 100 Grams Quantity',

'mussels:Estimate Calories 172 For 100 Grams Quantity',

'nachos:Estimate Calories 224 For 100 Grams Quantity',

'omelette:Estimate Calories 181 For 100 Grams Quantity',

'onion rings:Estimate Calories 356 For 100 Grams Quantity',

'oysters:Estimate Calories 163 For 100 Grams Quantity',

'pad thai:Estimate Calories 170 For 100 Grams Quantity',

'paella:Estimate Calories 183 For 100 Grams Quantity',

'pancakes:Estimate Calories 227 For 100 Grams Quantity',

'panna cotta:Estimate Calories 319 For 100 Grams Quantity',

'peking duck:Estimate Calories 241 For 100 Grams Quantity',

'pho:Estimate Calories 90 For 100 Grams Quantity',

'pizza:Estimate Calories 266 For 100 Grams Quantity',

'pork chop:Estimate Calories 209 For 100 Grams Quantity',

'poutine:Estimate Calories 222 For 100 Grams Quantity',

'prime rib:Estimate Calories 341 For 100 Grams Quantity',

'pulled pork sandwich:Estimate Calories 175 For 100 Grams Quantity',

'ramen:Estimate Calories 135 For 100 Grams Quantity',

'ravioli:Estimate Calories 179 For 100 Grams Quantity',

'red velvet cake:Estimate Calories 337 For 100 Grams Quantity',

'risotto:Estimate Calories 122 For 100 Grams Quantity',

'samosa:Estimate Calories 261 For 100 Grams Quantity',

'sashimi:Estimate Calories 124 For 100 Grams Quantity',

'scallops:Estimate Calories 111 For 100 Grams Quantity',

'seaweed salad:Estimate Calories 115 For 100 Grams Quantity',

'shrimp and grits:Estimate Calories 149 For 100 Grams Quantity',

'spaghetti bolognese:Estimate Calories 101 For 100 Grams Quantity',

'spaghetti carbonara:Estimate Calories 199 For 100 Grams Quantity',

'spring rolls:Estimate Calories 230 For 100 Grams Quantity',

'steak:Estimate Calories 278 For 100 Grams Quantity',

'strawberry shortcake:Estimate Calories 172 For 100 Grams Quantity',

'sushi:Estimate Calories 165 For 100 Grams Quantity',

'tacos:Estimate Calories 206 For 100 Grams Quantity',

'takoyaki:Estimate Calories 149 For 100 Grams Quantity',

'tiramisu:Estimate Calories 329 For 100 Grams Quantity',

'tuna tartare:Estimate Calories 176 For 100 Grams Quantity',

'waffles:Estimate Calories 291 For 100 Grams Quantity']

### nu\_link = '[https://www.nutritionix.com/food/'](https://www.nutritionix.com/food/)

# Loading the best saved model to make predictions. tensorflow.keras.backend.clear\_session() model = tf.keras.models.load\_model('food.h5') print('model successfully loaded!')

start = [0] passed = [0] pack = [[]] num = [0]

nutrients = [

{'name': 'protein', 'value': 0.0},

{'name': 'calcium', 'value': 0.0},

{'name': 'fat', 'value': 0.0},

{'name': 'carbohydrates', 'value': 0.0},

{'name': 'vitamins', 'value': 0.0}

]

with open('nutrition101.csv', 'r') as file: reader = csv.reader(file) nutrition\_table = dict() for i, row in enumerate(reader):

if i == 0: name = '' continue

else:

name = row[1].strip() nutrition\_table[name] = [

{'name': 'protein', 'value': float(row[2])},

{'name': 'calcium', 'value': float(row[3])},

{'name': 'fat', 'value': float(row[4])},

{'name': 'carbohydrates', 'value': float(row[5])},

{'name': 'vitamins', 'value': float(row[6])}

]

@app.route("/") @app.route("/index") def index():

return render\_template('index.html')

@app.route("/login") def login():

return render\_template('login.html')

@app.route("/chart") def chart():

return render\_template('chart.html') @app.route('/recognize') def recognize():

return render\_template('recognize.html')

@app.route('/upload', methods=['POST']) def upload():

file = request.files.getlist("img") for f in file:

filename = secure\_filename(str(num[0] + 500) + '.jpg') num[0] += 1 name = os.path.join(app.config['UPLOAD\_FOLDER'], filename) print('save name', name) f.save(name)

pack[0] = []

return render\_template('recognize.html', img=file)

@app.route('/predict') def predict(): result = [] # pack = [] print('total image', num[0]) for i in range(start[0], num[0]):

pa = dict()

filename = f'{UPLOAD\_FOLDER}/{i + 500}.jpg' print('image filepath', filename)

pred\_img = filename pred\_img = image.load\_img(pred\_img, target\_size=(128, 128)) pred\_img = image.img\_to\_array(pred\_img) pred\_img = np.expand\_dims(pred\_img, axis=0) pred\_img = pred\_img / 255.

pred = model.predict(pred\_img) print("Pred") print(pred)

if math.isnan(pred[0][0]) and math.isnan(pred[0][1]) and \ math.isnan(pred[0][2]) and math.isnan(pred[0][3]):

pred = np.array([0.05, 0.05, 0.05, 0.07, 0.09, 0.19, 0.55, 0.0, 0.0, 0.0, 0.0])

top = pred.argsort()[0][-3:] label.sort() \_true = label[top[2]] \_trues = label[top[2]] print(\_trues)

pa['image'] = f'{UPLOAD\_FOLDER}/{i + 500}.jpg' x = dict() x[\_true] = float("{:.2f}".format(pred[0][top[2]] \* 100)) print(x[\_true]) x[label[top[1]]] = float("{:.2f}".format(pred[0][top[1]] \* 100)) print(x[label[top[1]]])

x[label[top[0]]] = float("{:.2f}".format(pred[0][top[0]] \* 100))

pa['result'] = x print(x) pa['nutrition'] = nutrition\_table[\_true] pa['food'] = f'{nu\_link}{\_true}' pa['idx'] = i - start[0] pa['quantity'] = 100

pack[0].append(pa) passed[0] += 1

start[0] = passed[0] print('successfully packed')

# compute the average source of calories for p in pack[0]:

nutrients[0]['value'] = (nutrients[0]['value'] + p['nutrition'][0]['value']) nutrients[1]['value'] = (nutrients[1]['value'] + p['nutrition'][1]['value']) nutrients[2]['value'] = (nutrients[2]['value'] + p['nutrition'][2]['value']) nutrients[3]['value'] = (nutrients[3]['value'] + p['nutrition'][3]['value']) nutrients[4]['value'] = (nutrients[4]['value'] + p['nutrition'][4]['value'])

nutrients[0]['value'] = nutrients[0]['value'] / num[0] nutrients[1]['value'] = nutrients[1]['value'] / num[0] nutrients[2]['value'] = nutrients[2]['value'] / num[0] nutrients[3]['value'] = nutrients[3]['value'] / num[0] nutrients[4]['value'] = nutrients[4]['value'] / num[0]

return render\_template('results.html', pack=pack[0], whole\_nutrition=nutrients, prediction = \_trues)

@app.route('/update', methods=['POST']) def update(): return render\_template('index.html', img='static/P2.jpg') if \_name\_ == "\_main\_":

import click

@click.command()

@click.option('--debug', is\_flag=True)

@click.option('--threaded', is\_flag=True)

@click.argument('HOST', default='127.0.0.1') @click.argument('PORT', default=5000, type=int) def run(debug, threaded, host, port):

"""

This function handles command line parameters. Run the server using python server.py Show the help text using python server.py --help

"""

HOST, PORT = host, port app.run(host=HOST, port=PORT, debug=debug, threaded=threaded) run()

-----------------------------------------------------------------------------

get\_nutrition\_data.py:

import requests import pandas as pd

def get\_nutrition(food\_name):

nutrition\_data = pd.DataFrame(columns=['name', 'protein', 'calcium', 'fat', 'carbohydrates',

'vitamins']) for name in food\_name: url

["https://api.nal.usda.gov/fdc/v1/foods/search?api\_key=d4D6dSOc81pTAOY2gsNZ0YhjkMlhSt](https://api.nal.usda.gov/fdc/v1/foods/search?api_key=d4D6dSOc81pTAOY2gsNZ0YhjkMlhStLJRoII5SJu&query=)

### [LJRoII5SJu&query="](https://api.nal.usda.gov/fdc/v1/foods/search?api_key=d4D6dSOc81pTAOY2gsNZ0YhjkMlhStLJRoII5SJu&query=) + name

response = requests.get(url) data = response.json() flatten\_json = pd.json\_normalize(data["foods"]) first\_food = flatten\_json.iloc[0] first\_food\_nutrition\_list = first\_food.foodNutrients for item in first\_food\_nutrition\_list: if item['nutrientNumber'] == "203":

protein = item['value'] continue if item['nutrientNumber'] == "301":

calcium = item['value'] continue if item['nutrientNumber'] == "204":

fat = item['value'] continue if item['nutrientNumber'] == "205":

carbs = item['value'] continue if item['nutrientNumber'] == "318":

vitamin\_a = item['value'] continue if item['nutrientNumber'] == "401":

vitamin\_c = item['value']

continue

vitamins = float(vitamin\_a) + float(vitamin\_c) print(name)

nutrition\_data = nutrition\_data.append({

'name': name,

'protein': protein,

'calcium': calcium / 1000,

'fat': fat,

'carbohydrates': carbs,

'vitamins': vitamins / 1000

}, ignore\_index=True)

return nutrition\_data

nutrition101 = get\_nutrition(['apple pie:Estimate Calories 237 For 100 Grams Quantity',

'baby back ribs:Estimate Calories 276 For 100 Grams Quantity',

'baklava:Estimate Calories 403 For 100 Grams Quantity',

'beef carpaccio:Estimate Calories 231 For 100 Grams Quantity',

'beef tartare:Estimate Calories 246 For 100 Grams Quantity',

'beet salad:Estimate Calories 231 For 100 Grams Quantity',

'beignets:Estimate Calories 291 For 100 Grams Quantity',

'bibimbap:Estimate Calories 113 For 100 Grams Quantity',

'bread pudding:Estimate Calories 188 For 100 Grams Quantity',

'breakfast burrito:Estimate Calories 169 For 100 Grams Quantity',

'bruschetta:Estimate Calories 206 For 100 Grams Quantity',

'caesar salad:Estimate Calories 158 For 100 Grams Quantity',

'cannoli:Estimate Calories 293 For 100 Grams Quantity',

'caprese salad:Estimate Calories 136 For 100 Grams Quantity',

'carrot cake:Estimate Calories 333 For 100 Grams Quantity',

'ceviche:Estimate Calories 68 For 100 Grams Quantity',

'cheese plate:Estimate Calories 389 For 100 Grams Quantity',

'cheesecake:Estimate Calories 321 For 100 Grams Quantity',

'chicken curry:Estimate Calories 104 For 100 Grams Quantity',

'chicken quesadilla:Estimate Calories 216 For 100 Grams Quantity',

'chicken wings:Estimate Calories 328 For 100 Grams Quantity',

'chocolate cake:Estimate Calories 389 For 100 Grams Quantity',

'chocolate mousse:Estimate Calories 225 For 100 Grams Quantity',

'churros:Estimate Calories 396 For 100 Grams Quantity',

'clam chowder:Estimate Calories 79 For 100 Grams Quantity',

'club sandwich:Estimate Calories 234 For 100 Grams Quantity',

'crab cakes:Estimate Calories 173 For 100 Grams Quantity',

'creme brulee:Estimate Calories 343 For 100 Grams Quantity',

'croque madame:Estimate Calories 199 For 100 Grams Quantity',

'cup cakes:Estimate Calories 389 For 100 Grams Quantity',

'deviled eggs:Estimate Calories 225 For 100 Grams Quantity',

'donuts:Estimate Calories 421 For 100 Grams Quantity',

'dumplings:Estimate Calories 230 For 100 Grams Quantity',

'edamame:Estimate Calories 121 For 100 Grams Quantity',

'eggs benedict:Estimate Calories 260 For 100 Grams Quantity',

'escargots:Estimate Calories 90 For 100 Grams Quantity',

'falafel:Estimate Calories 416 For 100 Grams Quantity',

'filet mignon:Estimate Calories 267 For 100 Grams Quantity',

'fish and\_chips:Estimate Calories 134 For 100 Grams Quantity', 'foie gras:Estimate Calories 462 For 100 Grams Quantity',

'french fries:Estimate Calories 312 For 100 Grams Quantity',

'french onion soup:Estimate Calories 137 For 100 Grams Quantity',

'french toast:Estimate Calories 263 For 100 Grams Quantity',

'fried calamari:Estimate Calories 249 For 100 Grams Quantity',

'fried rice:Estimate Calories 174 For 100 Grams Quantity',

'frozen yogurt:Estimate Calories 127 For 100 Grams Quantity',

'garlic bread:Estimate Calories 350 For 100 Grams Quantity',

'gnocchi:Estimate Calories 201 For 100 Grams Quantity',

'greek salad:Estimate Calories 113 For 100 Grams Quantity',

'grilled cheese sandwich:Estimate Calories 344 For 100 Grams Quantity',

'grilled salmon:Estimate Calories 206 For 100 Grams Quantity',

'guacamole:Estimate Calories 151 For 100 Grams Quantity', 'gyoza:Estimate Calories 211 For 100 Grams Quantity',

'hamburger:Estimate Calories 239 For 100 Grams Quantity',

'hot and sour soup:Estimate Calories 39 For 100 Grams Quantity',

'hot dog:Estimate Calories 322 For 100 Grams Quantity',

'huevos rancheros:Estimate Calories 143 For 100 Grams Quantity',

'hummus:Estimate Calories 166 For 100 Grams Quantity',

'ice cream:Estimate Calories 207 For 100 Grams Quantity',

'lasagna:Estimate Calories 156 For 100 Grams Quantity',

'lobster bisque:Estimate Calories 106 For 100 Grams Quantity',

'lobster roll sandwich:Estimate Calories 199 For 100 Grams Quantity',

'macaroni and cheese:Estimate Calories 190 For 100 Grams Quantity',

'macarons:Estimate Calories 384 For 100 Grams Quantity',

'miso soup:Estimate Calories 24 For 100 Grams Quantity',

'mussels:Estimate Calories 172 For 100 Grams Quantity',

'nachos:Estimate Calories 224 For 100 Grams Quantity',

'omelette:Estimate Calories 181 For 100 Grams Quantity',

'onion rings:Estimate Calories 356 For 100 Grams Quantity',

'oysters:Estimate Calories 163 For 100 Grams Quantity',

'pad thai:Estimate Calories 170 For 100 Grams Quantity',

'paella:Estimate Calories 183 For 100 Grams Quantity',

'pancakes:Estimate Calories 227 For 100 Grams Quantity',

'panna cotta:Estimate Calories 319 For 100 Grams Quantity',

'peking duck:Estimate Calories 241 For 100 Grams Quantity',

'pho:Estimate Calories 90 For 100 Grams Quantity',

'pizza:Estimate Calories 266 For 100 Grams Quantity',

'pork chop:Estimate Calories 209 For 100 Grams Quantity',

'poutine:Estimate Calories 222 For 100 Grams Quantity',

'prime rib:Estimate Calories 341 For 100 Grams Quantity',

'pulled pork sandwich:Estimate Calories 175 For 100 Grams Quantity',

'ramen:Estimate Calories 135 For 100 Grams Quantity',

'ravioli:Estimate Calories 179 For 100 Grams Quantity',

'red velvet cake:Estimate Calories 337 For 100 Grams Quantity',

'risotto:Estimate Calories 122 For 100 Grams Quantity',

'samosa:Estimate Calories 261 For 100 Grams Quantity',

'sashimi:Estimate Calories 124 For 100 Grams Quantity',

'scallops:Estimate Calories 111 For 100 Grams Quantity',

'seaweed salad:Estimate Calories 115 For 100 Grams Quantity',

'shrimp and grits:Estimate Calories 149 For 100 Grams Quantity',

'spaghetti bolognese:Estimate Calories 101 For 100 Grams Quantity',

'spaghetti carbonara:Estimate Calories 199 For 100 Grams Quantity',

'spring rolls:Estimate Calories 230 For 100 Grams Quantity',

'steak:Estimate Calories 278 For 100 Grams Quantity',

'strawberry shortcake:Estimate Calories 172 For 100 Grams Quantity',

'sushi:Estimate Calories 165 For 100 Grams Quantity',

'tacos:Estimate Calories 206 For 100 Grams Quantity', 'takoyaki:Estimate Calories 149 For 100 Grams Quantity',

'tiramisu:Estimate Calories 329 For 100 Grams Quantity',

'tuna tartare:Estimate Calories 176 For 100 Grams Quantity',

'waffles:Estimate Calories 291 For 100 Grams Quantity']

)

nutrition101 = nutrition101.reset\_index(drop=True) nutrition101.to\_csv("nutrition101.csv")

-----------------------------------------------------------------------------

# 6.RESULTS

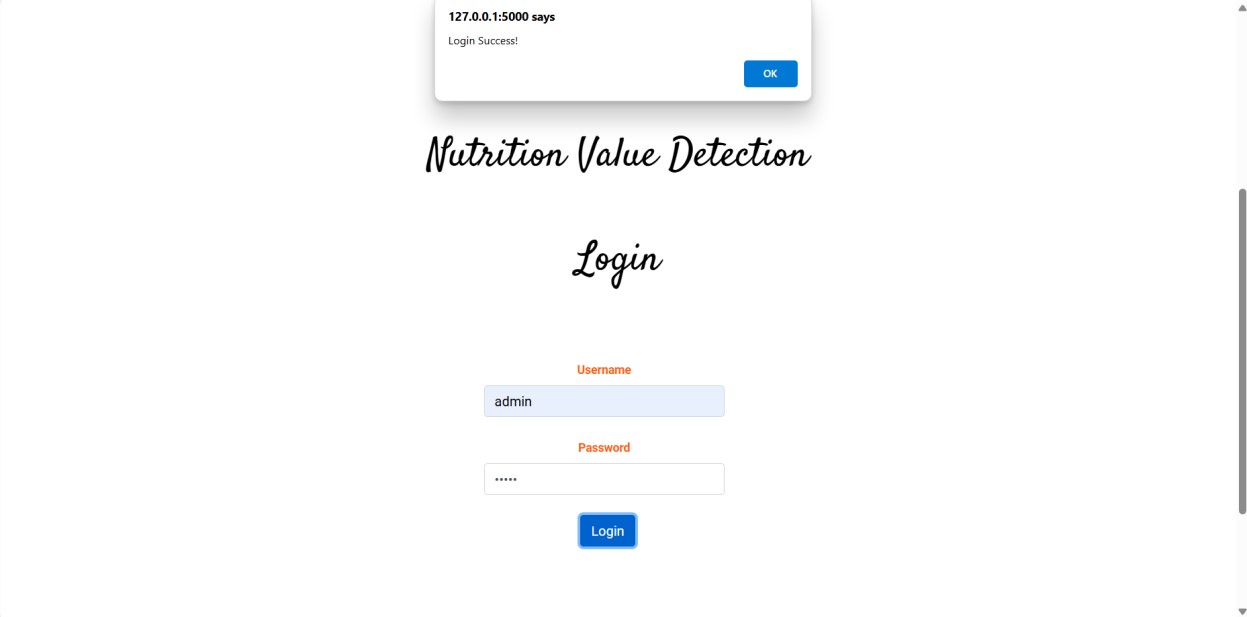
**6.1 Result**

## 6.1.1 Login



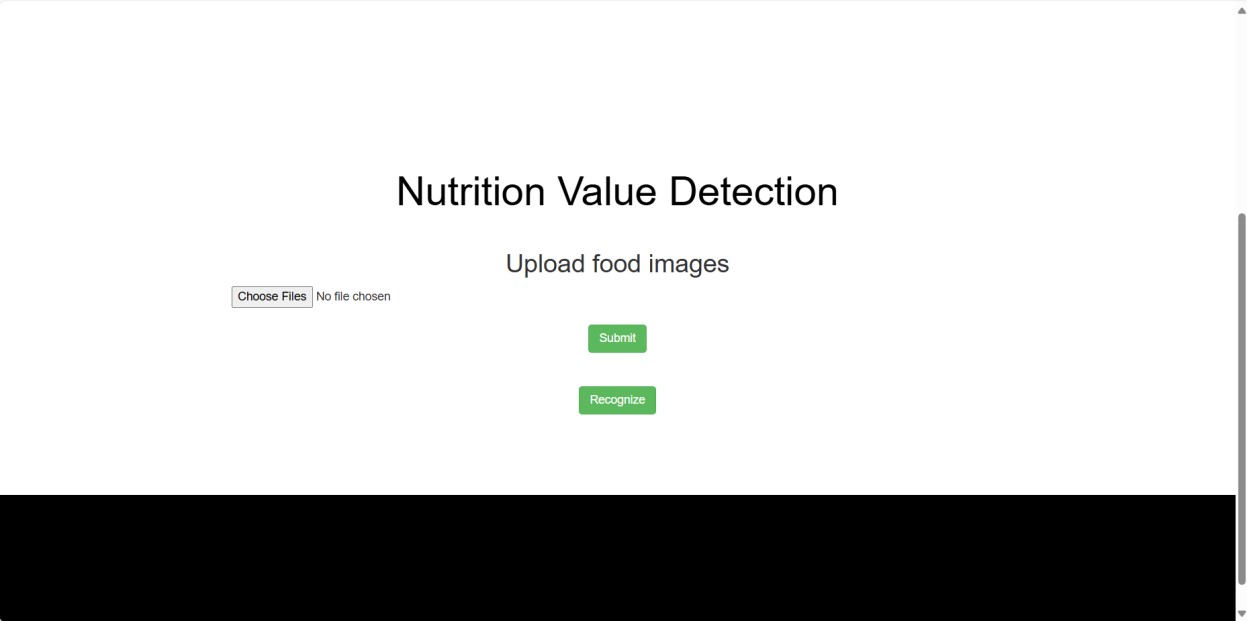
Users need to log in to obtain the nutritional value of the food item.

## 6.1.2 Successful Login



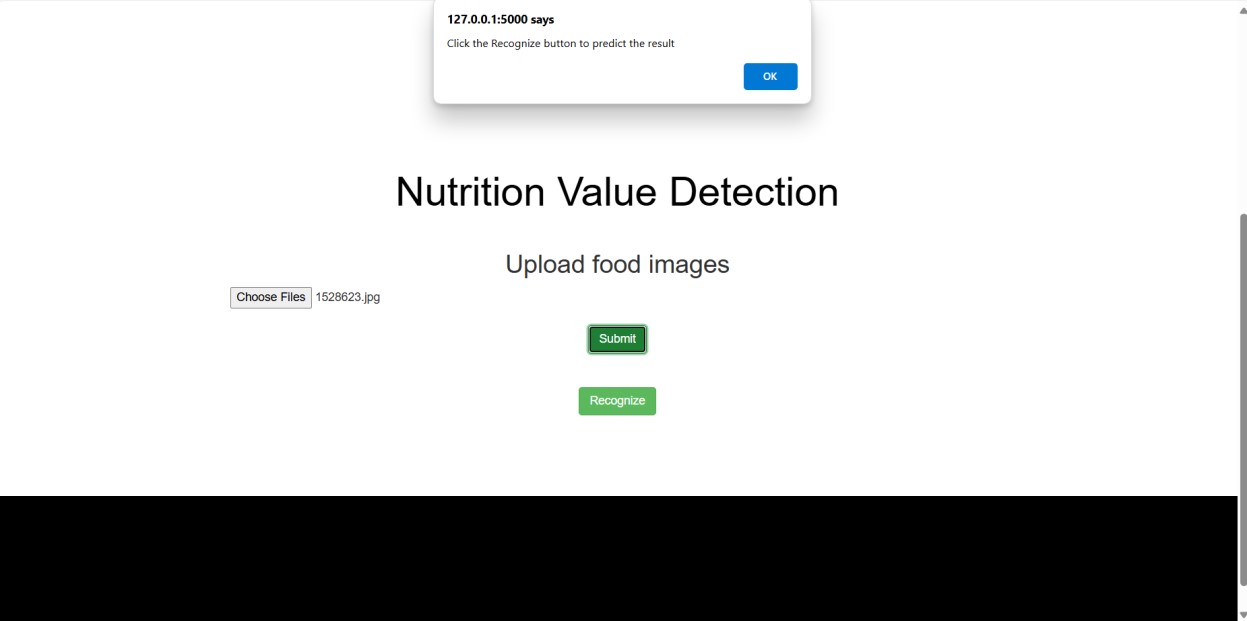
The user will receive a pop-up message confirming a successful login.

## 6.1.3 Choosing a File



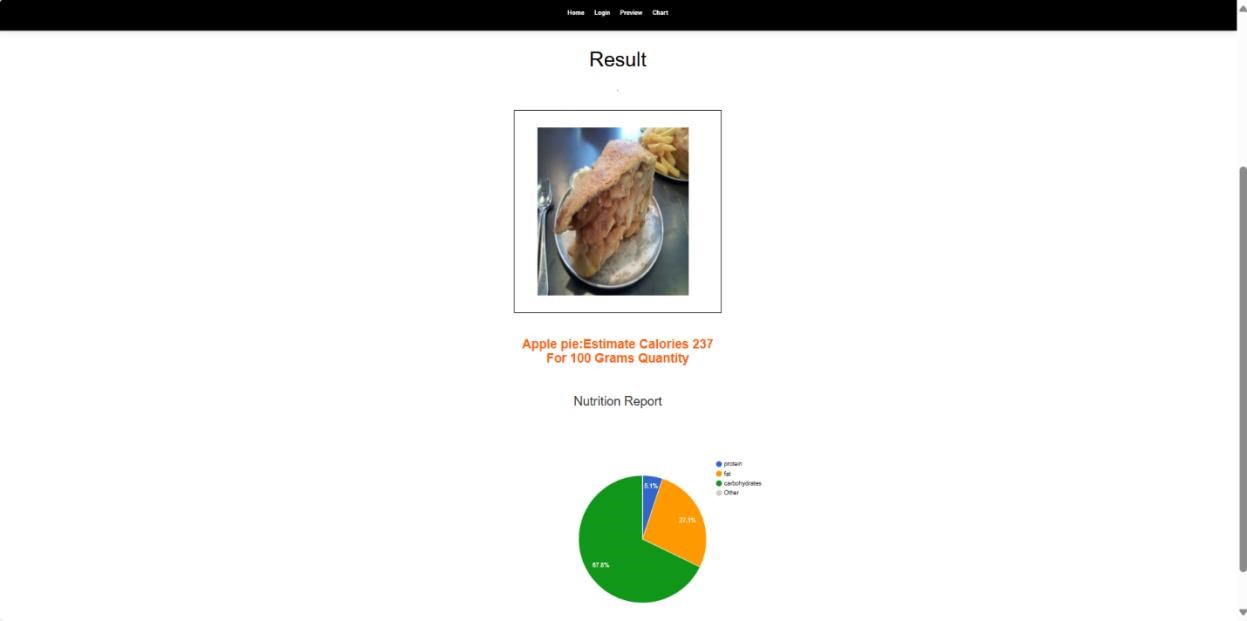
The user must choose a file.

## 6.1.4 Submitting the chosen file



After choosing a file, the user has to select “submit”. The user will receive a pop-up message confirming a successful submission and asking them to select “recognize”.

## 6.1.5 After Recognizing the chosen file



After the user selects “recognize”, they will get a nutrition report as shown above.

# 7.CONCLUSION AND FUTURE SCOPE

The Nutrition Value Detection System provides an efficient and user-friendly solution for tracking nutritional intake. By leveraging deep learning and image processing, it offers accurate and automated nutritional analysis. The system's design ensures ease of use, scalability, and accessibility, making it a valuable tool for a wide range of users. The successful implementation of our Nutrition Value Detection project marks a significant stride in integrating deep learning technology with everyday health and wellness practices. By leveraging the power of advanced image recognition techniques, we have created a user-friendly platform that simplifies the process of dietary monitoring. This project not only highlights the potential of artificial intelligence in health-related applications but also demonstrates the practical utility of machine learning models in providing accurate nutritional assessments.

**Enhanced Accuracy**: Improve the accuracy of nutritional estimations by incorporating more advanced deep learning models and larger, more diverse datasets.Expanded

**Database**: Expand the food image database to include a wider variety of food items from different cuisines and dietary preferences.

Real-Time Analysis: Develop real-time image analysis capabilities to provide immediate feedback on nutritional content during meal preparation.Mobile Application: Create a mobile application version to allow users to capture and analyze food images directly from their smartphones.

Integration with Wearables: Integrate with wearable devices to provide continuous dietary monitoring and personalized nutritional advice.

Dietary Recommendations: Incorporate personalized dietary recommendations based on user health data and nutritional goals.

Voice and AR Interfaces: Develop voice-activated features and augmented reality interfaces to enhance accessibility and user interaction.

Global Language Support: Add support for multiple languages to make the application accessible to non-English speakers.

User Community Features: Implement community features where users can share their food images and nutritional information, fostering a collaborative environment.

Regulatory Compliance: Ensure compliance with relevant health and data privacy regulations to enhance user trust and expand market reach.

# 8.REFERENCES

1. FAO. (2011). Global food losses and food waste – Extent, causes and prevention. Rome.
2. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
3. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. ICLR.
4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition.

CVPR.

1. Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. ECCV.
2. Chollet, F. (2017). Deep Learning with Python. Manning Publications.
3. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. ICLR.