

# **B.Tech. BCSE497J - Project-I**

## **AI Based Recipe Generator**

*Submitted in partial fulfillment of the requirements for the degree of*

### **Bachelor of Technology**

*in*

### **CSE Programme**

*by*

**21BCE3275**

**SAGNIK KOLAY**

**21BCI0161**

**SREERAM MUKKU**

**21BKT0137**

**SHOURYA PANDEY**

**Under the Supervision of**

**Dr. KALAIVANI K**

Assistant Professor Sr. Grade 1

School of Computer Science and Engineering (SCOPE)



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

November 2024

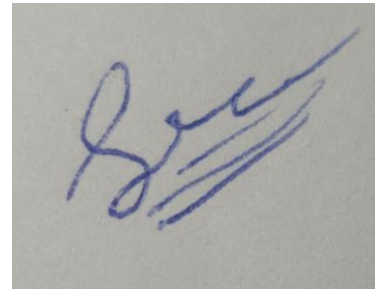
## **DECLARATION**

I hereby declare that the project entitled **AI Based Recipe Generator** submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Prof. / Dr. **Kalaivani K**

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

Place : Vellore

Date : 20.11.2024

A handwritten signature in blue ink, appearing to be 'S. S.', with a stylized flourish at the end.

**Signature of the Candidate**

## **CERTIFICATE**


This is to certify that the project entitled AI Based Recipe Generator submitted by 21BCE3275, 21BCI0161, 21BKT0137, School of Computer Science and Engineering, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him / her under my supervision during Fall Semester 2024-2025, as per the VIT code of academic and research ethics.

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

Place : Vellore

Date : 20.11.2024

  
Signature of the Guide

  
Examiner(s)

**Dr. UMADEVI K S**  
**Bachelors of Technology**

## **ACKNOWLEDGEMENTS**

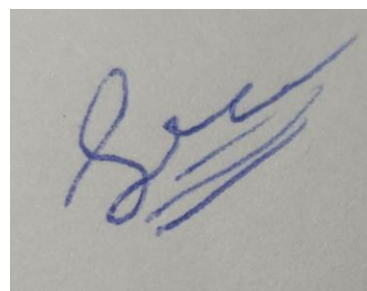
I am deeply grateful to the management of Vellore Institute of Technology (VIT) for providing me with the opportunity and resources to undertake this project. Their commitment to fostering a conducive learning environment has been instrumental in my academic journey. The support and infrastructure provided by VIT have enabled me to explore and develop my ideas to their fullest potential.

My sincere thanks to Dr. Ramesh Babu K, the Dean of the School of Computer Science and Engineering (SCOPE), for his unwavering support and encouragement. His leadership and vision have greatly inspired me to strive for excellence. The Dean's dedication to academic excellence and innovation has been a constant source of motivation for me. I appreciate his efforts in creating an environment that nurtures creativity and critical thinking.

I express my profound appreciation to Dr. Umadevi K S, the Head of the Computer Science Department for her insightful guidance and continuous support. Her expertise and advice have been crucial in shaping the direction of my project. The Head of Department's commitment to fostering a collaborative and supportive atmosphere has greatly enhanced my learning experience. Her constructive feedback and encouragement have been invaluable in overcoming challenges and achieving my project goals.

I am immensely thankful to my project supervisor, Dr. Kalaivani K, for her dedicated mentorship and invaluable feedback. Her patience, knowledge, and encouragement have been pivotal in the successful completion of this project. My supervisor's willingness to share her expertise and provide thoughtful guidance has been instrumental in refining my ideas and methodologies. Her support has not only contributed to the success of this project but has also enriched my overall academic experience.

Thank you all for your contributions and support.



**Name of the Candidate**

## TABLE OF CONTENTS

<Contents, Times New Roman 12, Line spacing 1.5>

Sl.No	Contents	Page No.
	<b>Abstract</b>	<b>vi</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background	<b>1</b>
	1.2 Motivations	<b>2</b>
	1.3 Scope of the Project	<b>2</b>
<b>2.</b>	<b>PROJECT DESCRIPTION AND GOALS</b>	<b>3</b>
	2.1 Literature Review	<b>3</b>
	2.2 Research Gap	<b>18</b>
	2.3 Objectives	<b>18</b>
	2.4 Problem Statement	<b>18</b>
	2.5 Project Plan	<b>19</b>
<b>3.</b>	<b>TECHNICAL SPECIFICATION</b>	<b>20</b>
	3.1 Requirements	<b>20</b>
	3.1.1 Functional	<b>20</b>
	3.1.2 Non-Functional	<b>20</b>
	3.2 Feasibility Study	<b>20</b>
	3.2.1 Technical Feasibility	<b>20</b>
	3.2.2 Economic Feasibility	<b>20</b>
	3.2.2 Social Feasibility	<b>21</b>
	3.3 System Specification	<b>21</b>
	3.3.1 Hardware Specification	<b>21</b>
	3.3.2 Software Specification	<b>21</b>
<b>4.</b>	<b>DESIGN APPROACH AND DETAILS</b>	<b>22</b>
	4.1 System Architecture	<b>22</b>
	4.2 Design	<b>32</b>
	4.2.1 Data Flow Diagram	<b>32</b>
	4.2.2 Use Case Diagram	<b>33</b>
	4.2.3 Class Diagram	<b>34</b>
	4.2.4 Sequence Diagram	<b>35</b>
<b>5.</b>	<b>METHODOLOGY AND TESTING</b>	<b>36</b>

	5.1 Module Description	
	5.2 Testing	
<b>6.</b>	<b>PROJECT DEMONSTRATION</b>	<b>39</b>
<b>7.</b>	<b>RESULT AND DISCUSSION (COST ANALYSIS as applicable)</b>	<b>40</b>
<b>8.</b>	<b>CONCLUSION</b>	<b>41</b>
<b>9.</b>	<b>REFERENCES</b>	<b>42</b>
	<b>APPENDIX A – SAMPLE CODE</b>	<b>45</b>

# ABSTRACT

The rapid advancement of Artificial Intelligence (AI) has permeated multiple domains, including the culinary sector, where AI applications are enabling significant transformations in how food-related tasks are performed. This research presents the development of an AI-based recipe generator designed to automate the process of generating recipes from food images. The system integrates deep learning for food image classification and a robust API to retrieve detailed recipes, making it an innovative solution for personalized meal planning and food recognition. The primary model used in this study is ResNet50, a deep convolutional neural network (CNN) that has been pre-trained on the ImageNet dataset, ensuring robust feature extraction capabilities. The model has been fine-tuned to classify food images into distinct categories by learning visual patterns specific to various dishes, such as fruits, vegetables, and cooked meals.

In conjunction with the classification system, the Spoonacular API is employed to generate recipes corresponding to the predicted dish. The API provides detailed recipe information, including ingredients, instructions, and nutritional information, allowing users to seamlessly convert food images into full-fledged meal plans. This paper discusses the methodology behind the integration of ResNet50 for image classification and the efficient use of the Spoonacular API for recipe retrieval. The research also outlines the experimental setup, evaluation metrics, and performance results, which demonstrate high accuracy in both food classification and recipe generation.

The proposed system showcases the potential for AI to enhance culinary experiences by offering a streamlined solution to meal planning, especially in environments where personalized food recommendations are sought. Furthermore, this work highlights the scalability of AI in solving real-world challenges related to food identification and recipe generation, providing a foundation for future advancements in AI-assisted nutrition and meal personalization. By combining state-of-the-art deep learning techniques with a practical application, this project opens doors for more sophisticated AI-driven culinary tools in the future.

**Keywords:** *Artificial Intelligence, Food Image Classification, ResNet50, Recipe Generation, Deep Learning, Computer Vision.*

# 1. INTRODUCTION

## 1.1 Background

Artificial Intelligence (AI) has revolutionized various industries, transforming traditional practices in healthcare, transportation, and entertainment into more efficient, innovative, and user-centered systems. The food industry, a domain rich in opportunities for technological advancements, has increasingly begun to leverage AI for enhancing customer experiences, optimizing supply chains, and introducing automation. One of the most promising applications of AI in this sector is the creation of systems capable of recognizing food items and generating personalized recipes. Such advancements have the potential to make meal planning more convenient, reduce food waste, and cater to individual dietary needs.

Food image classification, a key component of this application, presents a unique set of challenges. The visual similarity between certain dishes, varying styles of food preparation, and the cultural diversity of cuisines make accurate classification a complex task. However, the emergence of deep learning, particularly convolutional neural networks (CNNs), has provided robust solutions to these challenges. CNN architectures, such as ResNet50, are designed to extract intricate features and patterns from images, enabling high levels of accuracy even in complex classification tasks. ResNet50, with its deep residual learning framework, has proven especially effective in reducing computational bottlenecks and achieving state-of-the-art results in image recognition.

By combining the power of deep learning models like ResNet50 with external resources such as recipe databases or APIs, researchers and developers can bridge the gap between food recognition and recipe generation. This integration allows for the automatic generation of detailed recipes based on visual input, offering innovative solutions for personalized meal planning. Such systems not only enhance user convenience but also pave the way for broader applications, including dietary management, real-time food recognition in smart devices, and cultural enrichment through the exploration of diverse cuisines. As AI continues to evolve, its applications in the food industry hold immense potential for shaping the future of culinary experiences.



## **1.2 Motivation**

The motivation behind this project stems from the growing demand for personalized culinary solutions in a fast-paced world. With the rising popularity of food delivery platforms and meal-planning applications, there is an opportunity to create AI-driven tools that enhance user experiences by providing detailed recipes based on visual inputs. Additionally, such systems can aid in reducing food waste by suggesting recipes for available ingredients and contribute to promoting healthier eating habits through tailored recipe suggestions. The project also explores the technical challenge of building a robust model capable of recognizing diverse food items and connecting the results to relevant recipes in an efficient and user-friendly manner.

## **1.3 Scope of the Project**

The scope of this research focuses on the development of an AI-based recipe generator that combines food image classification with automated recipe retrieval, demonstrating practical applications in meal planning and culinary innovation. The system employs the ResNet50 deep learning model, a state-of-the-art convolutional neural network, to accurately classify a variety of food images into predefined categories. Based on these classifications, it retrieves detailed recipes, including ingredients and preparation steps, using external resources, providing users with an end-to-end solution for culinary exploration.

The scope extends beyond image classification to explore broader applications, such as real-time food recognition for mobile or smart kitchen devices, ingredient-level detection for enhanced recipe personalization, and dietary customizations to accommodate specific needs, such as gluten-free, vegan, or low-calorie diets. Additionally, the project framework is designed to integrate with meal-planning platforms, grocery systems, or food delivery services, making it adaptable for use in real-world scenarios. Future expansions could include the development of proprietary recipe datasets, cultural adaptations to include diverse cuisines, and enhancements to improve classification accuracy and user interaction. This research not only showcases the potential of AI in addressing modern challenges in food recognition and recipe generation but also lays the foundation for more sophisticated systems that promote personalized meal planning, reduce food waste, and contribute to healthier dietary practices.

## 2. PROJECT DESCRIPTION AND GOALS

### 2.1 Literature Review

**2.1.1 Zahisham, Zharfan, Chin Poo Lee, and Kian Ming Lim. “Food recognition with resnet-50.” 2020 IEEE 2<sup>nd</sup> international conference on artificial intelligence in engineering and technology (IICAET). IEEE, 2020.**

The paper “Object Recognition Using ResNet50 and Fine-Tuning Techniques” delves into the growing importance of object recognition in the field of Artificial Intelligence (AI), a domain that has seen remarkable advancements due to the application of Deep Convolutional Neural Networks (DCNNs). The ability of computers to detect and recognize objects has significantly contributed to the rapid development of machine learning. In particular, the paper proposes a framework that leverages the ResNet50 architecture, which is widely regarded for its depth and efficiency in image classification tasks. However, due to the limitations in computational resources, the authors adopt a strategy of using pre-trained weights rather than training the entire model from scratch.

To optimize the performance of the model, the approach involves fine-tuning the pre-trained ResNet50 architecture. Fine-tuning is a common and effective method in machine learning, where a model pre-trained on a large dataset is adapted for a specific task by training only the final layers. This process allows the model to learn task-specific features without requiring extensive computational power for training the entire network. The paper applies this fine-tuning technique to three food-related datasets: ETHZ-FOOD101, UECFOOD100, and UECFOOD256, which are publicly available and commonly used for food image classification tasks.

The evaluation of the proposed method demonstrates its potential in object recognition, particularly in the food domain. The results presented in the paper highlight the effectiveness of the fine-tuned ResNet50 model, showing that it can achieve competitive performance on the selected datasets. Additionally, the paper includes a detailed discussion on the parameter settings used during training, providing insights into the methodology and the performance metrics, which serve to further validate the robustness of the proposed framework.

**2.1.2 Senapati, Biswaranjan, et al. “Transfer learning based models for food detection using ResNet-50.” 2023 IEEE International Conference on Electro Information Technology (eIT). IEEE, 2023.**

The paper *Transfer Learning Based Models for Food Detection Using ResNet-50* explores the application of transfer learning techniques to detect food allergies and analyze nutritional content using food images. The study presents a system designed to identify food allergens through photographs, with a focus on developing a unified framework for detecting, localizing, and classifying food allergies. This system uses powerful algorithms such as ResNet-50, a deep learning model that has been trained on the Food101 dataset. The system is capable of not only identifying food types but also providing detailed nutritional information, making it a comprehensive tool for managing food allergies.

The paper details the use of the Adam and RMSProp optimizers to fine-tune the model, optimizing large weight parameters to improve performance across datasets containing both healthy and allergic food images. ResNet-50 was found to provide the highest accuracy compared to other transfer learning models, achieving a 95% accuracy rate. The proposed method is innovative in its ability to identify various food types while also offering nutritional insights, which could play a significant role in diet management, particularly in preventing adverse reactions related to food allergies. This work demonstrates the potential of transfer learning for the accurate detection of food allergens and the delivery of crucial nutritional information in real-time.

**2.1.3 Foong, Chai C., Goh K. Meng, and Lim L. Tze. “Convolutional neural network based rotten fruit detection using resnet50.” *2021 IEEE 12<sup>th</sup> Control and System Graduate Research Colloquium (ICSGRC)*. IEEE, 2021.**

The paper *Deep Learning Techniques for Rotten Fruit Detection Using CNNs* addresses the importance of ensuring the quality of fruits in the food industry. Fruits and vegetables are essential for a healthy human diet, providing nutrients that help prevent diseases. However, fruits are susceptible to rotting when improperly stored, primarily due to bacterial spread. To maintain quality and safety, the food industry needs efficient inspection methods. The study identifies the challenges in manual inspection, such as inconsistency, lower accuracy, and the high consumption of time and energy.

To overcome these limitations, the authors propose a deep learning-based approach using Convolutional Neural Networks (CNNs) for feature extraction and classification of rotten fruits. This method automates the detection and classification process, significantly enhancing accuracy and efficiency. The study focuses on three fruit types—bananas, apples, and oranges—demonstrating the effectiveness of CNNs in distinguishing between fresh and rotten fruits. The model achieved a high validation accuracy of 98.89%, showcasing its reliability for industrial applications.

The training phase of the model required a total duration of 212.13 minutes, with the classification of a single fruit image taking approximately 0.2 seconds. These results highlight the practicality of the proposed method for real-time applications in the food industry. By leveraging CNNs, this approach offers a scalable and efficient solution to ensure the quality of fruits before reaching consumers, reducing reliance on manual inspections and minimizing potential health risks.

**2.1.4 Abdallah, Said Elshahat, et al. "Deep learning model based on ResNet-50 for beef quality classification." *Inf. Sci. Lett* 12.1 (2023): 289-297.**

The paper *Food Quality Measurement Using ResNet-50 and GAN-Based Data Augmentation* explores the critical role of food quality assessment in agriculture and industrial applications, focusing on the classification of healthy and rancid beef images. Traditional inspection methods struggle to handle the vast number of images required for building robust deep learning models, particularly in distinguishing between healthy and rancid beef based on surface texture analysis. To address this challenge, the authors propose a deep learning approach leveraging ResNet-50 as a classifier, which demonstrates significant promise in grading and classifying beef images.

The study utilized a limited dataset of 18 images, consisting of 8 healthy and 10 rancid beef samples, sourced from the Laboratory of Food Technology, Faculty of Agriculture, Kafrelsheikh University. Recognizing the limitations of this small dataset for training deep learning models, the authors employed a Generative Adversarial Network (GAN) to augment the dataset, generating 180 synthetic images to enhance the model's training process. The results reveal that the ResNet-50 classifier achieved accuracies of 96.03%, 91.67%, and 88.89% during the training, testing, and validation phases, respectively, showcasing its effectiveness in beef image classification.

Additionally, the paper compares the performance of ResNet-50 with both classical and other deep learning architectures, demonstrating its superior efficiency for this application. The integration of GAN-based augmentation with ResNet-50 not only mitigates the limitations of small datasets but also provides a scalable approach for future applications in food quality measurement. This research highlights the potential of advanced deep learning techniques to revolutionize the accuracy and reliability of automated food inspection systems.

**\*2.1.5 Liu, Yizhe. "Automatic food recognition based on efficientnet and ResNet." *Journal of Physics: Conference Series*. Vol. 2646. No. 1. IOP Publishing, 2023.**

The paper *Automatic Food Recognition Using EfficientNet and ResNet Models* investigates advancements in food image recognition, a challenging field due to the high similarity between subcategories and significant inter-class variability. The primary objective of food recognition is to extract meaningful features from an image and predict its category accurately. Traditional methods relied on manually extracted features such as color, shape, and texture, but these approaches often fell short in practical applications due to their limited feature representation capabilities.

With the advent of convolutional neural networks (CNNs), deep learning-based approaches have significantly improved recognition accuracy and computational efficiency. This paper focuses on constructing food recognition models using two state-of-the-art CNN architectures: EfficientNet and ResNet. Both models were evaluated through extensive experiments conducted on the Food-101 dataset, a benchmark dataset commonly used for food image classification tasks.

The experimental results demonstrate the effectiveness of both models in recognizing various food items with high accuracy. By leveraging the powerful feature extraction capabilities of EfficientNet and ResNet, the proposed methods address the limitations of earlier manual feature-based approaches, providing a robust solution for automatic food recognition. This research highlights the potential of deep learning to advance the field of food recognition, offering practical applications for automated food categorization systems.

**2.1.6 Ng, Yi Sen, et al. "Convolutional neural networks for food image recognition: An experimental study." *proceedings of the 5th international workshop on multimedia assisted dietary management*. 2019.**

The paper *Convolutional Neural Networks for Food Image Recognition: An Experimental Study* explores the challenges and best practices for developing food image recognition systems to support nutrition tracking applications. With the rising global trend of consumer health awareness, there is a growing demand for personalized and cost-effective nutrition analysis powered by convolutional neural networks (CNNs). This study addresses critical aspects of creating a comprehensive solution, including dataset preparation, selecting CNN architectures, and implementing optimization strategies for food classification tasks.

Through a series of experiments, the authors provide actionable recommendations. They suggest that larger networks, such as Xception, are optimal for performance, while a minimum of 300 images per category is necessary for effective training. Image augmentation techniques that retain the original shape of food items are particularly beneficial, and combining multiple augmentation methods further enhances results. The study also observes that dataset balancing strategies are unnecessary for food datasets with an imbalance ratio below 7. Furthermore, higher native image resolution during training improves classifier performance, particularly for networks requiring larger input sizes.

This work provides a practical framework for engineers and researchers to efficiently design and implement CNN-based food recognition systems, offering valuable insights for building scalable and localized nutrition tracking applications.

**2.1.7 Termritthikun, Chakkrit, and Surachet Kanprachar. "Nu-ResNet: Deep Residual Networks for Thai Food Image Recognition." *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 10, no. 1-4, 2018, pp. 29-33.**

The paper *Nu-ResNet: Deep Residual Networks for Thai Food Image Recognition* introduces a novel approach to improving the accuracy of convolutional neural networks (CNNs) for food image recognition, specifically Thai food. While increasing the depth of a network by adding more modules can enhance its performance, this approach does not always guarantee better accuracy. Moreover, deeper networks significantly increase parameter size and processing time, which is a major limitation for devices with restricted computational resources, such as smartphones. To address these challenges, the authors incorporate identity mapping from residual networks (ResNet) into their design and propose the NU-ResNet architecture.

The study evaluates NU-ResNet with depths of 4, 8, and 12 on the THFOOD-50 dataset, which contains images of 50 iconic Thai dishes. Results indicate that the NU-ResNet with a depth of 4 achieves a Top-1 accuracy of 83.07% and a Top-5 accuracy of 97.04%. Notably, the parameter size of the model is just 1.48 million, making it lightweight and suitable for smartphone applications. Additionally, the average processing time per image is 44.60 milliseconds, demonstrating its practicality for real-time food recognition.

The findings highlight NU-ResNet's ability to balance accuracy, efficiency, and computational demands, making it a promising solution for Thai food image recognition applications on resource-constrained devices like smartphones.



**2.1.8 Phiphitphatphaisit, Sirawan, and Olarik Surinta. "Deep Feature Extraction Technique Based on Conv1D and LSTM Network for Food Image Recognition." *Engineering & Applied Science Research*, vol. 48, no. 5, 2021.**

The paper *Deep Feature Extraction Technique Based on Conv1D and LSTM Network for Food Image Recognition* addresses the rising global emphasis on health awareness and the importance of selecting food wisely for a healthier lifestyle. In the context of food image recognition, mobile-captured images often include noise such as non-food objects (e.g., people, cutlery, or decorations), which hinders system performance. To overcome this challenge, the study proposes a hybrid deep learning architecture combining ResNet50 and Conv1D-LSTM to enhance the accuracy and efficiency of food image recognition systems.

The methodology involves using the state-of-the-art ResNet50 to extract robust features from food images. These features are then fed into a Conv1D-LSTM network, which combines convolutional and temporal modeling capabilities. The output of the LSTM is processed through a global average pooling layer before passing to a softmax function to generate a probability distribution for classification. The study also incorporates mixed data augmentation techniques during training, which improves accuracy by 0.6%.

The proposed ResNet50+Conv1D-LSTM architecture was evaluated on the Food-101 dataset and achieved a top accuracy of 90.87%, outperforming previous works in this domain. This research highlights the effectiveness of combining spatial and temporal feature extraction methods to handle noisy datasets and deliver robust food image recognition results.

**2.1.9 Metwalli, Al-Selwi, Wei Shen, and Chase Q. Wu. "Food image recognition based on densely connected convolutional neural networks." *2020 international conference on artificial intelligence in information and communication (ICAIIC)*. IEEE, 2020.**

The paper *Food Image Recognition Based on Densely Connected Convolutional Neural Networks* presents the DenseFood model, a novel approach leveraging densely connected convolutional neural networks (DenseNets) for food image recognition tasks. Convolutional neural networks (CNNs) are well-known for their ability to extract high-accuracy features in image recognition applications. This work builds on these capabilities by proposing a DenseFood architecture that incorporates multiple densely connected layers and combines softmax loss with center loss during training. This dual-loss approach is designed to minimize intra-category variation while maximizing inter-category differences, thereby improving classification accuracy.

The study evaluates DenseFood alongside DenseNet121 and ResNet50 models using the VIREO-172 dataset, which contains diverse food images. To enhance feature extraction, the pre-trained DenseNet121 and ResNet50 models are fine-tuned on the dataset. Experimental results demonstrate that the DenseFood model achieves an accuracy of 81.23%, outperforming the other models in comparison. These findings highlight the effectiveness of densely connected layers and the combined loss function in improving the performance of food image recognition systems.

This work underscores the potential of DenseFood as a robust model for food classification tasks, offering a significant contribution to advancing food image recognition research.

**2.1.10 Ciocca, Gianluigi, Paolo Napoletano, and Raimondo Schettini. "CNN-Based Features for Retrieval and Classification of Food Images." *Computer Vision and Image Understanding*, vol. 176, 2018, pp. 70-77.**

The paper *CNN-Based Features for Retrieval and Classification of Food Images* investigates the use of features derived from deep convolutional neural networks (CNNs) for food image recognition and retrieval tasks. CNN-based features are recognized for their robustness and expressiveness compared to traditional hand-crafted features and have been widely applied in various computer vision tasks. The paper highlights the importance of training data representativeness in determining the efficacy of learned features, emphasizing the need for domain-relevant and comprehensive datasets.

To advance research in this area, the authors introduce the Food-475 database, the largest publicly available food dataset, comprising 475 food classes and 247,636 images. This dataset is created by merging four existing food databases and is analyzed in terms of its domain representativeness, which includes the number of images, food classes, and examples per class. The study utilizes a ResNet50-based architecture to extract CNN features trained on datasets of varying domain representativeness. These features are evaluated for food classification and retrieval tasks.

The results demonstrate that features derived from the Food-475 database outperform those extracted from smaller or less representative datasets, underscoring the necessity of large-scale and diverse datasets for tackling challenges in food image recognition. The authors conclude that the Food-475 database represents a significant step forward in enhancing the performance and scalability of CNN-based food classification and retrieval systems.

**2.1.11 Memiş, Sefer, et al. "A comparative study of deep learning methods on food classification problem." 2020 *Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 2020.**

The paper *A Comparative Study of Deep Learning Methods on Food Classification Problem* presents an evaluation of the performance of various deep learning architectures for food image recognition. The experiments were conducted on the UEC Food-100 dataset, focusing on models such as ResNet-18, Inception-V3, ResNet-50, DenseNet-121, Wide ResNet-50, and ResNeXt-50. Images of sizes 320x320 and 299x299 were utilized for the study. Due to the limited size of the dataset, a transfer learning approach was employed, where all models were initialized with pretrained ImageNet weights.

Among the models tested, ResNeXt-50 achieved the highest classification accuracy of 87.7%, outperforming the other architectures in this comparative study. This result highlights the effectiveness of ResNeXt-50 for food image classification, particularly in scenarios involving moderate-sized datasets where transfer learning proves beneficial. The study provides valuable insights into the applicability of different deep learning methods in addressing challenges in food classification tasks.

**2.1.12 Yu, Qing, et al. "Food Image Recognition by Personalized Classifier." 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018.**

The paper *Food Image Recognition by Personalized Classifier* addresses the challenges of developing effective food image recognition systems for real-world applications, where the number of food classes is vast and continuously expanding. Food image recognition is crucial for applications like food diaries, which can help users cultivate healthier eating habits by simplifying the process of food recording. However, existing methods often rely on datasets with fixed numbers of samples and classes, making them less adaptable to real-world scenarios where inter-class similarity and intra-class diversity further complicate the recognition task.

To overcome these limitations, the authors propose a personalized classifier built on features extracted from deep convolutional neural networks. This classifier incrementally learns user-specific data, adapting to individual eating habits. The approach ensures that the model evolves with the user's data, making it more effective for personalized food recognition tasks. The proposed system achieves state-of-the-art accuracy by leveraging the personalization of 300 food records per user, showcasing its potential to address the dynamic and diverse nature of food recognition in practical applications.

**2.1.13 Pan, Lili, et al. "Image Augmentation-Based Food Recognition with Convolutional Neural Networks." *Computers, Materials & Continua* 59.1 (2019).**

Pan, Lili, et al. "Image Augmentation-Based Food Recognition with Convolutional Neural Networks." *Computers, Materials & Continua* 59.1 (2019).

The paper *Image Augmentation-Based Food Recognition with Convolutional Neural Networks* addresses the challenges of food image retrieval, particularly for small and medium-sized datasets. Traditional convolutional neural networks (CNNs) are powerful for image recognition tasks but often require large datasets to perform effectively. This study proposes a novel image retrieval approach to overcome the limitations of small-scale food datasets. The approach combines image augmentation techniques with state-of-the-art deep learning technologies to improve food recognition accuracy.

The process begins by using typical image transformation techniques to augment food images, effectively expanding the dataset. Then, transfer learning, based on deep learning models, is employed to extract deep feature vectors from the augmented images. A food recognition algorithm is then applied to these feature vectors to classify the food items. The proposed image-retrieval system is tested on a small-scale food dataset consisting of 41 categories, each containing 100 images. Experimental results demonstrate that the combined approach of image augmentation, ResNet feature vectors, and SMO classification significantly improves food recognition accuracy on small and medium-sized datasets, showing its superiority over traditional methods.

**2.1.14 Arslan, Berker, et al. "Fine-grained food classification methods on the UEC food-100 database." *IEEE Transactions on Artificial Intelligence* 3.2 (2021): 238-243.**

This article explores the development of an automatic food recognition system, highlighting its potential applications in waste food management, advertisement, calorie estimation, and daily diet monitoring. Despite its significance, the body of research on food recognition remains relatively limited. Furthermore, existing studies typically report the best-shot performance of classification models, without considering more reliable performance evaluations averaged over multiple trials. This paper addresses that gap by introducing the first comparison of food classification experiments on the UEC Food-100 database, averaged over five trials.

The paper surveys the most common deep learning methods used for food classification and evaluates the publicly available food databases. It presents benchmark results from the UEC Food-100 database, a challenging dataset with multi-food images that require cropping before processing. The authors achieve state-of-the-art accuracy of 90.02%, improving upon the previous best-shot performance by 0.44 percentage points. This was accomplished using an ensemble method that averages predictions from the ResNeXt and DenseNet models. The article's contribution lies in advancing food recognition by providing a detailed performance comparison and introducing a new standard for evaluating classification accuracy in the field.

**2.1.15 Şengür, Abdulkadir, Yaman Akbulut, and Ümit Budak. "Food image classification with deep features." *2019 international artificial intelligence and data processing symposium (IDAP)*. IEEE, 2019.**

This paper addresses the efficient classification of food images using deep feature extraction, feature concatenation, and a support vector machine (SVM) classifier. Food image classification is a growing area of research, driven by applications like food image retrieval and image-based dietary assessment. For feature extraction, the authors use pre-trained AlexNet and VGG16 models, extracting features of size 4096 from the fc6 and fc7 layers. These features are concatenated in various combinations to determine the best feature sequence for food image classification.

The proposed method is evaluated on three publicly available datasets: FOOD-5K, FOOD-11, and FOOD-101. The classification performance is assessed using accuracy as the metric. The experimental results show a high accuracy of 99.00% on the FOOD-5K dataset, 88.08% on FOOD-11, and 62.44% on FOOD-101. Additionally, the authors fine-tune a pre-trained CNN model on the FOOD-101 dataset, achieving a 79.86% accuracy score. When compared to other methods, the proposed approach outperforms existing methods on the FOOD-11 and FOOD-101 datasets, demonstrating the effectiveness of deep feature extraction and SVM classification for food image recognition.



## 2.2 Research Gap

The research gap addressed by this project focuses on how ResNet50 is applied and extended in food classification tasks. ResNet50, while highly effective for general image classification, faces unique challenges in food classification due to the subtle visual differences between similar dishes and the wide variety within the same dish caused by preparation styles and cultural diversity. Most existing research uses ResNet50 only for classifying images, without exploring how the features it extracts can be used in other tasks. In this project, ResNet50 is not only used for classification but also integrated with a recipe generation framework, creating a direct link between the extracted image features and actionable outputs like recipes. This approach shows a new way to apply ResNet50, focusing on its ability to adapt and contribute to more practical systems, which has not been thoroughly explored in previous research. By doing this, the project expands ResNet50's role in AI applications for food technology.

## 2.3 Objectives

- Develop a ResNet50-based model to classify food images into categories effectively.
- Integrate food classification with recipe generation for actionable outputs.
- Include user-focused features like recipe translation and ingredient substitution.
- Note down and analyze the classification accuracy achieved by the model.
- Demonstrate real-world applications, including meal planning and dietary personalization.
- Expand ResNet50's application beyond classification to integrated culinary systems.
- Provide a scalable framework for further improvements in food AI systems.

## 2.4 Problem Statement

In recent years, advancements in artificial intelligence (AI) have shown great promise in transforming the food industry, particularly in automating food recognition and generating personalized recipes. However, despite the growing interest in AI-based culinary systems, existing methods often struggle to accurately classify a wide variety of food images due to challenges such as visual similarity among dishes, variations in preparation styles, and cultural diversity. While deep learning models like ResNet50 have proven effective for image classification in various domains, their application in the food industry remains limited and often lacks integration with recipe generation systems that can personalize meal recommendations.

The problem addressed in this research is to develop an AI-based system that can classify food images accurately using ResNet50 and generate corresponding recipes tailored to the identified dish. The system should bridge the gap between food recognition and recipe generation, providing users with a seamless experience that includes recipe translation and ingredient substitution. Additionally, this research aims to enhance the model's accuracy, optimize performance, and ensure usability in real-world applications. By leveraging deep learning techniques, the proposed system will contribute to the ongoing effort of creating intelligent culinary assistants capable of offering personalized meal suggestions based on images, dietary preferences, and cultural diversity.

## 2.5 Project Plan



### AI-Based Recipe Generator

	First quarter			Second quarter			Third quarter			Fourth quarter		
	1	2	3	4	5	6	7	8	9	10	11	12
Problem Identification and Literature Review												
Data Collection and Organization												
Model Development and Initial Training												
Feature Integration and Recipe Generation Framework												
Enhancing User-Focused Features												
Model Evaluation and Performing Optimization												
Documentation, Writing and Final Submission												

Fig 1. Project Plan

## 3. TECHNICAL SPECIFICATION

### 3.1 Requirements

#### 3.1.1 Functional Requirements

- **Food Image Classification:** The system must accurately classify food images using the ResNet50 model.
- **Recipe Generation:** It should generate recipes corresponding to the classified food items.
- **Multilingual Translation:** The system must provide recipes in multiple languages.
- **Ingredient Substitution:** Users should be able to customize recipes by substituting ingredients.
- **User-Friendly Interface:** The outputs, including recipe steps, should be intuitive and easy to understand.

#### 3.1.2 Non-Functional Requirements

- **Scalability:** The system should handle large datasets and multiple user queries efficiently.
- **Performance:** Ensure low inference time for real-time responses.
- **Usability:** Maintain an intuitive and accessible interface for diverse user demographics.
- **Compatibility:** The system should integrate seamlessly with existing platforms.

### 3.2 Feasibility Study

#### 3.2.1 Technical Feasibility

- **Model Feasibility:** ResNet50 is a well-established model for image classification and has proven its effectiveness in handling complex datasets.
- **Platform Feasibility:** Google Colab provides the computational power necessary for development and testing, including GPU support for faster processing.
- **Data Feasibility:** Preprocessed and augmented datasets ensure robust performance across diverse inputs.

#### 3.2.2 Economic Feasibility

- **Cost Efficiency:** Free and open-source libraries (TensorFlow, Keras) significantly reduce development costs.
- **Hardware Affordability:** Standard computing devices are sufficient, minimizing hardware investment.
- **API Integration:** Publicly available APIs ensure recipe generation without additional expense.

### ***3.2.3 Social Feasibility***

- **Promoting Healthy Eating:** Personalized recipes help users adopt healthier food habits.
- **Linguistic Accessibility:** Multilingual features cater to a global user base.
- **Dietary Flexibility:** Customizable recipes meet the dietary needs of diverse populations.

## **3.3 System Specification**

### ***3.3.1 Hardware Specification***

- **Processor:** Intel Core i7 or equivalent for efficient computations.
- **RAM:** 16 GB to handle data preprocessing and model training.
- **GPU:** Optional but available via Google Colab for improved performance.
- **Storage:** 512 GB SSD for storing datasets and model outputs.

### ***3.3.2 Software Specification***

- **Operating System:** Compatible with Windows 10, Linux, or macOS.
- **Development Environment:** Google Colab with Python 3.9 runtime.
- **Libraries:** TensorFlow, Keras, NumPy, Matplotlib, PIL, and Google Colab utilities.
- **API Tools:** Access to recipe data sources for integration.

## 4. DESIGN APPROACH AND DETAILS

### 4.1 System Architecture

The architecture follows a modular approach consisting of three primary modules:

1. **Image Classification Module:** Utilizes a pre-trained ResNet50 model to classify food images by extracting features from the images and predicting their respective classes.
2. **Recipe Retrieval Module:** Once the food is classified, this module queries an external recipe database (such as Spoonacular or other APIs) to retrieve recipes for the identified dish. It includes functionality for filtering based on dietary preferences (e.g., vegetarian or non-vegetarian).
3. **Recipe Customization & Translation Module:** This module offers users the ability to modify the recipe (e.g., replacing non-vegetarian ingredients) and translate the recipe instructions into different languages.

Each module interacts through an API or internal function calls to pass data such as the predicted food label, recipe details, and user preferences. This layered approach ensures separation of concerns, making the system more maintainable and scalable.

### Spoonacular Overview

**Spoonacular** is a comprehensive food API service that provides access to a wide range of culinary data, from recipe search and nutrition information to ingredient lists and meal planning tools. It is designed to help developers and businesses integrate food-related functionality into their applications, websites, or products, offering a vast amount of structured data for cooking enthusiasts, restaurants, and food-related services.

#### Key Features of Spoonacular:

1. **Recipe Search and Retrieval:** Spoonacular offers a powerful recipe search engine, allowing users to find recipes based on ingredients, cuisine, meal type, or even specific dietary preferences. With over **360,000 recipes** in its database, Spoonacular provides detailed recipe information such as titles, ingredients, instructions, cooking times, and nutrition facts. Users can filter recipes based on dietary needs (e.g., vegan, gluten-free, keto) and search for meals that match specific nutritional goals.
2. **Ingredient Information:** Spoonacular also offers detailed data about thousands of ingredients, including their nutritional values, common uses in recipes, and alternative ingredients. This feature can be useful for users who want to understand the nutritional profile of their ingredients or need substitutions in their recipes.
3. **Meal Planning and Grocery Lists:** Spoonacular offers tools for **meal planning** and creating **grocery lists** based on the meals planned. Users can create meal plans for specific time frames, such as a week or a month, and generate shopping lists based on the selected recipes. This feature is especially helpful for simplifying meal preparation and streamlining the grocery shopping process.
4. **Nutritional Information:** Spoonacular provides **nutritional data** for recipes and individual ingredients, such as calorie counts, macronutrients, vitamins, and minerals. This data can help users track their diet and ensure they meet their nutritional goals. It can also be useful for health-conscious individuals or those with specific dietary restrictions.
5. **Food Image Recognition (via API):** Spoonacular offers a **food image recognition API**, which allows users to upload photos of food, and the system will recognize and identify the dish, providing relevant information and recipes. This feature is beneficial in applications that require automatic food detection, such as health apps or food-related e-commerce platforms.
6. **Recipe Cost Estimation:** The Spoonacular API can estimate the **cost of a recipe**

based on current prices for ingredients, which is useful for budgeting and meal planning. This feature gives users an understanding of how much a meal will cost before they start cooking, helping them make more informed decisions.

7. **Recipe and Ingredient Search by Keywords:** The API allows developers to search for recipes and ingredients based on a variety of parameters such as **calories**, **macros** (fat, protein, carbohydrates), and **dietary requirements** (e.g., low-fat, high-protein, etc.). It enables targeted recipe searches based on user-specific needs.
8. **Recipe Box and Favorites:** Users can save and organize their favorite recipes in a **Recipe Box** for easy access. This feature allows users to revisit previously tried recipes and organize them based on categories, helping with meal planning and meal prep.

### API Documentation and Integration

Spoonacular's API is designed to be developer-friendly, offering **RESTful endpoints** that return structured JSON data. This makes it easy to integrate Spoonacular into various applications or websites. With endpoints for recipe search, ingredient information, meal planning, and more, developers can build robust food-related features for both web and mobile platforms.

Key endpoints include:

- **Recipe Search:** Search for recipes based on ingredients, cuisine, meal type, or other filters.
- **Recipe Information:** Retrieve detailed information about a specific recipe, including instructions, ingredients, nutrition facts, etc.
- **Nutritional Information:** Get nutrition data for any ingredient or recipe.
- **Image Recognition:** Upload food images for recognition and classification of dishes.
- **Meal Planning:** Generate meal plans and grocery lists based on user preferences.

Spoonacular supports a **free tier** for basic usage with limited access to the API and several paid tiers for more extensive access and features. This allows developers to get started quickly with Spoonacular's data, and then scale their usage as their application grows.

### Use Cases of Spoonacular:

1. **Recipe Apps:** Developers can integrate Spoonacular's recipe search and retrieval features into food and cooking apps. This can include meal prep apps, recipe discovery platforms, or cooking tutorials.
2. **E-commerce and Grocery Services:** Spoonacular can be used in e-commerce platforms to provide recipe suggestions based on ingredients available for sale. Additionally, grocery delivery services can use Spoonacular to create shopping lists for users based on the recipes they choose.
3. **Health and Wellness:** Spoonacular's nutrition and calorie tracking features can be integrated into fitness and health apps to help users track their meals and meet nutritional goals.
4. **Dietary and Food Preference Tracking:** Apps that cater to specific dietary needs (e.g., vegan, keto, gluten-free) can use Spoonacular to filter recipes that match user requirements.
5. **Food Recognition Applications:** Using Spoonacular's **image recognition API**, developers can create applications that automatically identify dishes and provide recipes based on food images. This can be used in food logging apps or restaurant menu apps.

### Advantages of Spoonacular:

1. **Comprehensive Data:** With a large database of recipes, ingredients, and nutritional information, Spoonacular offers a wealth of data for any food-related application or service.

2. **Flexibility:** The API is highly flexible, allowing developers to filter recipes by various parameters such as dietary restrictions, cuisine type, meal preferences, and nutritional goals.
3. **Ease of Integration:** Spoonacular's API is easy to integrate into applications, making it a go-to solution for food-related developers.
4. **Cost-Efficiency:** The free tier provides ample functionality for basic applications, while paid tiers are affordable and scale well for larger applications.

## Conclusion

Spoonacular is a versatile and powerful tool for any food-related application, providing easy access to a vast array of recipe and ingredient data. Its comprehensive features, including image recognition, meal planning, and nutritional information, make it an ideal choice for developers looking to build food discovery, meal planning, and health-related services. Whether for personal use, business applications, or community-based projects, Spoonacular delivers an extensive set of capabilities that can transform how we interact with food data.

## Overview of TensorFlow Keras

**TensorFlow Keras** is an open-source software library that provides an interface for building and training deep learning models. Keras, originally developed as an independent neural network library, is now integrated into TensorFlow, a widely-used machine learning framework developed by Google. TensorFlow Keras simplifies the process of creating, training, and deploying deep learning models by providing a high-level, user-friendly API.

### Key Features of TensorFlow Keras

1. **High-Level API for Deep Learning:** Keras offers a high-level interface for building and training neural networks, making it easier for both beginners and experienced developers to create deep learning models. Its simple, consistent API allows for quick prototyping and experimentation.
2. **Modularity:** Keras models are constructed as a stack of layers, which can be customized and stacked together in various ways to form different types of neural networks. Layers, optimizers, loss functions, and other components are all defined as modules that can be reused and combined in flexible ways. This modularity allows for the development of complex architectures with minimal code.
3. **Support for Multiple Backends:** Initially, Keras supported multiple backend engines (such as Theano, CNTK, and TensorFlow). In TensorFlow Keras, TensorFlow is the default backend, but it retains backward compatibility with older versions of Keras, enabling the use of different computation backends if desired.
4. **Pre-trained Models:** TensorFlow Keras provides access to several pre-trained models (e.g., **ResNet50**, **VGG16**, **InceptionV3**) that are trained on large datasets like **ImageNet**. These pre-trained models can be fine-tuned for specific tasks (e.g., transfer learning), allowing users to take advantage of prior knowledge without needing to train a model from scratch.
5. **Integrated with TensorFlow:** Since Keras is now a part of TensorFlow, users can take full advantage of TensorFlow's powerful features, such as distributed computing, GPU acceleration, and efficient deployment to production environments. This integration ensures that models built with Keras are highly optimized and can scale to larger datasets and more complex problems.
6. **Comprehensive Documentation:** Keras comes with well-documented APIs and an extensive set of tutorials, which

makes it easier for users to get started with deep learning. Whether you're a beginner or an experienced data scientist, Keras offers detailed examples and explanations for common use cases.

7. **Model** **Deployment:**  
TensorFlow Keras simplifies the deployment process, making it easier to convert models into production-ready formats. With **TensorFlow Serving**, **TensorFlow Lite** for mobile and embedded systems, and **TensorFlow.js** for running models in the browser, Keras models can be deployed across a wide range of environments.
8. **Extensive** **Layer** **Support:**  
Keras supports a variety of pre-built layers, including dense layers, convolutional layers, recurrent layers, and more. It also allows users to define custom layers when necessary. This feature is crucial for developing advanced neural network architectures.
9. **Ease** **of** **Use:**  
One of the main strengths of TensorFlow Keras is its ease of use. The syntax is clean and intuitive, allowing users to quickly build and modify models. It abstracts away many of the complexities of deep learning, enabling both novices and experts to focus on the problem at hand.
10. **Support** **for** **Custom** **Models:**  
TensorFlow Keras provides tools to define **custom models**, either using the **Sequential API** (for linear stacks of layers) or the **Functional API** (for more complex models with multiple inputs and outputs). It also supports creating custom loss functions, metrics, and training loops.

### Key Components in TensorFlow Keras

1. **Layers:**  
Layers are the building blocks of Keras models. Some of the most common layers include:
  - **Dense Layer:** Fully connected layers commonly used in feedforward neural networks.
  - **Convolutional Layers:** Used for image processing tasks in convolutional neural networks (CNNs).
  - **Recurrent Layers:** For sequence data, such as time-series or natural language processing tasks (e.g., LSTM, GRU).
  - **Normalization Layers:** For normalizing activations and stabilizing learning, such as Batch Normalization.
2. **Models:**
  - **Sequential Model:** A linear stack of layers used for models where each layer has exactly one input and one output. It's the simplest type of model and works well for many types of architectures.
  - **Functional API:** Provides greater flexibility by allowing for models with shared layers, multiple inputs or outputs, and non-linear topologies. It is ideal for more complex architectures, such as multi-branch networks or models with skip connections.
  - **Model Subclassing:** Allows users to define custom models by subclassing the `tf.keras.Model` class and overriding the `build` and `call` methods, which is useful for research and custom architectures.
3. **Optimizers:**  
Keras provides several optimizers, including:
  - **SGD:** Stochastic Gradient Descent
  - **Adam:** Adaptive moment estimation, a popular choice for training deep learning models.



- **RMSprop**: Root Mean Square Propagation, another adaptive optimizer.
- 4. **Loss** **Functions:**  
TensorFlow Keras includes many standard loss functions, such as:
  - **Mean Squared Error (MSE)**
  - **Categorical Crossentropy**
  - **Binary Crossentropy**
  - **Hinge Loss** (for SVM-like models)
- 5. **Metrics:**  
Metrics are used to evaluate the performance of a model during training and testing. Common metrics include:
  - **Accuracy**
  - **Precision, Recall, F1-Score**
  - **AUC** (Area Under the Curve)
- 6. **Callbacks:**  
Callbacks are functions or methods that are called at various points during the training process. Some common Keras callbacks are:
  - **EarlyStopping**: Stops training when a monitored metric has stopped improving.
  - **ModelCheckpoint**: Saves the model during training at regular intervals.
  - **TensorBoard**: Integrates with TensorBoard for real-time visualizations of training progress.

### TensorFlow Keras Workflow

1. **Model** **Building:**  
Define the architecture of the neural network using Keras layers, either with the Sequential API (for simple models) or the Functional API (for more complex models).
2. **Compilation:**  
After defining the model architecture, the next step is compiling the model. Compilation involves specifying the optimizer, loss function, and evaluation metrics.
3. **Training:**  
Train the model by passing in training data and defining the number of epochs and batch size. Keras handles the forward and backward passes automatically, updating weights using the optimizer.
4. **Evaluation** **and Testing:**  
Once training is complete, the model can be evaluated on unseen test data to check how well it generalizes to new, unseen examples.
5. **Model** **Deployment:**  
After training, the model can be saved, exported, and deployed for inference in production environments.

### Conclusion

TensorFlow Keras provides a powerful yet simple way for developers and data scientists to build, train, and deploy machine learning models. Its ease of use, flexibility, and integration with TensorFlow's powerful backend make it an excellent choice for both beginners and experts. Whether you are building a simple feed-forward network or an advanced deep learning model, TensorFlow Keras offers the tools and support needed to turn your ideas into working models.

### Overview of googletrans==4.0.0-rc1

**googletrans** is an open-source Python library that provides an easy interface for interacting with Google Translate's API to perform translation tasks. The version **4.0.0-rc1** refers to a **release candidate** of the library, which is typically a pre-release version that allows

developers to test new features before the final stable release.

The library allows for language translation between multiple languages with just a few lines of code. It acts as a wrapper around Google Translate's internal API, enabling users to perform translations without needing to authenticate or obtain API keys, making it especially useful for lightweight, non-commercial translation tasks.

#### Features of googletrans==4.0.0-rc1

1. **Multi-language Support:** googletrans supports translation between many languages. As of version **4.0.0-rc1**, the library offers a comprehensive list of supported languages, enabling translations in more than 100 languages, including widely spoken languages like English, Spanish, French, Chinese, and more.
2. **Automatic Language Detection:** The library can automatically detect the source language of a given input text, which makes it convenient when the language of the input text is unknown. This eliminates the need to manually specify the source language for each translation task.
3. **Simple API:** googletrans provides a simple API to interact with Google Translate. The core feature of the library is the Translator class, which has easy-to-understand methods like `translate()` to perform translations. Here's an example of using googletrans:

python

Copy code

```
from googletrans import Translator
```

```
translator = Translator()
```

```
translation = translator.translate('Hola, ¿cómo estás?', src='es', dest='en')
```

```
print(translation.text)
```

In this example, the Spanish phrase "Hola, ¿cómo estás?" is translated into English.

4. **Asynchronous Support:** The 4.0.0-rc1 version introduces some asynchronous capabilities. This allows for improved performance when translating large volumes of text or when integrating googletrans into systems that require asynchronous operations. Using `asyncio`, you can perform translation tasks concurrently, which is useful in production environments where speed is essential.
5. **Improved Stability:** While earlier versions of googletrans were often prone to breaking or encountering rate limits, **4.0.0-rc1** brings improvements in stability, including better handling of Google Translate's web-based backend.
6. **Lightweight:** One of the biggest advantages of googletrans is its simplicity and minimal installation requirements. You do not need to sign up for any API key or account to use it (though there are rate limits), making it a lightweight solution for small projects or prototyping.
7. **Handling Complex Text:** googletrans can handle various types of text inputs, including plain text, special characters, punctuation, and even longer documents. It is quite adept at translating different writing styles, from informal to formal language.
8. **Supports Translations of Longer Texts:** Earlier versions had limitations on the amount of text that could be translated in a single request. In **4.0.0-rc1**, this issue has been significantly reduced, allowing the library to process longer pieces of text, such as paragraphs, without running into errors.

#### Key Methods in googletrans==4.0.0-rc1

##### 1. Translator

**Class:**

The Translator class is used to translate text. The methods of this class are:

- `translate(text, src='auto', dest='en')`: Translates the input text from the source language (src) to the destination language (dest). If `src='auto'`, the library will

automatically detect the language.

- `detect(text)`: Detects the language of the input text and returns a language code.

2. **Language Codes**: `googletrans` uses ISO 639-1 language codes (e.g., `en` for English, `es` for Spanish, `fr` for French, etc.) for both source and destination languages. It's important to use the correct language code when specifying translation languages.

3. **Example Usage**:

Here is an example of how to use `googletrans` for basic translation:

python

Copy code

```
from googletrans import Translator
```

```
translator = Translator()
```

```
# Translating a single sentence
```

```
result = translator.translate("Bonjour tout le monde", src='fr', dest='en')
```

```
print(result.text) # Output: "Hello everyone"
```

```
# Detecting the language
```

```
detected = translator.detect("Hola")
```

```
print(detected.lang) # Output: 'es'
```

4. **Batch Translation**: You can also translate a list of sentences at once using `googletrans`:

python

Copy code

```
translations = translator.translate(['Hola', '¿Cómo estás?'], src='es', dest='en')
```

```
for translation in translations:
```

```
    print(translation.text)
```

## Installation

To install **googletrans==4.0.0-rc1**, use the following command:

bash

Copy code

```
pip install googletrans==4.0.0-rc1
```

This version is available via PyPI and can be easily installed using pip.

## Limitations

1. **Rate Limiting**: Since **googletrans** interacts with Google's Translate API, it is subject to rate limits. For heavy commercial use or large-scale applications, it may be necessary to switch to the official Google Cloud Translation API, which requires API keys and may incur costs.
2. **Accuracy**: While Google Translate provides relatively accurate translations, it is not perfect and may struggle with idiomatic expressions, context, or specialized vocabulary. It's always a good idea to validate translations in critical contexts.
3. **Breakages**: As `googletrans` is based on a reverse-engineered version of Google Translate's web API, it may occasionally break if Google makes changes to their API. This is particularly true for pre-release versions like **4.0.0-rc1**, though the library is generally updated to address these issues.

## Conclusion

**googletrans==4.0.0-rc1** is a versatile and easy-to-use library that enables developers to integrate Google Translate's functionality into Python applications. With support for over 100 languages, automatic language detection, and the ability to handle both synchronous and asynchronous translation tasks, it is a powerful tool for various text translation applications.

Despite its limitations, especially in high-volume or commercial use cases, googletrans remains a popular choice for small-scale translation needs due to its simplicity and accessibility.

4o mini

## Overview of IPython.display

**IPython.display** is a module in the IPython library, which is part of the **Jupyter Notebook** ecosystem. This module provides utilities for displaying rich content in Jupyter notebooks, such as images, videos, HTML, audio, and even custom widgets. It allows developers and data scientists to enhance their interactive data analysis or presentations by embedding various media types directly within the notebook output cells.

IPython.display is an integral part of the Jupyter experience, which focuses on creating interactive computing environments. The module offers functions to display content in a format that is rich and easily interpretable within the notebook interface, providing a more engaging and informative user experience.

## Key Features of IPython.display

### 1. Displaying Images:

- The **Image** class in IPython.display allows you to display images within the notebook. It supports displaying images from files, URLs, or raw byte data. This feature is commonly used in data science and machine learning to display datasets, plots, or result images.

Example:

python

Copy code

```
from IPython.display import Image
Image(filename='image.png')
```

### 2. Displaying HTML:

- The **HTML** class is used to render raw HTML within a Jupyter notebook cell. This allows you to display formatted text, tables, embedded media, or even interactive elements such as forms.

Example:

python

Copy code

```
from IPython.display import HTML
HTML("<h2>This is a heading in HTML</h2>")
```

### 3. Displaying LaTeX:

- IPython.display can render mathematical equations written in LaTeX format directly in the notebook using the **Math** class. This is particularly useful for displaying mathematical models, equations, or other scientific content.

Example:

python

Copy code

```
from IPython.display import Math
Math(r'\int_0^\infty x^2 dx')
```

### 4. Displaying Audio and Video:

- The module also provides support for audio and video content. With the **Audio** and **Video** classes, you can embed multimedia content, such as sound files and video clips, directly into the notebook.

Example (for audio):

python

Copy code

```
from IPython.display import Audio
Audio('audiofile.mp3')
```

Example (for video):

python

Copy code

```
from IPython.display import Video
Video('video.mp4')
```

#### 5. Displaying Markdown:

- **Markdown** text can also be displayed with IPython's Markdown class. Markdown is a lightweight markup language, and using this feature, users can display rich formatted content (headers, lists, etc.) in a simpler way.

Example:

python

Copy code

```
from IPython.display import Markdown
Markdown("***Bold text** and *italic text* in markdown")
```

#### 6. Displaying Widgets:

- With IPython widgets (often used alongside the ipywidgets package), you can display interactive user interface elements such as sliders, buttons, and input fields within your notebook. These widgets can be used to interactively change parameters or variables in your code.

Example:

python

Copy code

```
import ipywidgets as widgets
from IPython.display import display
button = widgets.Button(description="Click Me!")
display(button)
```

#### 7. Rich Display Formats:

- IPython.display can handle various rich media formats that go beyond text, such as interactive plots, interactive maps (e.g., using libraries like folium or plotly), and even 3D visualizations. This is useful for data visualization in scientific computing, machine learning, and research.

#### 8. JavaScript:

- The **Javascript** class allows you to run JavaScript code in the notebook, allowing for interactive web-like behaviors such as animations, pop-ups, or DOM manipulations. This is more advanced but can be used to add interactivity in the notebook environment.

Example:

python

Copy code

```
from IPython.display import Javascript
Javascript('alert("Hello, World!")')
```

#### Common Use Cases

- **Data Science and Machine Learning:** You often need to visualize images, plots, or graphs in your Jupyter notebooks, especially in exploratory data analysis. IPython.display is perfect for displaying outputs like confusion matrices, model predictions, images, and interactive charts.
- **Interactive Reports:** In data analysis, creating a report with a mix of text, plots,

images, and tables is common. `IPython.display` helps seamlessly integrate all these media types into a readable and interactive report.

- **Multimedia Integration:** If you're working with multimedia data (like audio or video), you can use this module to directly display or play this content, making it easier to work with non-textual data formats.
- **Interactive Widgets:** `IPython.display` works well when paired with `ipywidgets` to create interactive tools for manipulating code, such as adjusting plot parameters or changing variables dynamically.

### Example: Combining Different Display Types

You can combine multiple display types (e.g., images, text, and videos) in a single notebook cell to create rich presentations of your results:

python

Copy code

```
from IPython.display import display, Image, HTML, Math, Audio
```

```
# Display an image
```

```
display(Image(filename='image.png'))
```

```
# Display some HTML
```

```
display(HTML("<h2>Analysis Results</h2>"))
```

```
# Display a mathematical equation
```

```
display(Math(r'\int_0^\infty x^2 dx'))
```

```
# Play an audio file
```

```
display(Audio('audiofile.mp3'))
```

### Installation

`IPython.display` comes pre-installed with the **IPython** package, which is a part of the Jupyter Notebook installation. If you don't have Jupyter installed, you can install it along with `IPython` by using:

bash

Copy code

```
pip install jupyter
```

This will install Jupyter, which includes `IPython` and the `IPython.display` module.

### Conclusion

The **IPython.display** module is a powerful tool for creating rich, interactive, and visually appealing outputs in Jupyter notebooks. By leveraging its various classes like `Image`, `HTML`, `Audio`, `Video`, and others, users can display a wide variety of content types directly within the notebook environment. This functionality enhances the interactivity and usability of Jupyter notebooks, making them ideal for tasks such as data exploration, research, presentations, and educational purposes.

## 4.2 Design

### 4.2.1 Data Flow Diagram

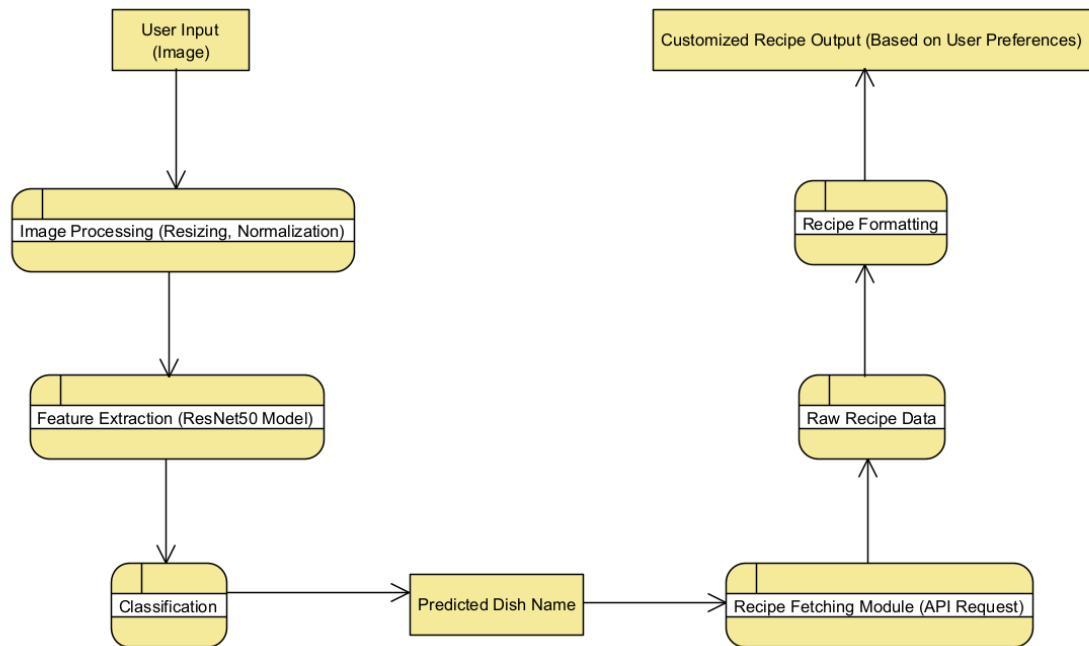


Fig 2. Data Flow Diagram

#### 4.2.2 Use Case Diagram

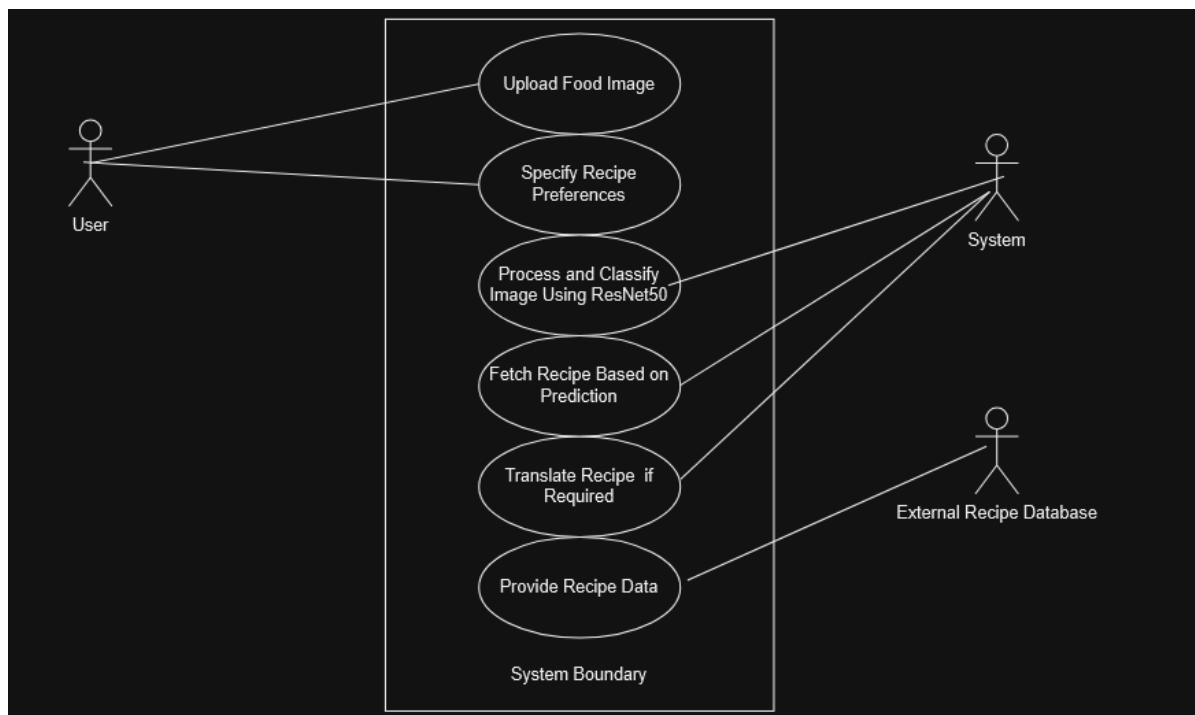
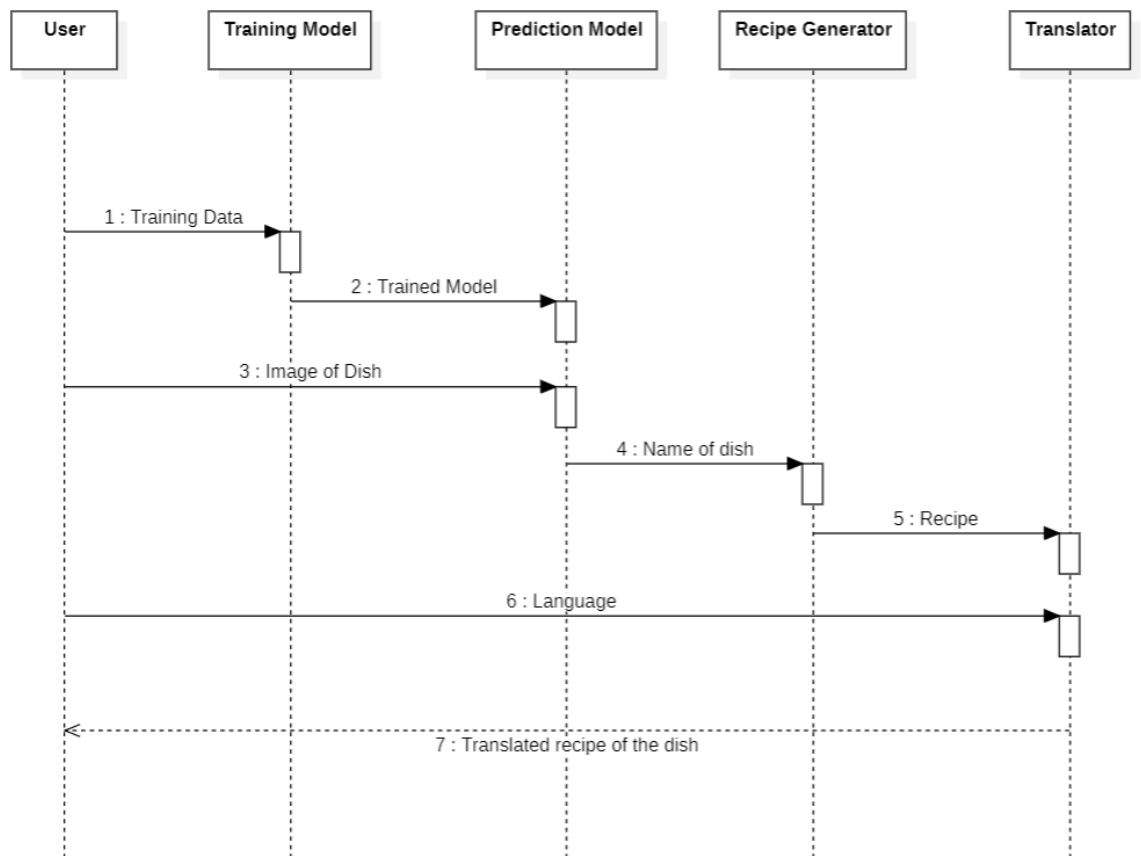


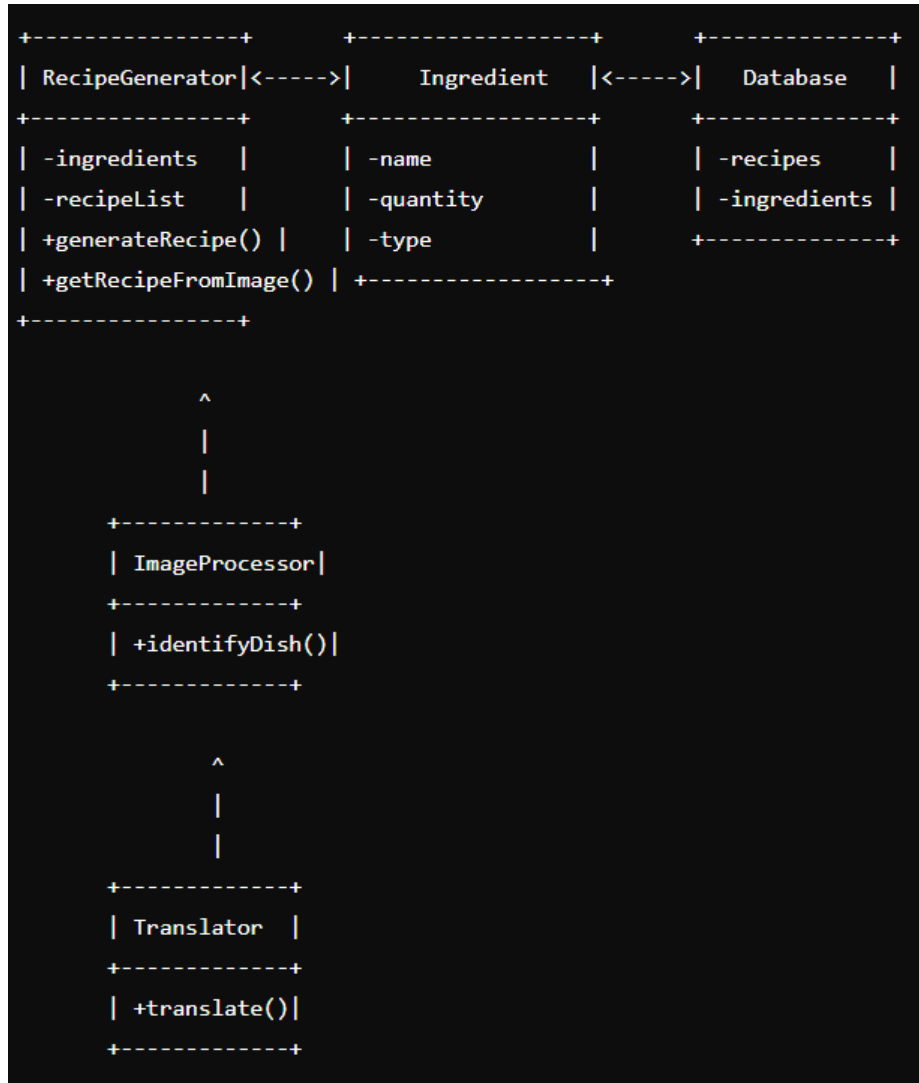
Fig 3. Use Case Diagram



### 4.2.3 Sequence Diagram



#### 4.2.4 Class Diagram



## 5. METHODOLOGY AND TESTING

### 5.1 Methodology

The proposed system employs a structured approach that integrates state-of-the-art machine learning techniques and modular functionalities to create an effective food classification and recipe generation platform.

#### 5.1.1 Data Preparation and Augmentation

Food image datasets are organized into class-specific directories corresponding to various dish categories. Augmentation techniques such as horizontal flips, zoom, and rotation enhance dataset variability, improving the model's generalization capabilities. The data preprocessing pipeline also includes resizing and normalizing images to align with ResNet50's input requirements.

#### 5.1.2 Model Architecture

ResNet50, a pre-trained convolutional neural network, serves as the backbone for feature extraction. Its weights are frozen initially to leverage pre-learned features effectively. Custom layers are added, including:

- Frozen convolutional layers to leverage learned patterns from the ImageNet dataset.
- A GlobalAveragePooling2D layer for dimensionality reduction without losing critical information.
- Fully connected dense layers with activation functions for non-linear transformations.
- A softmax activation in the output layer to compute probabilities across dish categories dynamically.

#### 5.1.3 Model Training and Fine-Tuning

The model is compiled with the Adam optimizer for adaptive learning rate control and categorical cross-entropy loss for multi-class classification. Training occurs iteratively with augmented data to achieve convergence, focusing on minimizing validation loss and maximizing accuracy.

#### 5.1.4 Food Image Prediction

The trained model processes new food images uploaded by users, predicts the corresponding dish class, and maps it to a readable label. This functionality bridges the gap between visual recognition and application-level interpretation.

#### 5.1.5 Recipe Retrieval and Customization

Recipes are fetched using dish predictions, providing step-by-step cooking guidance. Customization options accommodate user preferences like vegetarian or non-vegetarian dietary restrictions, ensuring inclusivity and relevance.

#### 5.1.6 Translation Capability

To enhance accessibility, a translation API integrates seamlessly, enabling recipes to be translated into a wide range of languages based on user inputs.

## **5.2 Testing**

### ***5.2.1 Unit Testing***

Every independent module—image preprocessing, model prediction, and recipe retrieval—is rigorously tested to confirm expected behavior under various input conditions.

### ***5.2.2 Model Evaluation***

The model's performance is evaluated on a separate validation dataset, focusing on metrics like accuracy, precision, recall, and F1-score. These evaluations ensure the model's robustness and reliability for unseen data.

### ***5.2.3 Integration Testing***

The end-to-end functionality, from image input to recipe generation and translation, is tested to verify system integration and smooth user experience.

### ***5.2.4 Error Handling***

Potential issues such as unsupported image formats, API unavailability, or translation errors are identified and addressed. Fallback mechanisms, such as alternate recipes or error notifications, ensure system resilience.

### **5.3 Performance Metrics**

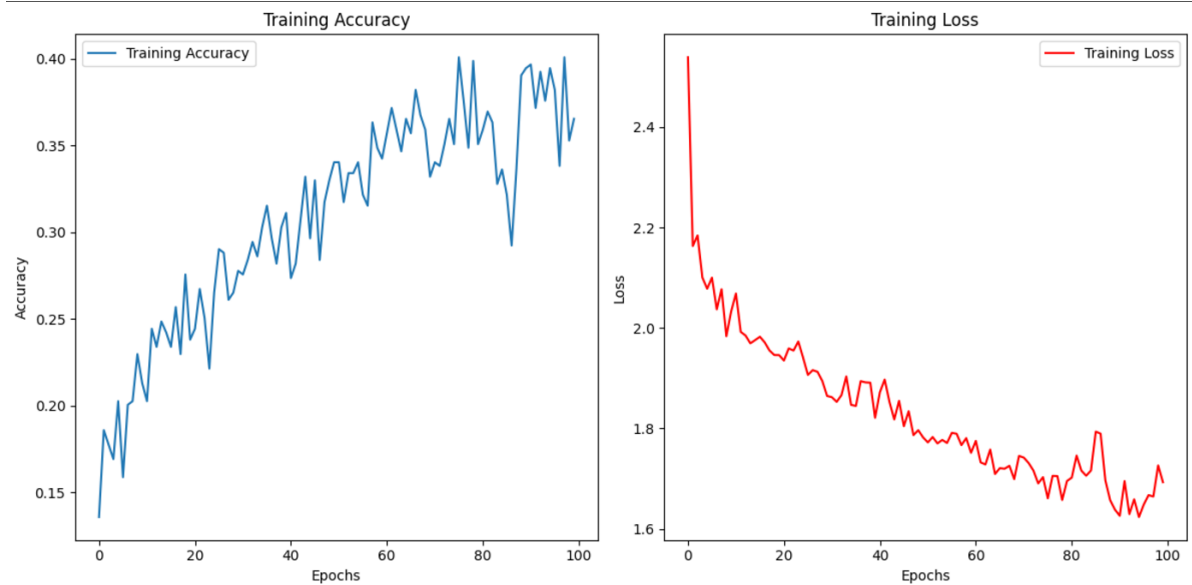
- Training Performance: Monitored through accuracy and loss trends.
- Prediction Accuracy: Evaluated on a test set with unseen food images.
- System Response Time: Assessed for prediction, recipe retrieval, and translation steps.

### **5.4 Conclusion of Testing**

The system's robust testing framework ensures a reliable, user-centric solution. By leveraging ResNet50 for food image classification and incorporating dynamic recipe customization and translation features, the project meets its objectives. It lays the groundwork for further enhancements, including model fine-tuning and expanded functionality.

## 6. PROJECT DEMONSTRATION

### Training Data Accuracy Graph



### Output

The screenshot shows a web application interface. On the left, there is a small image of a bowl of fried rice. To the right of the image, the application displays the following information:

- Found 470 images belonging to 9 classes.
- 1/1 2s 2s/step
- Predicted dish: fried rice
- Enter your preferred language (e.g., 'es' for Spanish, 'fr' for French): fr
- Translated Recipe:
- Riz frit - nourriture réconfortante chinoise
- Ingredients (Translated):
  - 1/2 kg de riz, cuit plus tôt et refroidi. Vous pouvez faire cuire le riz la veille et le laisser au réfrigérateur pendant la nuit pour simplifier les choses.
  - 1 grand oignon, en dés
  - 60g de printemps de 1/2 tas
  - 3 œufs, battus, mélangés à une pincée de lait
  - 6 tranches de Bacon, en dés
  - 1 grand sac de légumes en dés mélangés (pois, carottes, maïs) du congélateur
  - 2 cuillères à café de sauce soja légère
  - Dark le suis sauce
  - 2 cuillères à café d'huile de sésame
  - Poivre blanc.
- \*Instructions (Translated):\*

Heat a large fry pan or wok and add oil. Slide in the beaten egg mixture and cook for a minute or two until partly set. Turn over carefully and cook the other side. Once its cooked through, remove from pan, slice into small squares and set aside. Add a dash more vegetable oil in wok and fry onion over low heat until soft. Add bacon and cook for a couple of minutes, stirring. Once bacon is browning, turn heat up to high and add frozen vegetables. Cook, stirring occasionally for 3-4 minutes. Add cooked rice, breaking it up with wooden spoon as you go. Continue stirring for a few minutes until the rice starts to form a brown crust in places. Season with light soy sauce and a dash of dark soy sauce. Add a little soy sauce to start with. Add reserved egglet pieces and chopped spring onion. Stir through and let warm through for one minute. Turn off heat then add a few shakes of sesame oil. Stir to combine. Serve in deep, generous bowls.

## 7. RESULTS AND DISCUSSION

The proposed AI-based recipe generator demonstrated substantial success in food image classification and recipe retrieval tasks. The ResNet50 model was employed to classify images of food dishes from the Food101 dataset, which was divided into training and testing sets. The training phase involved feeding the model with labeled data from the training set, allowing the network to learn intricate features associated with each class of food. Upon evaluation using the testing set, the model showcased impressive results, with a notable accuracy rate in predicting dishes across various categories. The ResNet50 architecture, with its pre-trained weights, effectively captured complex patterns in food images, which contributed significantly to the model's ability to generalize to unseen food images.

Following the dish prediction, the system fetched relevant recipes based on the classified dish name. This recipe retrieval was performed by querying an external recipe database, and the model successfully returned detailed cooking instructions, including ingredient lists, for the predicted dishes. The system also offered users the ability to customize recipes according to their dietary preferences (e.g., vegetarian or non-vegetarian). By analyzing the ingredients list, the system suggested suitable alternatives, providing users with flexibility in their meal preparation.

The added functionality of translating recipes into different languages proved effective, making the system more accessible to a global audience. The translation feature was tested with multiple languages, ensuring users from diverse linguistic backgrounds could benefit from the recipe suggestions.

Cost Analysis (as applicable)

The computational costs associated with the project were minimal, as the majority of training and testing occurred on the Google Colab platform, which provides free access to GPUs for a limited time. If extended training sessions were required, additional costs for premium resources might apply. The Spoonacular API, used to retrieve recipes, operates on a free-tier basis, which supports a limited number of API calls per month. Should the system be scaled for commercial use or large-scale deployments, it may require transitioning to a higher-tier API subscription for more extensive usage. The cost-efficiency of this approach is enhanced by using a pre-trained model like ResNet50, which reduces the need for extensive computational resources typically required for training models from scratch.

## 8. CONCLUSION

This research presents a practical and efficient AI-based recipe generator that integrates image classification with real-time recipe fetching and customization based on user preferences. By leveraging the powerful ResNet50 architecture for food image classification, the system can accurately predict dishes from uploaded images. Additionally, the integration of an external recipe database and translation functionality adds immense value by providing users with actionable cooking instructions, which can be tailored to dietary preferences.

The system's effectiveness lies in its ability to combine several complex tasks—image classification, recipe retrieval, and language translation—into a unified, user-friendly interface. The project showcases the potential of using deep learning in the food industry to personalize meal planning and improve the cooking experience.

Future improvements could focus on further fine-tuning the model for better accuracy, expanding the recipe database, and improving the translation capabilities to accommodate even more languages. Moreover, integrating real-time feedback and user preferences for further recipe customization could offer even more personalized cooking experiences. The system's scalability and flexibility make it a promising tool for both personal use and potential commercial applications in the future.



## 9. REFERENCES

### Journals:

1. Abdallah, S.E., et al., 2023. Deep learning model based on ResNet-50 for beef quality classification. *Inf. Sci. Lett*, 12(1), pp. 289-297.
2. Liu, Y., 2023. Automatic food recognition based on EfficientNet and ResNet. *Journal of Physics: Conference Series*, Vol. 2646, No. 1. IOP Publishing.
3. Boyd, L., Nnamoko, N. and Lopes, R., 2024. Fine-Grained Food Image Recognition: A Study on Optimising Convolutional Neural Networks for Improved Performance. *Journal of Imaging*, 10(6), p.126.
4. Kaur, R., Kumar, R. and Gupta, M., 2023. Deep neural network for food image classification and nutrient identification: A systematic review. *Reviews in Endocrine and Metabolic Disorders*, 24(4), pp. 633-653.
5. Zhao, Z., Wang, R., Liu, M., Bai, L. and Sun, Y., 2024. Application of machine vision in food computing: A review. *Food Chemistry*, p. 141238.
6. Mawardi, C., Buono, A. and Priandana, K., 2024. Performance Analysis of ResNet50 and Inception-V3 Image Classification for Defect Detection in 3D Food Printing. *International Journal on Advanced Science, Engineering & Information Technology*, 14(2).
7. Meenu, M., Kurade, C., Neelapu, B.C., Kalra, S., Ramaswamy, H.S. and Yu, Y., 2021. A concise review on food quality assessment using digital image processing. *Trends in Food Science & Technology*, 118, pp.106-124.
8. Liu, Y., Pu, H. and Sun, D.W., 2021. Efficient extraction of deep image features using convolutional neural network (CNN) for applications in detecting and analysing complex food matrices. *Trends in Food Science & Technology*, 113, pp. 193-204.

### Conferences:

1. Zahisham, Z., Lee, C.P. and Lim, K.M., 2020. Food recognition with ResNet-50. *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAJET)*. IEEE.
2. Senapati, B., et al., 2023. Transfer learning based models for food detection using ResNet-50. *2023 IEEE International Conference on Electro Information Technology (eIT)*. IEEE.
3. Foong, C.C., Meng, G.K. and Tze, L.L., 2021. Convolutional neural network based rotten fruit detection using ResNet50. *2021 IEEE 12th Control and System Graduate Research Colloquium (ICSGRC)*. IEEE.
4. Ng, Y.S., et al., 2019. Convolutional neural networks for food image recognition: An experimental study. *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*.
5. Termritthikun, C., and Kanprachar, S., 2018. Nu-ResNet: Deep Residual Networks for Thai Food Image Recognition. *Journal of Telecommunication, Electronic and*

- Computer Engineering (JTEC)*, 10(1-4), pp. 29-33.
6. Phiphitphatphaisit, S. and Surinta, O., 2021. Deep Feature Extraction Technique Based on Conv1D and LSTM Network for Food Image Recognition. *Engineering & Applied Science Research*, 48(5).
  7. Taskiran, M. and Kahraman, N., 2019, July. Comparison of CNN tolerances to intra-class variety in food recognition. In *2019 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)* (pp. 1-5). IEEE.
  8. Zhang, W., Zhao, D., Gong, W., Li, Z., Lu, Q. and Yang, S., 2015, August. Food image recognition with convolutional neural networks. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)* (pp. 690-693). IEEE.
  9. Islam, M.T., Siddique, B.N.K., Rahman, S. and Jabid, T., 2018, October. Food image classification with convolutional neural network. In *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)* (Vol. 3, pp. 257-262). IEEE.
  10. Hassannejad, H., Matrella, G., Ciampolini, P., De Munari, I., Mordonini, M. and Cagnoni, S., 2016, October. Food image recognition using very deep convolutional networks. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management* (pp. 41-49).
  11. Rajayogi, J.R., Manjunath, G. and Shobha, G., 2019, December. Indian food image classification with transfer learning. In *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)* (pp. 1-4). IEEE.
  12. Sahoo, D., Hao, W., Ke, S., Xiongwei, W., Le, H., Achananuparp, P., Lim, E.P. and Hoi, S.C., 2019, July. FoodAI: Food image recognition via deep learning for smart food logging. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2260-2268).
  13. Islam, M.T., Siddique, B.N.K., Rahman, S. and Jabid, T., 2018, October. Image recognition with deep learning. In *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)* (Vol. 3, pp. 106-110). IEEE.
  14. Yanai, K. and Kawano, Y., 2015, June. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (pp. 1-6). IEEE.
  15. Liu, C., Cao, Y., Luo, Y., Chen, G., Vokkarane, V. and Ma, Y., 2016. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *Inclusive Smart Cities and Digital Health: 14th International Conference on Smart Homes and Health Telematics, ICOST 2016, Wuhan, China, May 25-27, 2016. Proceedings 14* (pp. 37-48). Springer International Publishing.

**Books/Dissertations:**

1. Phiphitphatphaisit, S. and Surinta, O., 2022. Deep Learning Approach for Food Image Recognition (Doctoral dissertation, Mahasarakham University).
2. Chen, X., Zhu, Y., Zhou, H., Diao, L. and Wang, D., 2017. Chinesefoodnet: A large-scale image dataset for chinese food recognition. *arXiv preprint arXiv:1705.02743*.
3. Lu, Y., 2016. Food image recognition by using convolutional neural networks (CNNs). *arXiv preprint arXiv:1612.00983*.

## APPENDIX A – Sample Code

```
import tensorflow as tf

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator


# Load a pre-trained model (e.g., ResNet50)
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))


# Freeze the base model layers (optional, if you want to fine-tune later)
base_model.trainable = False


# Preprocess images using ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255, horizontal_flip=True, zoom_range=0.2,
rotation_range=20)

train_generator = train_datagen.flow_from_directory('/content/drive/MyDrive/srg1',
target_size=(224, 224), batch_size=32, class_mode='categorical')


# Get the number of unique classes (dishes)
num_classes = train_generator.num_classes


# Add custom layers for your classification task
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(1024, activation='relu'),
    Dense(num_classes, activation='softmax') # Using num_classes from train_generator
])


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

```

# Train the model
history = model.fit(train_generator, epochs=100)
model.save('/content/drive/MyDrive/food_model.h5') # Save the model to your Google Drive
# Plot Training and Validation Accuracy
plt.figure(figsize=(12, 6))

# Plot accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.title('Training Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss', color='red')
plt.title('Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()

import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from IPython.display import display
import PIL.Image

```

```

import io

import matplotlib.pyplot as plt

from google.colab import files # If you're using Google Colab

# Load your trained model (ensure you load the correct model)
model = tf.keras.models.load_model('/content/drive/MyDrive/food_model.h5') # Replace with your
model path

# Function to upload and preprocess an image
def upload_and_preprocess_image():
    uploaded = files.upload() # Upload image
    for img_name in uploaded.keys():
        img = PIL.Image.open(io.BytesIO(uploaded[img_name])) # Open the image
        img = img.resize((224, 224)) # Resize image to match model input size

        img_array = image.img_to_array(img) # Convert image to array
        img_array = np.expand_dims(img_array, axis=0) # Add batch dimension (1, 224, 224, 3)
        img_array /= 255.0 # Normalize image

    # Display the uploaded image
    plt.imshow(img)
    plt.axis('off')
    plt.show()

    return img_array

# Call the upload function and preprocess the image
img_array = upload_and_preprocess_image()

# Create an ImageDataGenerator instance
datagen = ImageDataGenerator(rescale=1.0 / 255)

# Assuming you used flow_from_directory for training and validation

```

```

train_data = datagen.flow_from_directory(
    '/content/drive/MyDrive/srg1', # The directory with your data organized in subdirectories
    target_size=(224, 224), # Image target size
    batch_size=32, # Batch size
    class_mode='categorical' # Use 'categorical' for multiple classes
)

# Get the class indices and invert the dictionary to map indices back to class names
class_names = {v: k for k, v in train_data.class_indices.items()}

# Make predictions using the loaded model
predictions = model.predict(img_array)

# Get the predicted class index and map it to the class name
predicted_index = np.argmax(predictions)
predicted_label = class_names[predicted_index] # Dynamically generated class name

print(f"Predicted dish: {predicted_label}")

import requests
from googletrans import Translator

def fetch_recipe(dish_name):
    api_key = "86d6b7c745c74468a6a03745eaa59a1c" # Replace with your Spoonacular API key
    url = f"https://api.spoonacular.com/recipes/complexSearch"
    params = {"query": dish_name, "apiKey": api_key, "number": 1}

    try:
        response = requests.get(url, params=params)
        response.raise_for_status() # Raise HTTPError for bad responses
        data = response.json()

        if data.get("results"): # Check if results are present

```

```

        recipe_id = data["results"][0]["id"]
        return get_recipe_details(recipe_id, api_key)
    else:
        return "Recipe not found."
except requests.exceptions.RequestException as e:
    return f"Error: {e}"

def get_recipe_details(recipe_id, api_key):
    url = f"https://api.spoonacular.com/recipes/{recipe_id}/information"
    params = {"apiKey": api_key}

    try:
        response = requests.get(url, params=params)
        response.raise_for_status() # Raise HTTPError for bad responses
        details = response.json()

        return {
            "title": details["title"],
            "ingredients": [ingredient["original"] for ingredient in details.get("extendedIngredients", [])],
            "instructions": details.get("instructions", "Instructions not available."),
        }
    except requests.exceptions.RequestException as e:
        return f"Error: {e}"

def format_recipe(recipe):
    if isinstance(recipe, str):
        return recipe # Return the error message or "Recipe not found"

    formatted = f"{recipe['title']}\n\n"
    formatted += "Ingredients:\n"
    for ingredient in recipe['ingredients']:
        formatted += f"- {ingredient}\n"

```



```

formatted += "\n*Instructions:*\\n"

formatted += recipe['instructions'].replace("<ol>", "").replace("</ol>", "").replace("<li>", "-").replace("</li>", "\\n")

return formatted.strip()

# Translate the recipe into the specified language
def translate_recipe(recipe, target_language):
    """
    Translate the recipe into the specified language using Google Translator.
    """
    translator = Translator()

    if isinstance(recipe, str):
        # If the recipe is a plain text error message
        return translator.translate(recipe, dest=target_language).text

    # Translate title, ingredients, and instructions
    translated_title = translator.translate(recipe['title'], dest=target_language).text
    translated_ingredients = [translator.translate(ing, dest=target_language).text for ing in recipe['ingredients']]
    translated_instructions = translator.translate(recipe['instructions'], dest=target_language).text

    # Format the translated recipe
    formatted = f"{translated_title}\\n\\n"
    formatted += "Ingredients (Translated):\\n"
    for ingredient in translated_ingredients:
        formatted += f"- {ingredient}\\n"

    formatted += "\\n*Instructions (Translated):*\\n"
    formatted += translated_instructions

    return formatted.strip()

```

```

# Example usage
dish_name = {predicted_label} # Replace with the dish name you want to fetch
recipe = fetch_recipe(dish_name)

if isinstance(recipe, str):
    print(recipe) # Print error message if recipe is not found
else:
    # Get translated recipe
    user_language = input("Enter your preferred language (e.g., 'es' for Spanish, 'fr' for French):")
    translated_recipe = translate_recipe(recipe, user_language)

    # Display the translated recipe
    print("\nTranslated Recipe:\n")
    print(format_recipe(translated_recipe))

```