Culinary Curator: An AI-Powered Recipe Assistant

Project Report submitted in the partial fulfilment

Of

Bachelor of Technology
Computer Engineering

By

Aanya Lari (B101)

Khushi Tejwani (B076)

Ved Naik (B051)

Under the supervision of

Ishani Saha

(Assistant Professor, Department of Computer Engineering, MPSTME)

SVKM's NMIMS University

(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

Vile Parle (W), Mumbai-56

(2023-2024)

CERTIFICATE

This is to certify that the project entitled **"Culinary Curator: an AI-Powered Recipe Assistant"**, has been done by Aanya Lari, Khushi Tejwani, and Ved Naik, under my guidance and supervision & has been submitted in partial fulfilment of degree of Bachelor of Technology in Computer Engineering of MPSTME, SVKM's NMIMS (Deemed-to-be University), Mumbai, India.

_____ _____

Project mentor (name and Signature) Examiner (name and Signature)

(Intenal Guide)

Date:

Place: Mumbai

_____

(HoD) (name and Signature)

ACKNOWLEDGEMENT

NAME	ROLL NO.	SAP ID
Aanya Lari	B101	70092000060
Khushi Tejwani	B076	70022000178
Ved Naik	B051	70022000109

ABSTRACT

Cooking at home is a treasured and vital part of our daily routines in an era typified by increased urbanization, hectic lifestyles, and our growing reliance on technology. It builds bridges across cultures, strengthens familial relationships, and allows us to experience a wide culinary environment. Yet, in the middle of modern life's complications, deciding what to prepare with the supplies at hand can be a daunting task.

This culinary realm has inspired the development of our project. The significance of our research lies within the domain of food and health, addressing the challenges that modern people face. We propose the Culinary Curator, a technological innovation at the intersection of artificial intelligence, natural language processing and computer vision. This cooking assistant serves as a guidance to individuals seeking inspiration in the culinary world, particularly in moments when one needs to decide what to cook from the ingredients available at hand, which is a daunting task. It also provides nutritional analysis for the recipes, thereby encouraging healthy and mindful eating.

This core of this project lies in the rigorous research . These models have undergone extensive inspection to guarantee that the finished product is of the greatest calibre. The selected models, ‘BERT’ for recipe recommendations, ‘MobileNetV2’ for ingredient detection from photos, and GPT-2 for recipe generation are the result of a great deal of research and demonstrate how well they can meet consumers' culinary needs.

This project offers a solution that enhances culinary experiences, makes cooking at home easier, and is in line with the values of our contemporary, technologically advanced lifestyle. It's proof of the timeless importance of home-cooked meals and their ability to unite people. The creative methodology of our project has the potential to completely transform the way we think about cooking and guarantee that this beloved custom survives the difficulties of the contemporary world.

Table of Contents

Topics	Pages
List of Figures	**VII**
List of Tables	**IX**
Chapter 1: Introduction	10
1.1 Background	10
1.2 Motivation and Scope	11
1.3 Problem statement	13
1.4 Salient contribution	14
1.5 Organization of report	15
Chapter 2: Literature survey	17
2.1 Introduction to Domain	17
2.2 Literature Survey	19
2.3 Research Gaps	22
Chapter 3 Methodology and Implementation	24
3.1 Data Collection and Description	24
3.2 Data Cleaning and Preprocessing	26
3.3 Model Implementation	28
3.4 Additional Features	41
3.5 Project Setup	44
Chapter 4 Result and Analysis	58
4.1 Performance Metrics	58
4.2 Ingredient Recognition Models	59

4.3 Recipe Recommendation Models	64
Chapter 5 Advantages, Limitations and Applications	70
5.1 Advantages	70
5.2 Limitations	72
5.3 Applications	73
Chapter 6 Conclusion and Future Scope	74
References	76

List of Figures

Fig. No.	Name of the figure	Page No.
3.5.1	Website of Culinary Curator for Recipe Recommendation	45
3.5.2	Website of Culinary Curator for GPT Recipe Generation	45
3.5.3	Website of Culinary Curator for Nutrition Analysis	46
3.5.4	Website of Culinary Curator for Recipe Recommendation (with User Text Input)	46
3.5.5	Website of Culinary Curator for Recipe Recommendation (With recommended recipe)	47
3.5.6	Website of Culinary Curator for Recipe Recommendation (With recommended recipe)	47
3.5.7	Website of Culinary Curator for Recipe Recommendation (With recommended recipe output)	48
3.5.8	Website of Culinary Curator for Recipe Recommendation (With recommended recipe output)	48
3.5.9	Website of Culinary Curator for Recipe Recommendation (With Image Input)	49
3.5.10	Website of Culinary Curator for Recipe Recommendation (With name output of ingredient recognized)	49
3.5.11	Website of Culinary Curator for Recipe Recommendation (With more ingredients recognized)	50
3.5.12	Website of Culinary Curator for Recipe Recommendation (With recommended recipe output for image input)	50
3.5.13	Website of Culinary Curator for Recipe Recommendation (With recommended recipe output for Image input)	51
3.5.14	Website of Culinary Curator for Recipe Recommendation (With recommended recipe output for Image input)	51
3.5.15	Website of Culinary Curator for GPT Recipe Generator (With input)	52
3.5.16	Website of Culinary Curator for GPT Recipe Generator (With generated recipe output)	52

3.5.17	Website of Culinary Curator for Nutrition Analysis (With input)	53
3.5.18	Website of Culinary Curator for Nutrition Analysis (with output)	53
3.5.19	Website of Culinary Curator for Nutrition Analysis (With output)	54
3.5.20	Website of Culinary Curator for Nutrition Analysis (With output)	54
3.5.21	Model Architecture for Recipe Recommendation Using BERT	55
3.5.22	Model Architecture for Ingredient Recognition Using MobileNetV2	55
3.5.23	Model Architecture for Recipe Generation using GPT-2	56
3.5.24	Model Architecture for Frontend	57
4.2.1	Performance Metrics for CNN	59
4.2.2	Confusion Matrix for CNN	60
4.2.3	Performance Metrics for MobileNetV2	61
4.2.4	Confusion Matrix for MobileNetV2	62
4.3.1	Precision at k for BERT Model using BERT Embeddings	64
4.3.2	Precision at k for BERT Model using LSTM	65
4.3.3	Precision at k for BERT Model using GENSIM	65
4.3.4	Precision at k for BERT Model using Word2Vec and TF-IDF	66
4.2.5	Precision at k for BERT Model using Sentence Transformer	66
4.2.6	Precision at k for Logistic Regression	67

List of Tables

Table No.	List of Tables	Page No.
3.1.1	6000+ Indian Food Recipes Dataset: Features and Description	25
4.3.1	Precision at k for all the Recipe Recommendation Models	68

Chapter 1

Introduction

1.1 Background

The world of culinary is on the edge of a technological revolution driven by Artificial Intelligence. Traditional Indian cooking methods and recipe selection are now to be replaced by AI powered tools that streamline the cooking process. The development of recipe recommendation systems driven by AI is one of the most fascinating facets of this transition. These systems leverage the ability of artificial intelligence to comprehend ingredients, via text or images, and recommend personalized recipes on the basis of ingredients available at the user's hand. To fully appreciate the significance of this project, it is essential to delve into the evolving landscape of AI-driven culinary solutions.

AI has revolutionised meal planning, recipe accessibility, and dietary accommodations in the world of cooking. The sources of culinary guidance are no longer traditional cookbooks and manual searches. AI has opened up a whole new world of culinary possibilities, including personalised dietary guidance and automated recipe creation. Consideration of the availability of ingredients, or the essence of recipes, is the key to these breakthroughs.

The needs of the contemporary culinary scene are efficiency, personalisation, and convenience. AI successfully meets these changing needs and establishes itself as a reliable cooking partner for both beginners and experts. Through the usage of advanced algorithms and AI models, our project is a perfect sous chef for everyone since it can efficiently search through enormous recipe datasets. It can also accommodate dietary preferences and limits and suggest meals that are perfect for different occasions.

1.2 Motivation and Scope of the Project

Cooking at home is a treasured and vital part of our daily routines in an era typified by increased urbanization, hectic lifestyles, and our growing reliance on technology. It builds bridges across cultures, strengthens familial relationships, and allows us to experience a wide culinary environment. Yet, in the middle of modern life's complications, deciding what to prepare with the supplies at hand can be a daunting task, especially when we miss home-cooked Indian food. The significance of our project is established within this culinary realm.

With hectic schedules, innumerable types of diets, hectic schedules, and an abundance, or lack, of available ingredients, choosing what to cook can be a daunting task. AI-powered recipe recommendation systems address this challenge, which employ artificial intelligence to analyse large recipe datasets and user profiles to generate customized Indian meal ideas that take into account dietary preferences, ingredient availability, individual tastes, as well as providing an overview of the nutritional value of the recipe recommended. By recommending meals based on ingredients that are readily available, they not only make the process of making culinary decisions easier but also have the potential to decrease food waste.

Furthermore, cultural culinary traditions can be honoured and preserved by AI-powered systems, opening up a wider range of authentic and diverse recipes to a worldwide consumer base. In essence, AI-powered recipe recommendation systems offer a practical and innovative solution to enhance the cooking experience, minimize food waste, and promote the exploration of rich, diverse cuisines and healthy and mindful eating.

With this project, we hope to achieve the following objectives:

1. **Ingredient Recognition:** To enhance user involvement, it is recommended to create a strong system for recognising ingredients, capable of processing both textual inputs and photos of ingredients.
2. **Recipe Personalisation:** Develop an AI-driven recommendation system for recipes that can make customised recipe recommendations based on user preferences and identified ingredients.

3. **Efficiency and Convenience:** Make meal planning easier by providing recipe options quickly and easily. This will cut down on the time and effort needed to locate recipes that work.
4. **Variety & Exploration:** Encourage culinary discovery by providing consumers with a wide variety of Indian recipes that will enhance their perspectives on cooking.
5. **Nutritional Analysis:** Analyse the nutritional value of the recipe using the ingredients, encouraging mindful and healthy eating.
6. **Flexibility:** The system is able to adjust to users' dietary needs, tastes, and accessible ingredients to provide a customised cooking experience for each person.
7. **User-Friendly Interface:** Create an easy-to-use interface that allows users of all culinary skill levels to access the AI-powered recipe recommendation.
8. **Nutritional Considerations:** To encourage healthy and well-balanced meal choices, incorporate nutritional data and dietary guidelines into the recipe suggestions.
9. **Continuous Improvement:** Put in place systems for ongoing learning and improvement to make sure the system adapts to shifting user tastes and gastronomic fads.
10. **Reduce Food Wastage:** Optimize the utilization of available resources and reduce the wastage of food by recommending recipes that use the ingredients at hand.

1.3 Problem statement

The culinary landscape lacks a streamlined and personalized solution for individuals seeking efficient and diverse recipe recommendations that consider their ingredient availability, dietary preferences, and nutritional needs. Traditional recipe search methods do not harness the capabilities of ingredient recognition and AI-driven personalization, resulting in a gap between user expectations and the culinary resources available.

The aim is to design and develop an AI-powered recipe recommendation system that generates personalized and diverse Indian recipe suggestions based on user input of available ingredients. The system should leverage Natural Language Processing and Computer Vision techniques to process user input, analyse a comprehensive dataset of Indian recipes, and employ advanced recommendation algorithms as well as a large language model to offer users a curated list of recipes that suit their dietary preferences and the ingredients they have on hand.

The project aims to simplify recipe discovery, inspire Indian culinary creativity, and promote efficient meal planning for users to reduce food waste and user expenses.

1.4 Salient contribution

The project introduces a novel AI-powered Culinary Curator that leverages advanced technologies such as natural language processing (NLP) and computer vision. This innovation stands at the forefront of merging technology with the age-old tradition of cooking.

The project showcases the successful integration of two distinct AI models: "bert-base-uncased" for recipe recommendations and "MobileNetV2" for ingredient recognition from images. This cross-modal approach ensures a comprehensive and versatile solution.

The project places the user at the centre of its design, offering a practical and intuitive interface that recognizes ingredients from both text and images. It simplifies the cooking process for users, making it more user-friendly.

Through rigorous evaluation and comparison of multiple AI models, the research provides a robust foundation for its choice of the ‘BERT’ and ‘MobileNetV2’ models. This ensures that the end product is of the highest quality and precision.

The project acknowledges the cultural significance of home cooking and underscores how technology can strengthen cultural connections. By transcending geographical boundaries, this technology has the potential to bring diverse culinary experiences to users' kitchens.

In an era marked by urbanization, busy schedules, and technological integration, the AI-powered Culinary Curator is a relevant and essential tool that aligns with the contemporary way of life.

Thus, our project and study emphasizes the enduring value of home-cooked Indian meals and how technology can revolutionize this culinary tradition to thrive amidst the challenges and complexities of the modern world.

1.5 Organization of report

Chapter 1: Introduction

1.1 Background

In this chapter's opening section, we delve into the contextual backdrop of the project, shedding light on the primary factors that led to its inception.

1.2 Motivation and Scope

We explore the motivations behind this project, outlining the objectives and the scope of our research, which helped define the project's boundaries.

1.3 Problem Statement

This section articulates the specific challenges and issues we aim to address with our project, setting the stage for the subsequent discussions.

1.4 Salient Contribution

We highlight the key contributions and innovations brought forth by our project, emphasizing its significance and relevance in the field.

1.5 Organization of Report

Concluding the introductory chapter, we provide a road map for the report, offering readers a clear overview of what to expect in the following chapters.

Chapter 2: Literature Survey

2.1 Introduction to Domain

This chapter kicks off with an introduction to the project's domain, ensuring that readers have a foundational understanding of the subject matter.

2.2 Literature Survey

We present a thorough review of the existing literature in the domain, summarizing relevant studies, research, and technological advancements.

2.3 Research Gaps

This section identifies the gaps in the current body of knowledge that our project aims to address, underscoring the novel aspects of our work.

Chapter 3: Methodology and Implementation

3.1 Data Collection and Description

Here, we detail the methods used to collect the project's data and provide a comprehensive description of the dataset.

3.2 Data Cleaning and Preprocessing

We discuss the critical steps taken to clean and preprocess the data, ensuring its suitability for subsequent analysis.

3.3 Model Implementation

This chapter unveils the technical implementation of our models, walking through the architecture, algorithms, and code used.

3.4 Additional Features

We introduce any supplementary features or components that enhance the project's functionality or user experience.

3.5 Project Setup
This section elucidates the setup process of our project, offering practical insights for those interested in replicating or extending our work.

Chapter 4: Result and Analysis
4.1 Performance Metrics
We outline the metrics employed to evaluate the project's performance and provide an overview of the results achieved.
4.2 Ingredient Recognition Models
In this section, we delve into the performance and analysis of the ingredient recognition models developed as part of our project.
4.3 Recipe Recommendation Models
We present an in-depth analysis of the recipe recommendation models, discussing their effectiveness and limitations.

Chapter 5: Advantages, Limitations, and Applications
5.1 Advantages
We enumerate the advantages and strengths of our project, emphasizing its potential impact and benefits.
5.2 Limitations
This section candidly acknowledges the limitations and challenges encountered during the project's execution.
5.3 Applications
We explore the real-world applications and contexts where our project can be deployed to deliver value.

Chapter 6: Conclusion and Future Scope
In the final chapter, we draw conclusions from our project's outcomes, summarizing its contributions and implications. We also discuss potential future directions and areas for further research.

References
This section compiles all the sources, references, and citations used throughout the report, ensuring transparency and academic rigor.

Chapter 2

Literature Survey

2.1 Introduction to Domain:

Artificial intelligence (AI), natural language processing (NLP), as well as computer vision, are bringing about a revolution in the world of food and culinary by introducing innovative applications that cater to diverse aspects of the culinary domain. These advancements offer solutions to help users discover new recipes, cook healthier meals, and minimize food waste while transforming the way we approach cooking and meal planning.

One of the most popular applications of AI in the culinary domain is recipe recommendation systems. AI-powered recipe recommendation systems have gained prominence, offering personalized cooking suggestions to users. These systems recommend recipes to users based on their dietary restrictions, cooking skills, time constraints, and preferences using machine learning algorithms[1]. For example, the recipe website Yummly uses AI to recommend recipes to users based on their past browsing history, saved recipes, preferred cuisines, allergens, etc.

Furthermore, deep learning approaches can learn intricate patterns of user-item interaction through nonlinear transformations, they are superior in recommender systems. They are very good at representation learning, which eliminates the need for feature engineering by hand. Deep learning algorithms are perfect for problems involving temporal dynamics in user behaviour and item evolution since they can successfully simulate sequential data. Also, these methods are more flexible, allowing for the creation of hybrid recommendation models that are able to adjust to a variety of traits and variables [2].

For a recipe to be recommended, ingredients must be extracted from the recipe text. In order to enable computers to match recipes with accessible components, advanced NLP techniques are used to recognise and extract elements and quantities from recipes. Given the variety of ways components are mentioned in recipes, this process can be difficult to parse, tokenize, and recognise entities [3].

Another promising application of computer vision techniques for ingredient recognition. These systems can identify the ingredients in images and videos. This can be used to recommend recipes based on ingredients identified from the image [4] or identify the ingredients used to make a dish from the image of

the dish [5]. For example, the app Fooducate to identify the ingredients in food images and provide users with nutritional information.

AI and NLP can also be used to generate recipes from natural language descriptions [6]. This can be used to help users create custom recipes that meet their specific needs and preferences. For example, the website RecipeAI generates recipes based on users' dietary restrictions, cooking skills, and ingredient preferences.

Finally, AI can be used to develop virtual chef assistants. These AI-powered systems can help users with a variety of culinary tasks, such as meal planning, cooking, and shopping. For example, the Amazon Alexa virtual assistant can be used to find recipes, generate shopping lists, and control kitchen appliances.

In addition to these specific applications, AI and NLP are also being used to improve food safety and develop more sustainable food systems. For example, AI systems are being used to track and identify potential food hazards, perform nutrition analysis and eating behaviour analysis [7] and to develop new ways to reduce food waste [8].

Overall, AI, NLP and computer vision have the potential to revolutionize the culinary domain. By developing innovative applications, these technologies can help people to discover new and exciting recipes, cook healthier meals, and reduce food waste.

2.2 Literature Survey

This review of the literature dives into the field of AI-driven recipe recommendation systems, examining the developments, approaches, and discoveries that help the culinary industry come up with creative solutions. To obtain a thorough grasp of the state-of-the-art in this fascinating topic, we will examine the essential elements, methods, and trends in AI-based recipe recommendations in this survey.

The method used in one study employed a collaborative filtering to make recipe recommendations according to user preferences [1]. The item-based and user-based techniques are examined. The item-based method uses measures like the Tanimoto Coefficient and Log-Likelihood to evaluate recipe similarity without taking user ratings into account. The user-based approach, on the other hand, experiments with various neighbourhood sizes and thresholds and assesses user similarity using metrics like Pearson Correlation and Euclidean Distance. Evaluation metrics include Root Mean Squared Error (RMSE) and Average Absolute Difference (AAD). The results indicate that user-based collaborative filtering yields more accurate recipe recommendations than item-based techniques when there is enough data on user interactions.

Two primary components of another approach used are the identification of food ingredients and the suggestion of recipes. Using a Flask-based web application, food ingredient recognition uses the ResNet-50 CNN model to recognise ingredients from user-uploaded photos. Users can receive customised recipe recommendations by selecting their prefered diet and diet type, and the Edamam API and JavaScript are used to generate the recommendations. A user-friendly platform for recipe recommendations based on ingredients is produced using this methodology [4].

A cutting-edge recommendation system for sophisticated recipe recommendations is called the Hierarchical Graph Attention Network (HGAT) [9]. It is made up of several essential parts that allow HGAT to manage intricate interactions between users, recipes, and ingredients. Type-specific transformations guarantee data type compatibility, whereas node-level attention creates relation-specific embeddings by encoding neighbour nodes' information. These embeddings are combined by relation-level attention to create thorough node representations. Several layers of these representations are combined to improve the final node embedding. Recommendation scores are produced by a score predictor using user and recipe embeddings. HGAT is a valuable addition to recommendation systems since it is trained with a ranking-based objective function.

Recipe generation is an additional facet of recipe recommendation. Using a sizable corpus of recipes, RecipeMC refines the GPT-2 language model with an emphasis on producing coherent and structured recipes [10]. NAME→INGR (recipe name to ingredients) and NAME+INGR→INST (recipe name and ingredients to instructions) are the two activities that make up the text creation process. Reward functions direct the text generating process by promoting consistent outputs, preventing redundancy, and following predetermined recipe structures. During text generation, Monte Carlo Tree Search (MCTS) is employed to explore and choose the next token. The efficacy of this approach in generating recipes that satisfy particular standards and address prevalent constraints in language models is assessed.

An article highlights how important it is to comprehend the model's ability to associate food visuals with their corresponding elements. Because of the intricacies of zero-shot retrieval and visual changes, content-based ingredient recognition is difficult [11]. By using deep architectures to simultaneously learn food categorisation and ingredient recognition, this research tackles these problems. We then use the learned semantic labels and deep features to novel zero-shot recipe retrieval. This paper illustrates the unique zero-shot challenge in cooking recipe retrieval and demonstrates the viability of ingredient recognition through thorough testing on a large dataset of complex photographs of Chinese food.

Another research also delivers matching recipes and nutritional information based on real-time food photographs or image links that are entered. Rather than requiring embeddings, the methodology streamlines the procedure by using a Convolutional Neural Network (CNN) to produce image encodings for recipe retrieval. The authors included an API for retrieving dietary data. The system uses a K-nearest neighbour classifier (KNN) and DenseNet-121 for image processing to get the closest matching recipe for the user-provided food image [12].

The technique with Convolutional Neural Networks (CNNs) for deep multi-ingredient recognition is the main emphasis of another study [13]. It starts with contrasting two foundational architectures that were first intended for object recognition: InceptionV3 and ResNet50. These architectures are adapted for multi-label ingredient classification after being pre-trained on ILSVRC data. Several highly active outputs are possible by applying a sigmoid activation in place of the conventional softmax activation. By using this method, the issue is changed into a multi-label framework where each ingredient's binary output is predicted by the model. The binary cross-entropy loss function is used to optimise the model, encouraging it to match the target binary vectors—which show whether each ingredient is present in the input samples or not—with its predictions.

Another study explores the use of Factorization Machines (FMs) for context-aware recipe suggestion [14]. It uses techniques like Stochastic Gradient Descent, Alternating Least-Squares, and Markov Chain Monte Carlo for parameter estimation and integrates a variety of contextual information from food recipe datasets. The absence of regularisation parameters and learning rate makes MCMC unique. The datasets go through preprocessing and filtration. The goal of the experiments is to improve recommendation accuracy, particularly with sparse datasets. Model performance is evaluated by cross-validation using RMSE and MAE metrics, providing a thorough method for context-aware recipe suggestion for sparse food domain data.

Another approach uses convolutional neural networks (CNNs) to achieve deep multi-ingredient recognition [15]. The first section compares two basic architectures that were originally intended for object recognition: InceptionV3 and ResNet50. After pre-training using ILSVRC data, these designs are adjusted for the purpose of multi-label ingredient classification. To provide several highly active outputs, a sigmoid activation is utilised in place of the normal softmax activation. With this method, the issue is changed to a multi-label framework in which the model forecasts a binary result for every component. By using the binary cross-entropy loss function, which encourages the model to align its predictions with the target binary vectors—which show whether each ingredient is present in the input samples or not—the model is optimised.

2.3 Research Gaps

With this project, we aim to fill the following gaps:

Ingredient Availability: The main goal of many recipe suggestion systems is to provide meals that adhere to dietary restrictions, user preferences, or current trends. But these systems frequently miss an important detail: the ingredients the user currently has access to. In order to close this gap, our project takes into account the user's accessible ingredients, which can vary depending on personal stock, local availability, or dietary requirements. This system is more useful for people who wish to make the most out of the ingredients they have on hand because of its real-world context.

Indian food: With a broad variety of regional recipes and ingredients, Indian cuisine is renowned for its complexity and diversity. Although there are methods for recommending recipes, many of them are not specifically focused on Indian cuisine. Our product offers customers a customised experience while catering to a particular culinary culture. Users who like to delve deeper into the diverse and rich realm of Indian cooking will find this specialisation useful since it may include ingredients and methods not frequently found in other cuisines.

Nutritional Analysis: The inclusion of a nutritional analysis feature in our recipe recommendation app is a another significant and valuable addition to existing research. It addresses a critical aspect of modern dietary choices and health-conscious cooking. With this feature, users can gain a comprehensive understanding of the nutritional content of recipes they intend to prepare, enabling them to make informed choices aligned with their dietary needs and preferences. While recipe recommendation apps are becoming increasingly common, the integration of accurate and detailed nutritional analysis is still relatively scarce. Many apps focus solely on recipe discovery, often leaving users to manually calculate or estimate nutritional values, which can be tedious, time-consuming, and prone to inaccuracies.

Comparative Analysis: This project does more than just put one model for identifying and recommending ingredients into practise. As an alternative, it compares and contrasts various models. This is an important contribution since it clarifies the advantages and disadvantages of different approaches for the scientific community and practitioners. We have offered insights into the most effective approaches for particular tasks, such recipe recommendation or ingredient recognition, by assessing and contrasting these models. For the discipline to advance and to steer future efforts in the proper direction, this research gap is crucial.

By addressing these research gaps, the Culinary Curator contributes to a more comprehensive and practical approach to recipe recommendation, especially in the context of Indian cuisine. This can have a significant impact on users who want to cook with the ingredients they have on hand and explore the diverse world of Indian recipes.

Chapter 3

Methodology and Implementation

3.1 Data Collection and Description

A) Ingredient Recognition:

The first step in ingredient recognition is to collect the necessary data. The dataset used for this research is 'Fruits and Vegetables Image Recognition Dataset', which is available on Kaggle.
We have used a diverse collection of food-related images, including a variety of fruits and vegetables. The dataset encompasses a comprehensive selection of food items:

Fruits: banana, apple, pear, grapes, orange, kiwi, watermelon, pomegranate, pineapple, mango.

Vegetables: cucumber, carrot, capsicum, onion, potato, lemon, tomato, raddish, beetroot, cabbage, lettuce, spinach, soy bean, cauliflower, bell pepper, chilli pepper, turnip, corn, sweetcorn, sweet potato, paprika, jalepeño, ginger, garlic, peas, eggplant.

This rich dataset is organized into three main folders: "train," "test," and "validation," each containing a set of 100, 10, and 10 images, respectively, for different fruits and vegetables. These subfolders within the main categories host the images specific to each food item, making it a valuable resource for food recognition and classification tasks.

B) Recipe Recommendation:

The first step in preparing the data is collecting the necessary data. The dataset used for this research is the '6000+ Indian Food Recipes Dataset', which is available on Kaggle. The dataset consists of 5939 rows and 15 columns. The features of the dataset are the following:

Feature	Description
Srno	Serial number of the recipe
RecipeName	Name of the Recipe
TranslatedRecipeName	Recipe Name translated from Hindi to English (if any)
Ingredients	Ingredients required for the recipe
TranslatedIngredients	Ingredients translated from Hindi to English (if any)
PrepTimeInMins	Preparation Time in minutes
CookTimeInMins	Cooking Time in minutes
TotalTimeInMins	Total Time in minutes
Servings	Number of Servings
Cuisine	Name of Cuisine
Course	Type of Course
Diet	Vegetarian or Non-Vegetarian
Instructions:	Instructions for the recipe
TranslatedInstructions	Instructions translated from Hindi to English (if any)
URL:	Link to the recipe on the website ‘www.archanaskitchen.com’

Table 3.1.1 6000+ Indian Food Recipes Dataset: Features and Description

3.2 Data Cleaning and Preprocessing:

A) Ingredient Recognition:

Overall, data preprocessing and cleaning are crucial to ensure that the dataset is in the right format and that it's ready for training a deep learning model. These steps help improve the model's performance and make the training process smoother and more efficient.

1) **Image Resize:** Resize the images to a consistent size. In this code, the images are resized to (224, 224, 3) using target_size=(224, 224) when loading images with image.load_img.
2) **Image Normalization:** Normalize the pixel values of the images to a range between 0 and 1 by dividing by 255. This is done by img / 255.0.
3) **Data Augmentation:** Data augmentation is a technique to increase the diversity of the training dataset by applying random transformations to the images. In this code, the ImageDataGenerator is used to apply various augmentations, including rotation, width and height shifting, shearing, zooming, and horizontal flipping.
4) **Class Label Encoding:** Ensure that class labels are properly encoded. In this code, class_mode='categorical' is used, which implies that the classes are one-hot encoded.
5) **Train-Validation Split (Optional):** We have a separate validation dataset, so we can split it from the training dataset. The code doesn't show a validation dataset, so validation_data=None is used during model training.
6) **Missing class_names:** The code attempts to map the class index to class names using class_names[class_index].
7) **Error Handling and Exception Handling:** We have included error handling and exception handling in the code to handle various issues that may arise during data preprocessing and loading.

B) Recipe Recommendation:

Effective data cleaning and preprocessing is essential for ensuring the compatibility of the data, especially the ingredients, with the models/algorithms used. The following are the steps involved in preprocessing:

1) **Removing/replacing Parenthesis:** Regular expressions (regex) was used to remove any text within parentheses (e.g., "(extra cheese)") in the "ingredients" column. Further, the opening and closing parentheses containing empty strings are replaced, effectively removing them.
2) **Remove non-alphabet characters:** Words that contain non-alphabetic characters are filtered out, ensuring that only alphabetic words remain.
3) **Remove measuring words:** Measuring words are eliminated and phrases (e.g., "teaspoon," "tablespoon") by filtering out words that are present in list of measuring words defined before.
4) **Remove stop words:** Common English stop words are removed (e.g., "the," "and") using NLTK's corpus of stop words.
5) **Remove accents:** A function is used to remove accents from characters. For example, it converts accented characters to their non-accented equivalents.
6) **Remove extra spaces:** Extra spaces in the text were removed to ensure consistency and readability.
7) **Eliminate punctuations:** Punctuation marks were removed to standardize the text.
8) **Convert to lowercase:** All text was converted to lowercase to ensure case-sensitive analysis.
9) **Tokenization:** It is used in Natural Language Processing to split paragraph or sentence into individual units or tokens, making it easier to work with and analyse [16]. Ingredient descriptions are split into tokens by hyphens or spaces. For example, if the ingredient description is “tomato-sauce”, the tokenization process separates it into two tokens: “tomato” and “sauce”. Each row in the "ingredients" is split by commas, converting it into a list of ingredients.
10) **Lemmatization:** It is also used in Natural Language Processing to break a word down to its root form [17]. It standardizes variations of word to their common root form, enabling comparison and processing of ingredients. For example, for the word “tomatoes”, lemmatization reduces it to its base form “tomato”.

3.3 Model Implementation

A) Ingredient Recognition:

1) MobileNetV2:

MobileNetV2 stands as a notable neural network architecture meticulously crafted for the efficient execution of computer vision tasks, especially on resource-constrained devices. Its architectural prowess encompasses several key components:

Efficient Feature Extraction: MobileNetV2, known for its efficiency, acts as a feature extractor. It processes input images of fruits and vegetables and extracts meaningful features from them. By utilizing depthwise separable convolutions and linear bottlenecks, it efficiently captures essential information from the images while keeping the computational requirements in check.

Transfer Learning: The model used in the code is pre-trained on a vast dataset. This pre-training imparts it with a robust understanding of general visual features. In the context of fruits and vegetable recognition, these general features serve as an excellent starting point. Fine-tuning the model on a more specific dataset (in this case, fruits and vegetables) allows it to adapt and specialize in recognizing these particular items.

High Accuracy: Despite its efficiency, MobileNetV2 is known for maintaining a high level of accuracy. In the context of this code, it ensures that the recognition of fruits and vegetables is not compromised, and the model can make precise predictions.

The following steps demonstrate how MobileNetV2 has been implemented for ingredient recognition

- **Model Architecture:** It defines the number of classes, which is 36 in this case, representing different fruits and vegetables. It creates an instance of the MobileNetV2 model with pretrained weights. MobileNetV2 is a popular CNN architecture known for its efficiency and accuracy. The layers of the pretrained model are frozen (not trainable) to retain the learned features. A custom classifier is added on top of the base model. This custom classifier includes:
    a. A global average pooling layer to reduce spatial dimensions.
    b. A dense layer with 1024 units and ReLU activation.
    c. A dropout layer with a 50% dropout rate to prevent overfitting.

d. A dense output layer with a softmax activation function to predict class probabilities.

- **Model Compilation:** The model is compiled using the Adam optimizer and categorical cross-entropy loss. The "accuracy" metric is also monitored during training.
- **Data Augmentation**: Data augmentation is applied to the training dataset using an ImageDataGenerator. Data augmentation techniques include rescaling, rotation, shifting, shearing, zooming, horizontal flipping, and specifying how to fill missing pixels. This helps to reduce overfitting by increasing the size of our dataset.
- **Model Training:** The model is trained using the fit method. It trains for 10 epochs on the training data.

2) Convolutional Neural Network:

A CNN is used as a second model for image classification, specifically for classifying different fruits and vegetables based on input images. The model architecture defined in the code consists of convolutional layers for feature extraction and classification layers for making predictions.
The CNN is trained on a dataset of fruit and vegetable images. During training, it learns to recognize patterns and features that distinguish one fruit or vegetable from another. After training, the model can be used to predict the class labels of new, unseen images, making it suitable for tasks like automated image classification and object recognition.
In this context, the CNN serves as a powerful tool for automating the identification of fruits and vegetables, which can have various applications, including quality control in agriculture, inventory management in grocery stores, and automated sorting in food processing industries.

The following are the steps to creating and training a custom CNN model for ingredient recognition:

- **Model Architecture Definition**: A sequential CNN model is created using the Convolutional layers, max-pooling layers, and dense layers are added to create a feedforward architecture. Batch normalization is applied for better convergence.
- **Convolutional Layers**: Convolutional layers are added. These layers are responsible for learning spatial features from the input images. The activation function used is ReLU (Rectified Linear Unit).

- **Max-Pooling Layers**: Max-pooling layers are used to down-sample the feature maps. This reduces computational complexity and helps the model generalize better.
- **Flatten Layer**: After convolution and pooling layers, a flatten layer is added to convert the 2D feature maps into a 1D vector, which can be fed into the dense layers.
- **Dense Layers**: Dense layers are added for classification. These layers learn to make predictions based on the features extracted by the previous layers. ReLU activation is applied to these dense layers.
- **Dropout Layer**: A dropout layer is included for regularization. It randomly sets a fraction of input units to 0 during each update, preventing overfitting.
- **Output Layer**: The final dense layer has 36 units, corresponding to the number of classes (fruits and vegetables) in the dataset. The activation function is softmax, which provides probabilities for each class.
- **Model Compilation**: The model is compiled using the 'Adam' optimizer, categorical cross-entropy loss function (appropriate for multi-class classification), and accuracy as the evaluation metric.
- **Data Augmentation**: The training data is augmented with operations such as rescaling, rotation, shifting, shearing, zooming, and horizontal flipping.
- **Data Splitting**: The dataset is split into training, validation, and test sets. The validation set is used for monitoring the model's performance during training.
- **Training the Model**: The model is trained using the **fit** method. Training and validation data are provided, and the number of training epochs (36) is specified. The history of training is recorded.

B) Recipe Recommendation:

1) **Word2Vec:**

Model Training:

The Word2Vec model, specifically the Continuous Bag of Words (CBOW) variant, plays a pivotal role in transforming ingredients into continuous vector representations in a high-dimensional space. Here's a breakdown of this training process:

- **Model Variant:** The Word2Vec model, specifically CBOW, is selected for training. CBOW focuses on predicting a target word based on the context words that surround it. It is used to learn the semantic relationships between ingredients.
- **Training Data (Corpus):** The training process requires a corpus, which is a collection of ingredient lists represented as documents. Each ingredient list is essentially a list of words. This corpus is passed to the Word2Vec model for training**.**

Parameters:

a. **sg=0:** The parameter sg stands for "skip-gram." When set to 0, it signifies the utilization of the CBOW model. This choice is fundamental for the system to implement CBOW effectively.

b. **workers=8:** The workers parameter specifies the number of CPU cores allocated for training the Word2Vec model. Utilizing multiple cores can significantly speed up the training process.

c. **window=get_window(corpus):** The window parameter is set using the get_window function. This function calculates the average length of the documents (ingredient lists) in the corpus. The window size in CBOW represents the number of context words considered when predicting the target word. It's adjusted based on the document length to capture the appropriate context.

d. **min_count=1:** The min_count parameter is crucial for determining the minimum word frequency required for a word to be included in the vocabulary. In the system, words with a frequency of less than 1 are excluded from consideration.

e. **vector_size=100:** This parameter defines the dimensionality of the word vectors. In this case, each word is represented as a 100-dimensional vector. The vector size impacts the granularity and capacity of the model, and in this context, it is set to 100 dimensions.

Through this Word2Vec model training, our system equips itself with the ability to understand and represent ingredients in a continuous vector space. It captures the semantic relationships between ingredients, allowing for the comparison and recommendation of recipes based on ingredient similarity. The chosen parameters, such as the model variant and vector size, are pivotal in shaping the effectiveness of the trained model.

Similarity Scoring and Document Embeddings: This phase involves the calculation of document embeddings, which are crucial for comparing user input with the existing recipe dataset. The system attempted two approaches for generating these embeddings:

- **MeanEmbeddingVectorizer:**

  Document Embeddings Calculation: This method calculates document embeddings by averaging the word embeddings of ingredients within each recipe. Essentially, it computes the average vector representation of the ingredients in a recipe, resulting in a single vector representing the entire recipe.

  Use Case: This approach is suitable for creating a simple, yet effective, document representation based on the mean of ingredient vectors.

- **TfidfEmbeddingVectorizer:**

  Document Embeddings Calculation: This method calculates document embeddings using Term Frequency-Inverse Document Frequency (TF-IDF) weighting. It assigns weights to words (ingredients) based on their importance in the document. Rare ingredients are given more significance.

  Use Case: TF-IDF-based document embeddings are effective when we want to emphasize the importance of unique or distinguishing ingredients in the recipe.

Recipe Recommendation Function: The primary recipe recommendation function is the heart of the system, responsible for making personalized recipe suggestions based on user input. Here's how it works:

- **Word2Vec Model Loading:** The function loads the pre-trained Word2Vec model that was previously trained on a dataset of recipes. This model contains the knowledge of ingredient relationships.
- **Normalization:** The system normalizes the embeddings to ensure consistent and meaningful comparisons between ingredients.
- **Data Processing:** The user provides a list of available ingredients as input. The function processes this input and calculates document embeddings for user input. Depending on the chosen method (TF-IDF weighted, as it was more accurate), the user's ingredients are represented as a single vector.

- **Cosine Similarity Scores:** The function calculates cosine similarity scores between the user's input embedding and the embeddings of all recipes in the dataset. This measures how closely the user's ingredient preferences align with each recipe.
- **Top N Recommendations:** The system selects the top N recipe recommendations by sorting them based on their similarity scores. This ensures that the most relevant and appealing recipes are presented to the user.

2) **Long Short-Term Memory:**

The 'parsed' ingredients are prepared in the preprocessing part, which cleans, tokenizes, and lemmatizes the individual words. This processed ingredient list is used as input for the LSTM model.

The Tokenizer class from the Keras library is used to tokenize the ingredients. This means converting each word into a unique integer. A vocabulary is created, mapping words to integer IDs. The size of this vocabulary is determined by the number of unique words in the ingredient lists.

Model Training:

- **Word Embeddings:**

An Embedding layer is created in a Keras Sequential model. This layer is responsible for mapping the integers (word IDs) to dense vectors of a fixed size. It's similar to Word2Vec, but in this case, it's learned within the LSTM model. By training the model to generate word embeddings, it learns to capture the semantic relationships between ingredients. For example, it might learn that "tomato" and "tomatoes" are related terms and should have similar embeddings. These word embeddings help convert ingredient text into a numerical format that the model can work with.

Parameters:

a. **Input dimension:** Size of the vocabulary
b. **Output dimension:** Size of the word embeddings
c. **Input length:** length of each input sequence, which is the maximum number of words in a recipe.

The Embedding layer is used to create word embeddings for all words in the ingredients. These embeddings capture the semantic relationships between words in the context of the recipes.

- **Document Embeddings:**

An LSTM layer is added to the Sequential model. This LSTM layer takes the word embeddings as input and is responsible for creating document embeddings (representations of entire recipes).

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is particularly good at handling sequential data. In this case, it's used to understand the context and relationships between ingredients in a recipe.

The LSTM layer typically has a certain number of units or neurons, which can be adjusted to control the complexity and capacity of the model.

After processing the entire ingredient list through the Embedding and LSTM layers, the LSTM layer outputs a document embedding. This embedding represents the entire recipe, capturing the contextual information between ingredients.

The word and document embeddings serve as numerical representations of recipes and ingredients.

Parameters:

a. **Number of LSTM Units**: The number of LSTM units (also known as neurons or cells) determines the model's capacity to capture complex patterns.

b. **Number of LSTM Layers**: Single or multiple LSTM layers stacked on top of each other can be used to create a deeper network. Deeper networks can capture hierarchical features in the data but require more computational resources.

c. **Activation Functions**: This project employed the tanh function, which is the default activation function in Keras.

Recommendation Functions: In this phase, the system defines several functions to process and format recipe recommendations:

- **LSTM Model Loading:** The function loads the trained LSTM model that was previously trained on a dataset of recipes. This model contains the knowledge of ingredient relationships.
- **Cosine Similarity Scores:** The function calculates cosine similarity scores between the user's input embedding and the embeddings of all recipes in the dataset. This measures how closely the user's ingredient preferences align with each recipe.

- **Cleaning and Formatting:** These functions are responsible for tidying up the recipe titles and ingredient lists, making them more user-friendly and presentable. This ensures that users receive well-structured recommendations.
- **Ranking and Extraction:** The system implements a function to rank recipes based on similarity scores. It extracts the top N recommendations, presenting them to users. These recommendations include details such as recipe names, ingredients, URLs, cuisine, and their respective similarity scores.

3) BERT (Bidirectional Encoder Representations from Transformers):

Pre-trained BERT Model:

We chose to use a pre-trained BERT model, specifically " bert-base-nli-mean-tokens" from the Sentence Transformers library. BERT models are pre-trained on extensive text data, allowing them to understand the context and meaning of words in a given text.

BERT is a language model pre-trained on vast text data. It understands word meanings in context, considering words bidirectionally as it uses the powerful Transformer neural network architecture. It also consists of multiple layers for deep contextual understanding.

BERT generates context-aware word embeddings, unlike traditional static embeddings.

To process text effectively, BERT tokenizer was used. This tokenizer is designed to split text into smaller units, or tokens, making it suitable for BERT's input requirements. It can also manage special tokens used for structuring input.

Token Embeddings

- **Input Tokens:** The function takes a list of tokens as input. These tokens represent the textual elements for which embeddings have to be created. For instance, for ingredients, the tokens would be ingredient names.
- **Token-to-ID Conversion:** BERT models cannot work with raw text directly, so the input tokens are converted into numerical input IDs. Each token is mapped to a unique integer ID that the model understands.
- **Generating Token Embeddings:** To get embeddings, these input IDs are passed to the BERT model. The model processes these IDs and generates embeddings for each token.

These embeddings are high-dimensional numerical representations capturing the context and meaning of each token.

- **Embedding Output:** The function returns these embeddings as the output. These embeddings are numerical vectors, and each vector represents the context and semantics of the corresponding token.
- The code uses the FuzzyWuzzy library to perform fuzzy string matching when identifying recipe names. This helps find the closest matching recipe name in case there are slight variations or typos in the input.

Recommendation Functions: In this phase, the system defines several functions to process and format recipe recommendations:

- **BERT Model Loading:** The function loads the pre-trained BERT model that was previously trained on a dataset of recipes. This model contains the knowledge of ingredient relationships.
- **Cosine Similarity Scores:** The function calculates cosine similarity scores between the user's input embedding and the embeddings of all recipes in the dataset. This measures how closely the user's ingredient preferences align with each recipe.
- **Cleaning and Formatting:** These functions are responsible for tidying up the recipe titles and ingredient lists, making them more user-friendly and presentable. This ensures that users receive well-structured recommendations.
- **Ranking and Extraction:** The system implements a function to rank recipes based on similarity scores. It extracts the top N recommendations, presenting them to users. These recommendations include details such as recipe names, ingredients, URLs, cuisine, and their respective similarity scores.

4) FastText (Gensim):

FastText is a popular word embedding model developed by Facebook's AI Research (FAIR) lab. It is an extension of the Word2Vec model but with some key differences and improvements. The primary goal of FastText is to learn continuous vector representations (embeddings) for words or sub-words in a text corpus.

FastText takes into account not only whole words but also sub-word units, such as character n-grams and sub-tokens. This sub-word information enables the model to understand complex or out-of-vocabulary ingredients in recipes..

It is context-aware, meaning it considers the surrounding words when learning word embeddings, which is vital for recognizing ingredient meanings within recipes.

Furthermore, FastText uses a hierarchical soft-max approach to speed up training and make it efficient for large ingredient vocabularies.

Model Training:

The model's training process encompasses the following elements:

- **Contextual Learning:** FastText excels in understanding words in context, considering not just isolated tokens but their contextual surroundings. This is crucial for capturing the meaning of ingredients in recipes.
- **Semantic Significance:** Through training, FastText generates dense vector representations for tokens based on their dataset usage, enabling it to comprehend relationships and similarities between words or ingredients. For instance, it discerns that "tomato" and "bell pepper" share a closer connection than "tomato" and "computer."
- **Word Embedding Mastery:** The training fine-tunes model parameters by predicting the context of a target word. This process results in word embeddings rich in semantic content, valuable for various natural language tasks, including similarity assessments and recommendations.
- **Continuous Vector Space:** FastText establishes a coherent vector space where tokens with akin meanings are positioned closer. This spatial arrangement facilitates precise similarity calculations, underpinning our project's recipe recommendations based on ingredient similarity.

Parameters:

a. **ingredient_tokens:** This is the tokenized list of ingredients from our dataset.

b. **vector_size=100:** It specifies the dimensionality of the embeddings. In this case, the embeddings are 100-dimensional vectors. A higher dimension allows for more detailed representations

s

Creating Query Ingredient Embedding: Once the model was trained, it was used to create embeddings for a query ingredient list.

- **Tokenizing the Query:** The query ingredients are first tokenized, just like the training ingredients. This ensures that each ingredient in the query is represented as a sequence of tokens.
- **Embedding Calculation:** The code calculates an embedding for the entire query ingredient list by averaging the embeddings of the individual ingredient tokens. For each ingredient token, the corresponding embedding is retrieved from the trained FastText model.
- **Averaging Embeddings:** To create the query ingredient embedding, the embeddings of all the ingredient tokens are averaged. This process generates a single vector that represents the semantic content of the entire query.

The resulting query ingredient embedding can be used to compare the similarity between the query and the recipes in the dataset.

Recommendation Functions: In this phase, the system defines several functions to process and format recipe recommendations:

- **FastText Model Loading:** The function loads the trained FastText model that was previously trained on a dataset of recipes. This model contains the knowledge of ingredient relationships and the query embedding.
- **Cosine Similarity Scores:** The function calculates cosine similarity scores between the user's input embedding and the embeddings of all recipes in the dataset. This measures how closely the user's ingredient preferences align with each recipe.
- **Cleaning and Formatting:** These functions are responsible for tidying up the recipe titles and ingredient lists, making them more user-friendly and presentable. This ensures that users receive well-structured recommendations.
- **Ranking and Extraction:** The system implements a function to rank recipes based on similarity scores. It extracts the top N recommendations, presenting them to users. These recommendations include details such as recipe names, ingredients, URLs, cuisine, and their respective similarity scores.

5) Logistic Regression:

Logistic Regression is a classification algorithm that's well-suited for tasks where the output is a category or class label, as in this case:

The Logistic Regression model is trained using the training set. It learns to make predictions based on the TF-IDF features of ingredients and the corresponding recipe names.

After training, the model can predict the most likely recipe name for a given set of ingredient features. It computes a probability distribution over all possible recipe names and selects the one with the highest probability.

Then model assigns a recipe name to the input based on the class (recipe name) with the highest predicted probability.

TF-IDF Vectorization:

TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a technique used to convert text data into numerical features. Here's how it works:

- **Term Frequency (TF):** This part measures the frequency of a word within a document (recipe in this case). It gives higher weights to words that appear more frequently in the document.
- **Inverse Document Frequency (IDF):** This part evaluates the importance of a word across all documents (recipes). Words that are common across many documents get lower weights, while words that are unique to a few documents receive higher weights.
- **TF-IDF Score:** The final score for a word in a specific document is the product of its TF and IDF scores. This means that words that are frequent in a particular recipe but rare across other recipes get higher scores, making them more distinguishing.
- **Vectorization:** TF-IDF vectorization converts each recipe (document) into a numerical vector. Each dimension of this vector corresponds to a unique word (ingredient). The value in each dimension is the TF-IDF score of the ingredient in the recipe.

Training Set: This is used to train the Logistic Regression model. The model learns from this data by adjusting its parameters to make accurate predictions based on the input features.

Testing Set: This set is used to evaluate the model's performance. The model hasn't seen this data during training, so it's a good measure of how well it generalizes to new, unseen examples. By comparing the model's predictions on the testing set to the actual recipe names, we assessed its accuracy.

3.4 Additional Features

A) Recipe Generator:

In today's data-driven world, artificial intelligence and natural language processing have found applications in various domains. One intriguing area is recipe generation, where AI models like GPT-2 (Generative Pre-trained Transformer 2) can transform a list of ingredients into a full-fledged culinary recipe. Therefore, as an additional feature in our project, we enable users to use AI-generated recipes as well, using the ingredients input.

GPT-2, short for "Generative Pre-trained Transformer 2," is a state-of-the-art natural language processing model developed by OpenAI. It is a deep learning model that belongs to a class of models known as transformers. It is pre-trained on massive amounts of text data from the internet, enabling it to understand and generate human-like text.

Key Characteristics:

- **Language Generation:** GPT-2 is a text generator. It can generate coherent and contextually relevant text based on the input it receives. This makes it useful for a wide range of natural language understanding and generation tasks.
- **Scalability:** GPT-2 comes in different sizes, ranging from a small model with fewer parameters to a very large model with hundreds of millions of parameters. The larger models have a better understanding of context and can generate more coherent text.
- **Contextual Understanding:** GPT-2 can capture contextual relationships between words and phrases in a given text, which allows it to generate text that makes sense within a specific context.
- **Fine-tuning:** While GPT-2 is pre-trained on diverse internet text, it can also be fine-tuned for specific tasks, making it adaptable to various applications such as text completion, translation, summarization, and more.

The following are the important components of our implementation:

- **Data Collection and Preparation:** The first step is gathering a dataset that consists of recipes. This dataset contains recipe names, lists of ingredients, and cooking instructions. Data is a crucial component in training of GPT-2 in our project, as the model learns from the patterns and information within it. To ensure that the model isn't biased, the dataset is shuffled randomly.
- **Text Formatting:** To effectively train a language model, text formatting is essential. The "print_recipe" function is used to display the details of a recipe. It helps visualize what the generated recipes will look like. The "form_string" function is responsible for combining the ingredients and cooking instructions into a single, coherent text format. This formatted text is what the model will process.
- **Data Split:** The dataset is divided into two parts: a training set and a validation set. This is done to assess the model's performance during training and ensure that it generalizes well to unseen data. An 85-15% split is chosen, which reserves 85% of the data for training.
- **Custom Dataset Class:** To prepare the data for GPT-2 training, a custom "RecipeDataset" class is created. This class is designed to process and encode the data. It produces input IDs and attention masks, which are necessary for training the GPT-2 model.
- **Collate Function:** Batch processing is a crucial element of training efficiency. The "collate_fn" function is responsible for organizing the data into batches, making the training process more streamlined and memory-efficient
- **Training Configuration**: Proper configuration of the training process is vital. This includes setting up training arguments such as the output directories, batch sizes, gradient accumulation strategies, the number of training epochs, and the strategy for saving model checkpoints.
- **Model Training:** Training a sophisticated model like GPT-2 involves several steps. The code initializes an AdamW optimizer, which is a variant of the standard Adam optimizer, tailored for training deep learning models. Additionally, a learning rate scheduler is implemented to dynamically adjust the learning rate during training. The "Trainer" class from the transformers library is employed to manage the entire training workflow, encompassing data loading, model optimization, and checkpoint saving.
- **Model Saving:** A crucial step post-training is to save the trained GPT-2 model. This is essential for preserving the model's fine-tuned parameters and ensuring that the knowledge gained during training is retained for future use.

- **Text Generation:** After successful model training, the code is equipped to generate recipe instructions. A "create_prompt" function is introduced to format a list of ingredients into a proper prompt. With this prompt, the GPT-2 model is capable of generating detailed culinary instructions, making it an invaluable tool for cooking enthusiasts.

B) Edamam API

The integration of the Edamam API with our frontend represents a significant achievement in enhancing the user experience and providing valuable nutritional insights. With this connection, users can seamlessly access nutritional information directly through the frontend interface. This means that when a user inputs a recipe or food item, they receive instant, detailed data on its nutritional content. This information is presented in a user-friendly format, making it easy for individuals to make informed decisions about their dietary choices. The integration also enables real-time updates, ensuring that users receive the most accurate and up-to-date nutritional information available. By connecting the Edamam API to our frontend, we've created a valuable tool for users to achieve their health and wellness goals while using our application.

3.5 Project Setup

In this section, we provide a comprehensive overview of the initial setup and configuration of our project. This segment serves as a crucial starting point for readers and stakeholders to understand the foundation of our work.

By presenting a clear and concise account of our project's setup, we aim to make our work accessible and reproducible for both technical and non-technical audiences.

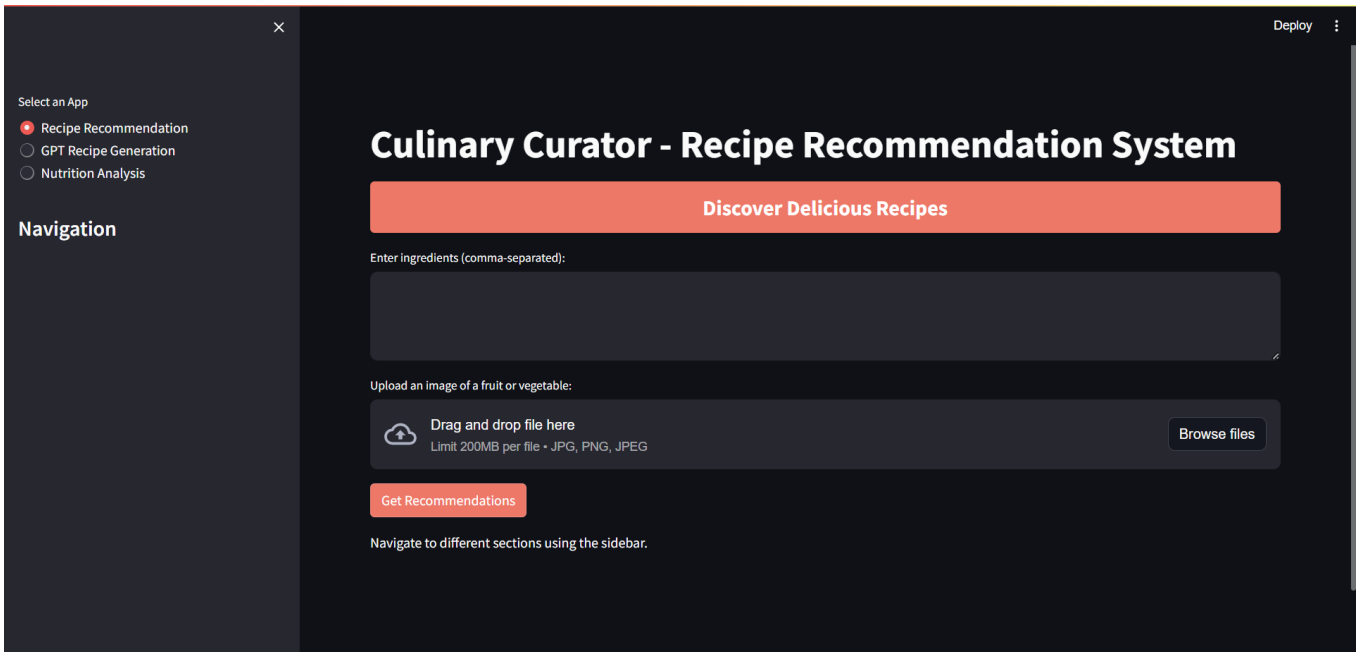

Figure 3.5.1 Website of Culinary Curator for Recipe Recommendation

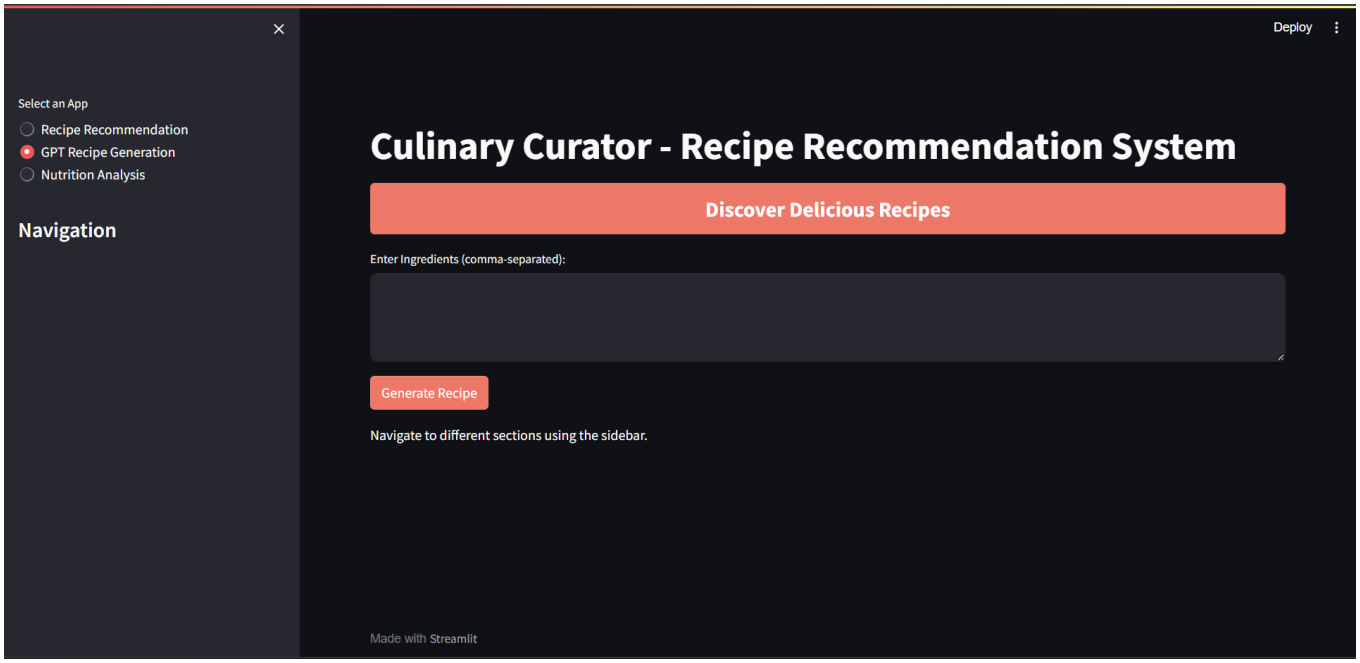

Figure 3.5.2 Website of Culinary Curator for GPT Recipe Generation

Figure 3.5.3 Website of Culinary Curator for Nutrition Analysis

Figure 3.5.4 Website of Culinary Curator for Recipe Recommendation (with User Text Input)

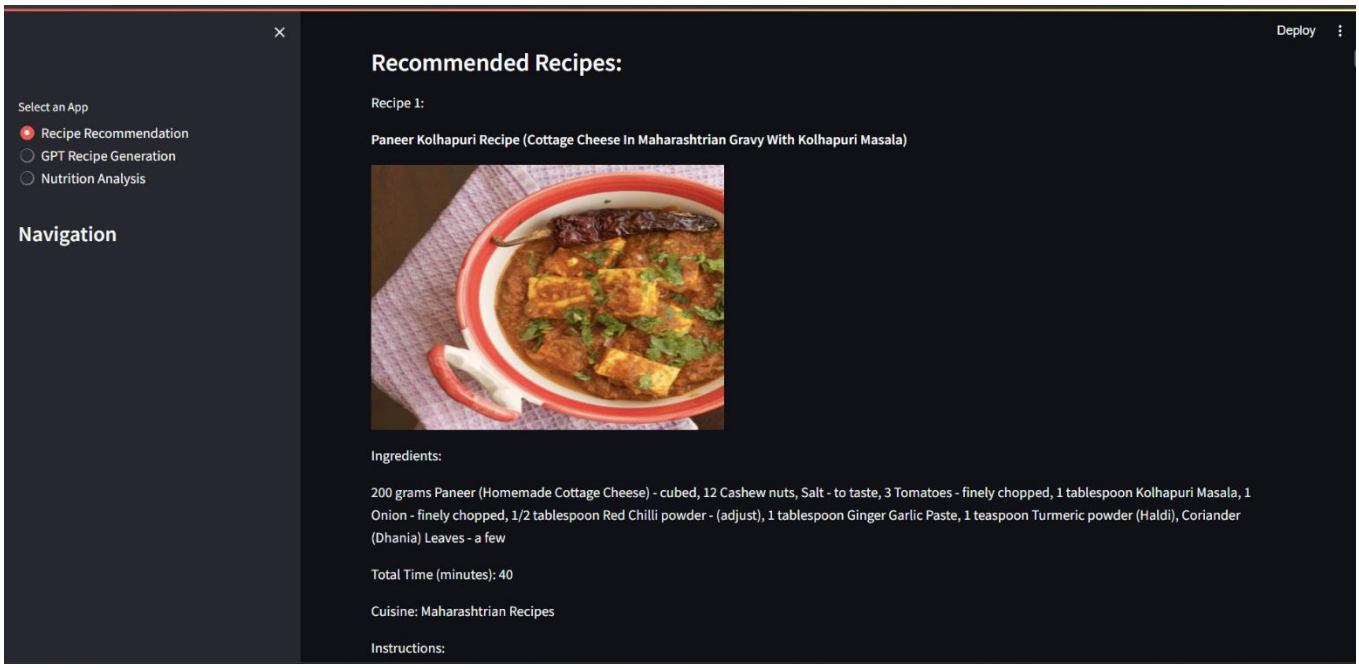

Figure 3.5.5 Website of Culinary Curator for Recipe Recommendation (With recommended recipe)

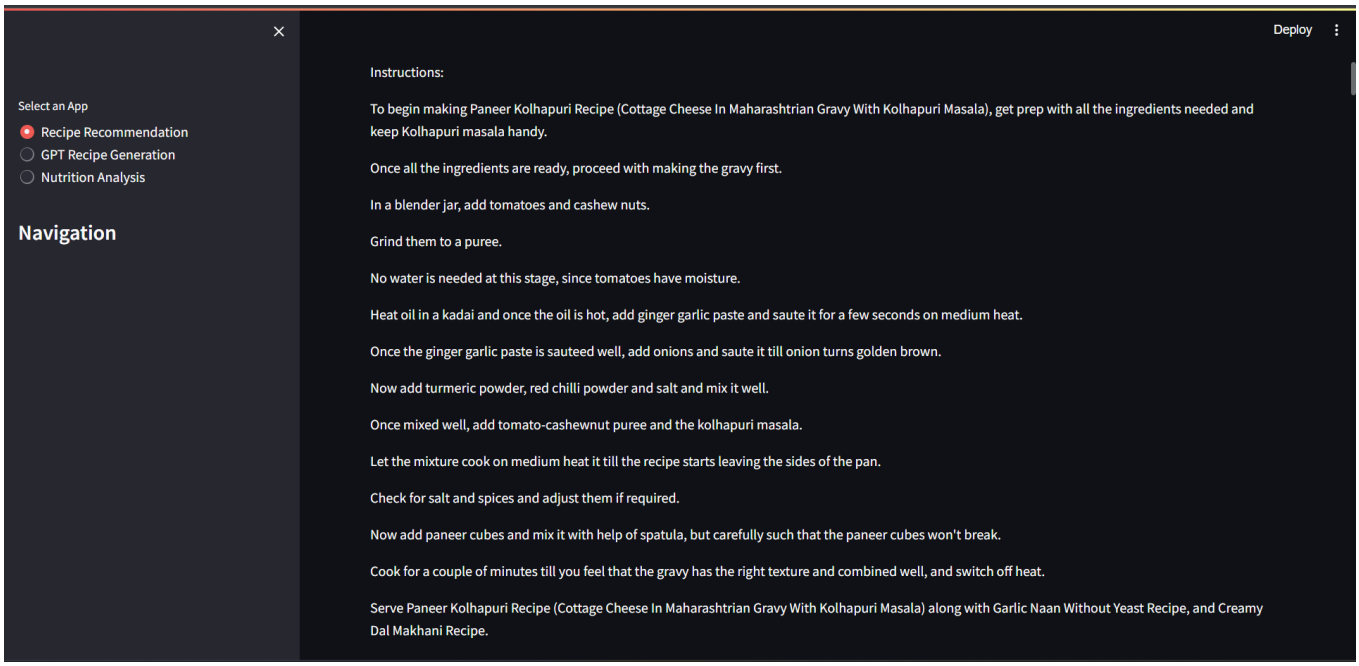

Figure 3.5.6 Website of Culinary Curator for Recipe Recommendation (With recommended recipe)

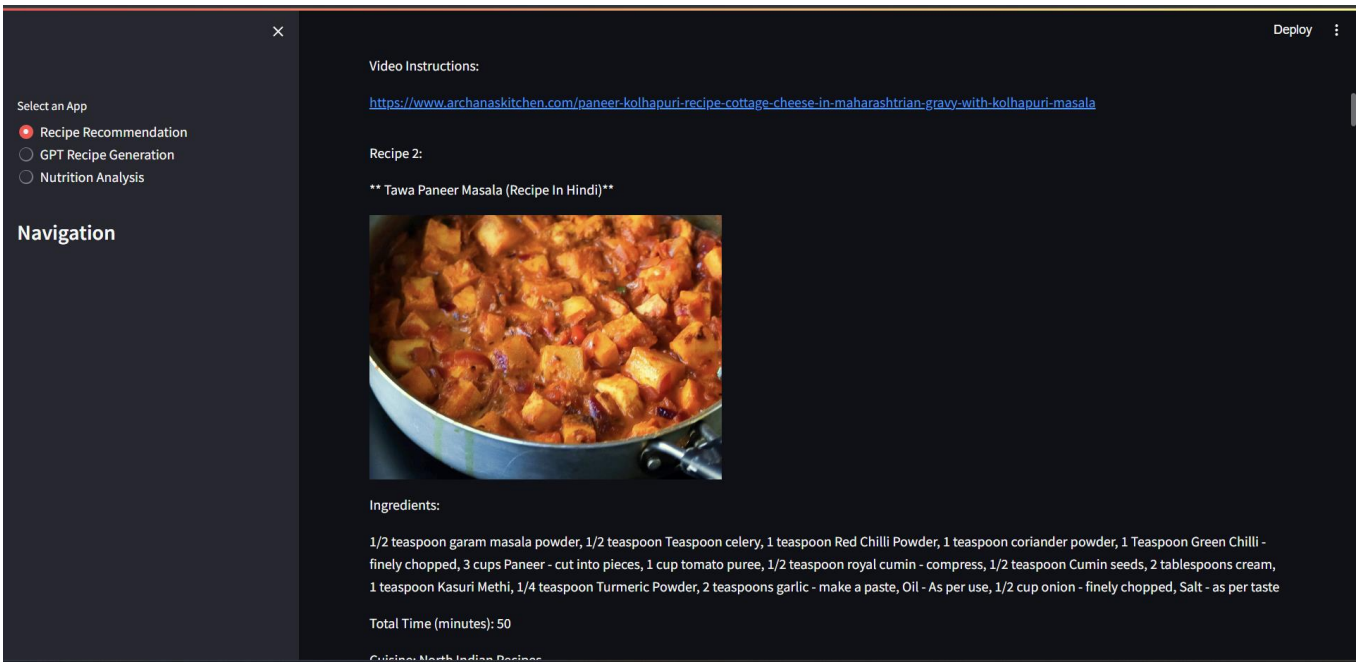

Figure 3.5.7 Website of Culinary Curator for Recipe Recommendation (With recommended recipe output)

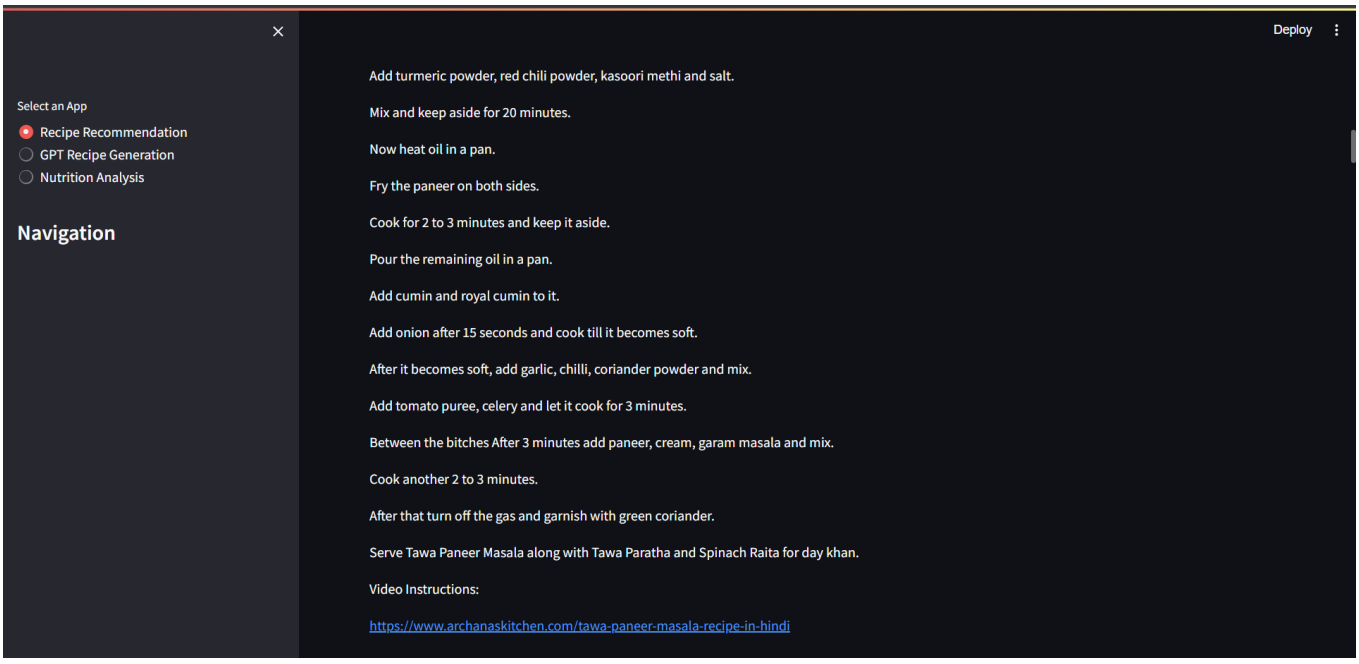

Figure 3.5.8 Website of Culinary Curator for Recipe Recommendation (With recommended recipe output)

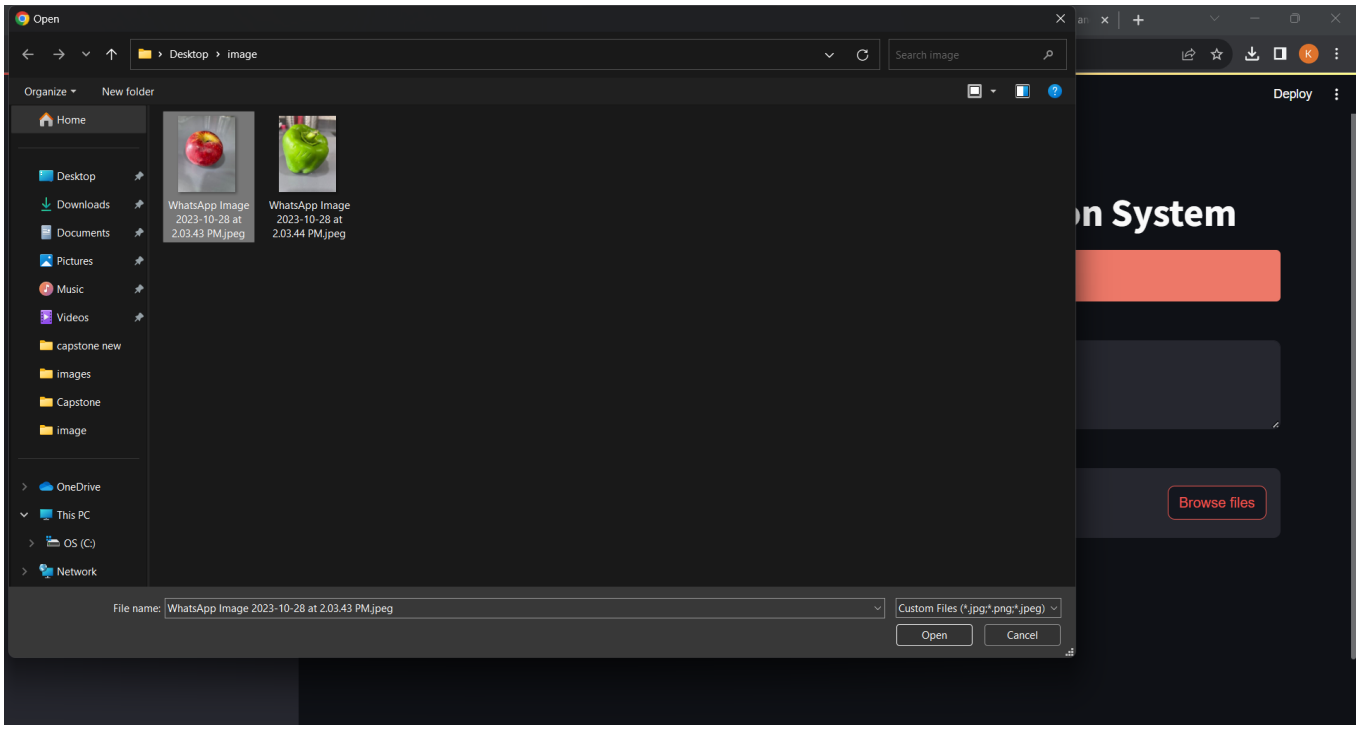

Figure 3.5.9 Website of Culinary Curator for Recipe Recommendation (With Image Input)

Figure 3.5.10 Website of Culinary Curator for Recipe Recommendation (With name output of ingredient recognized)

Figure 3.5.11 Website of Culinary Curator for Recipe Recommendation (With more ingredients recognized)

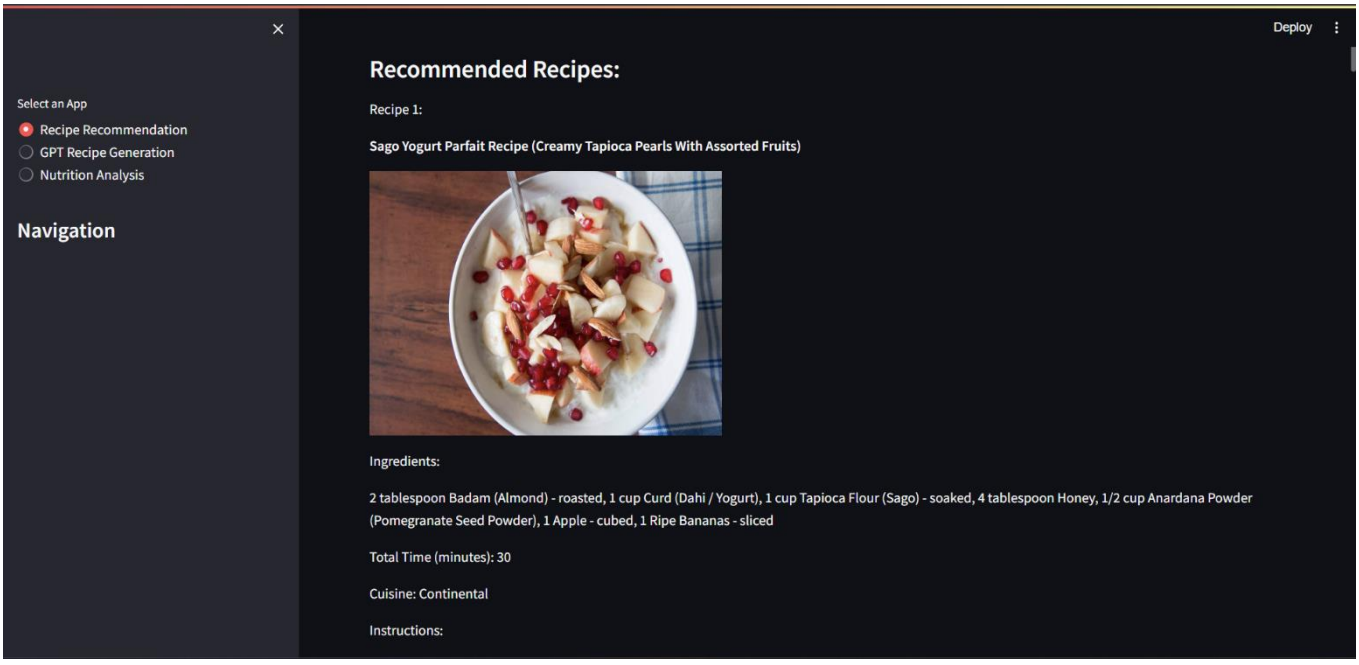

Figure 3.5.12 Website of Culinary Curator for Recipe Recommendation (With recommended recipe output for image input)

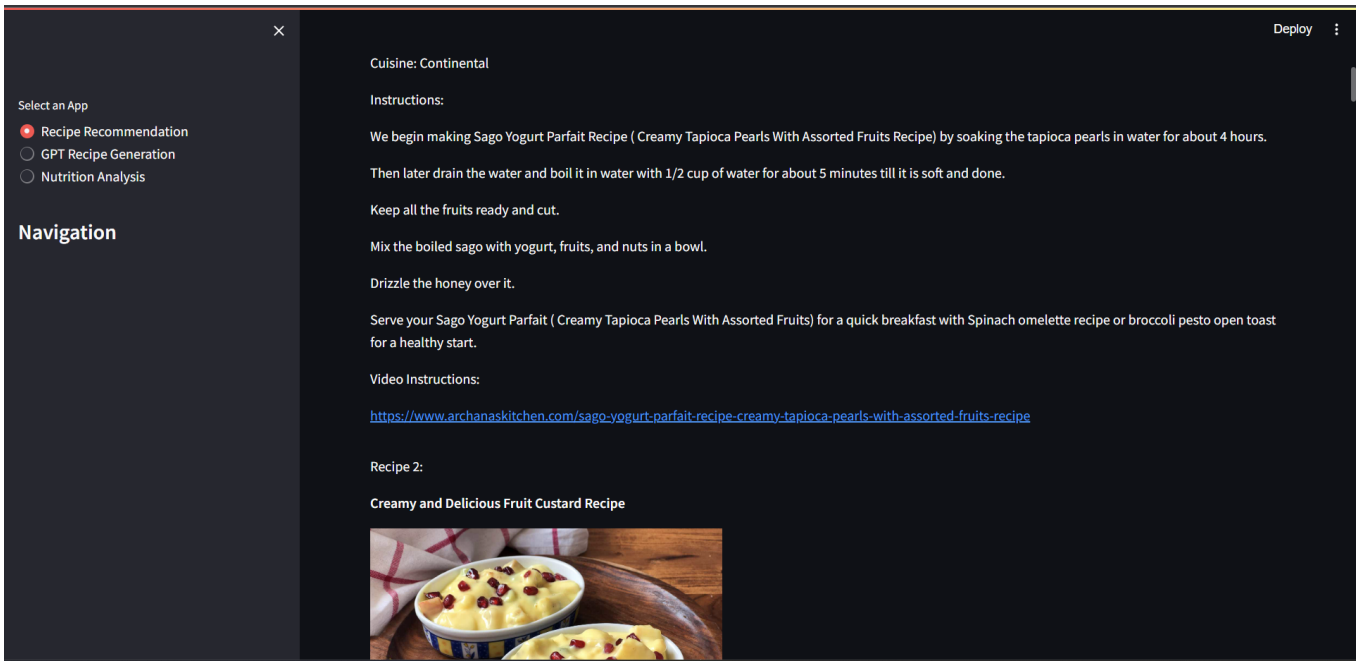

Figure 3.5.13 Website of Culinary Curator for Recipe Recommendation (With recommended recipe output for Image input)

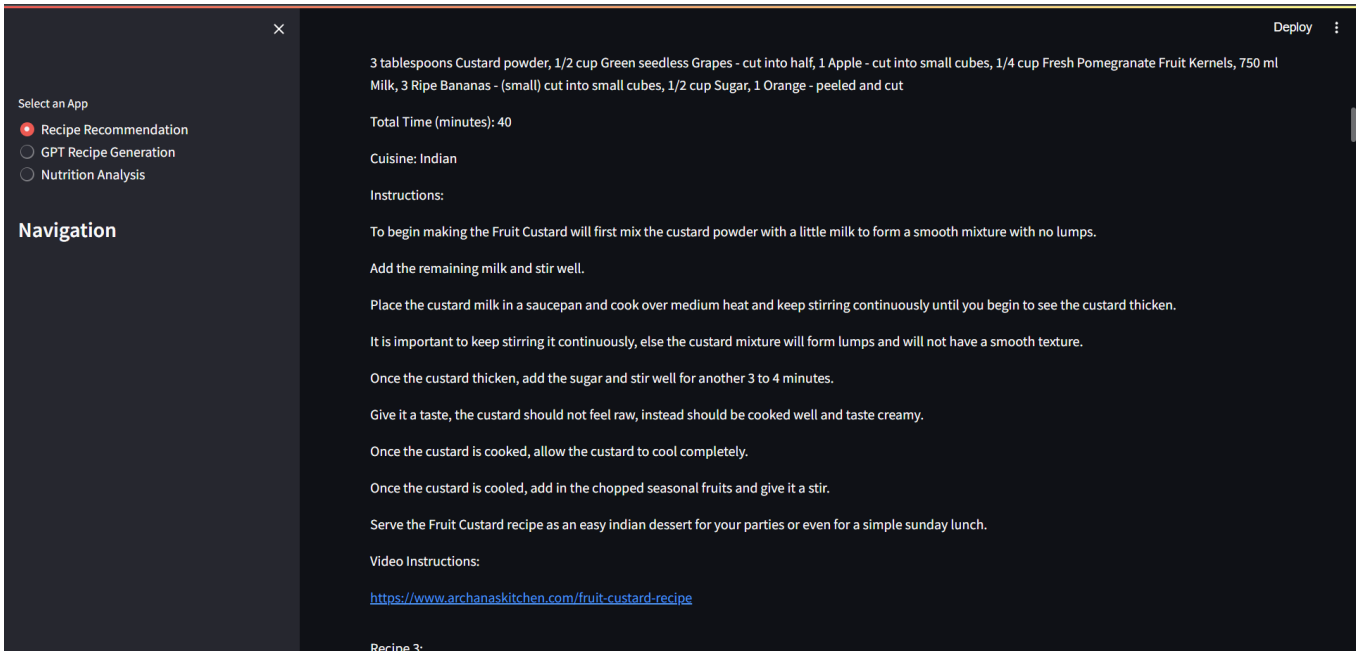

Figure 3.5.14 Website of Culinary Curator for Recipe Recommendation (With recommended recipe output for Image input)

Figure 3.5.15 Website of Culinary Curator for GPT Recipe Generator (With input)

Figure 3.5.16 Website of Culinary Curator for GPT Recipe Generator (With generated recipe output)

Figure 3.5.17 Website of Culinary Curator for Nutrition Analysis (With input)

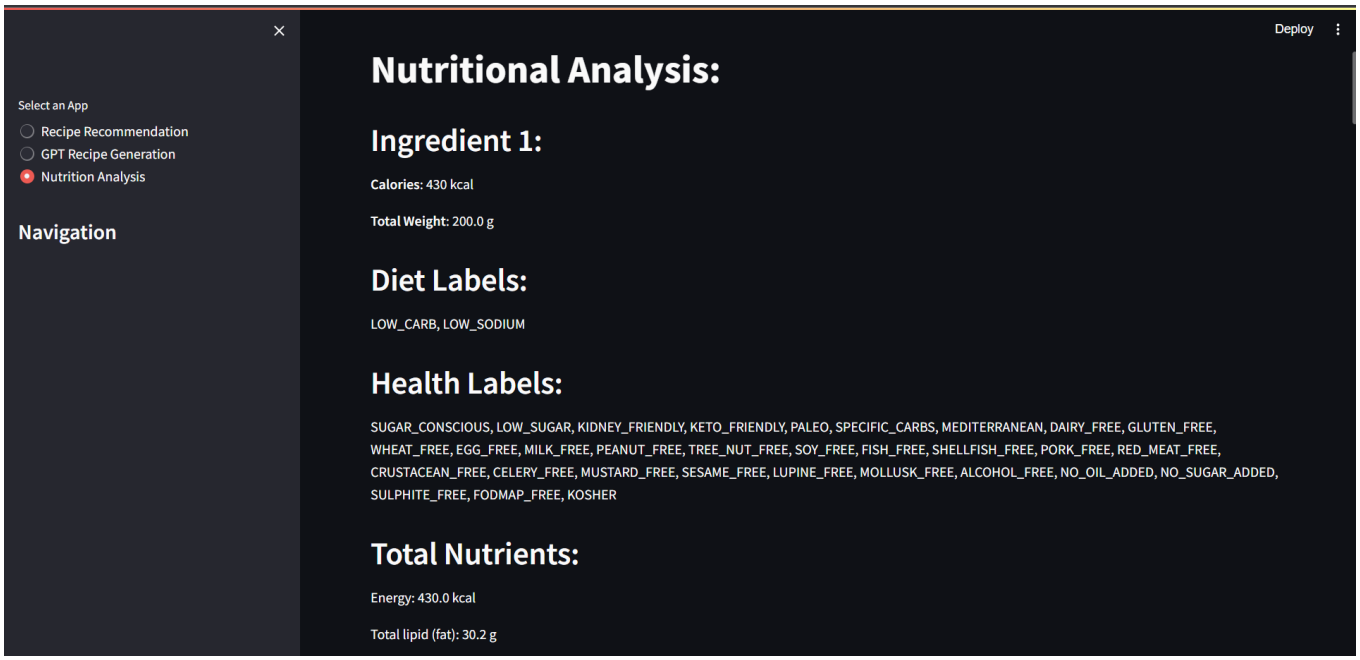

Figure 3.5.18 Website of Culinary Curator for Nutrition Analysis (with output)

Figure 3.5.19 Website of Culinary Curator for Nutrition Analysis (With output)

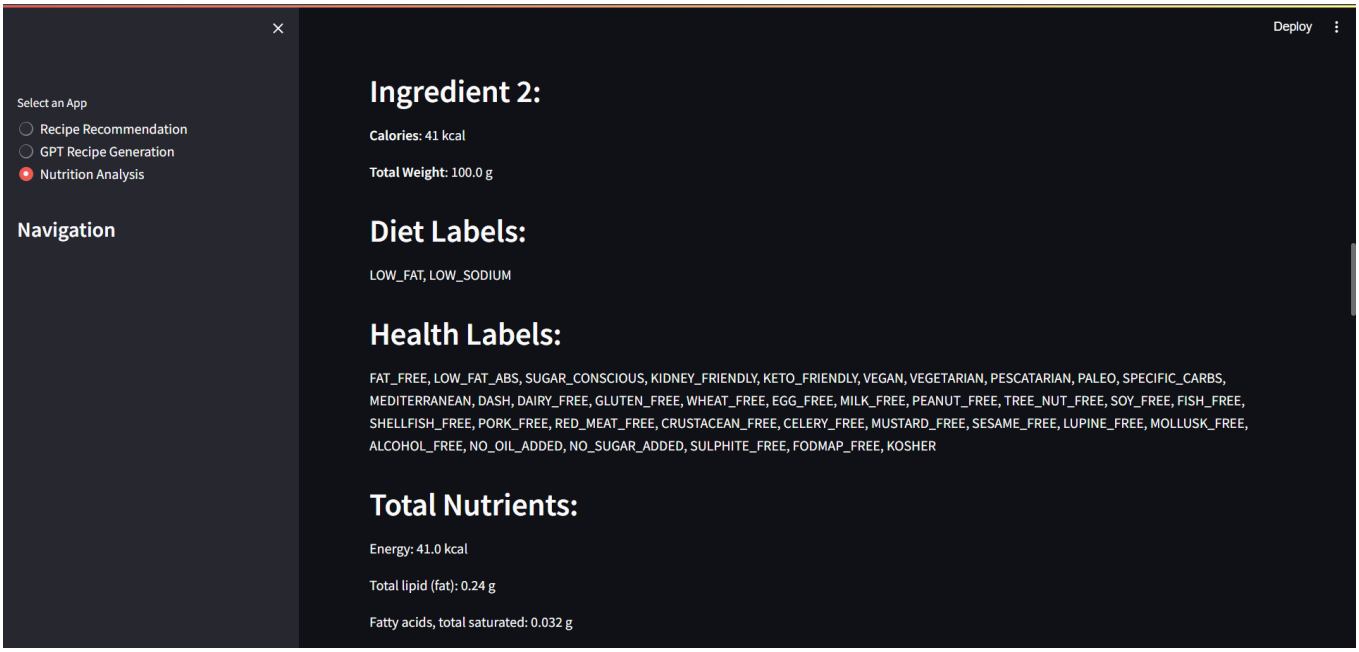

Figure 3.5.20 Website of Culinary Curator for Nutrition Analysis (With output)

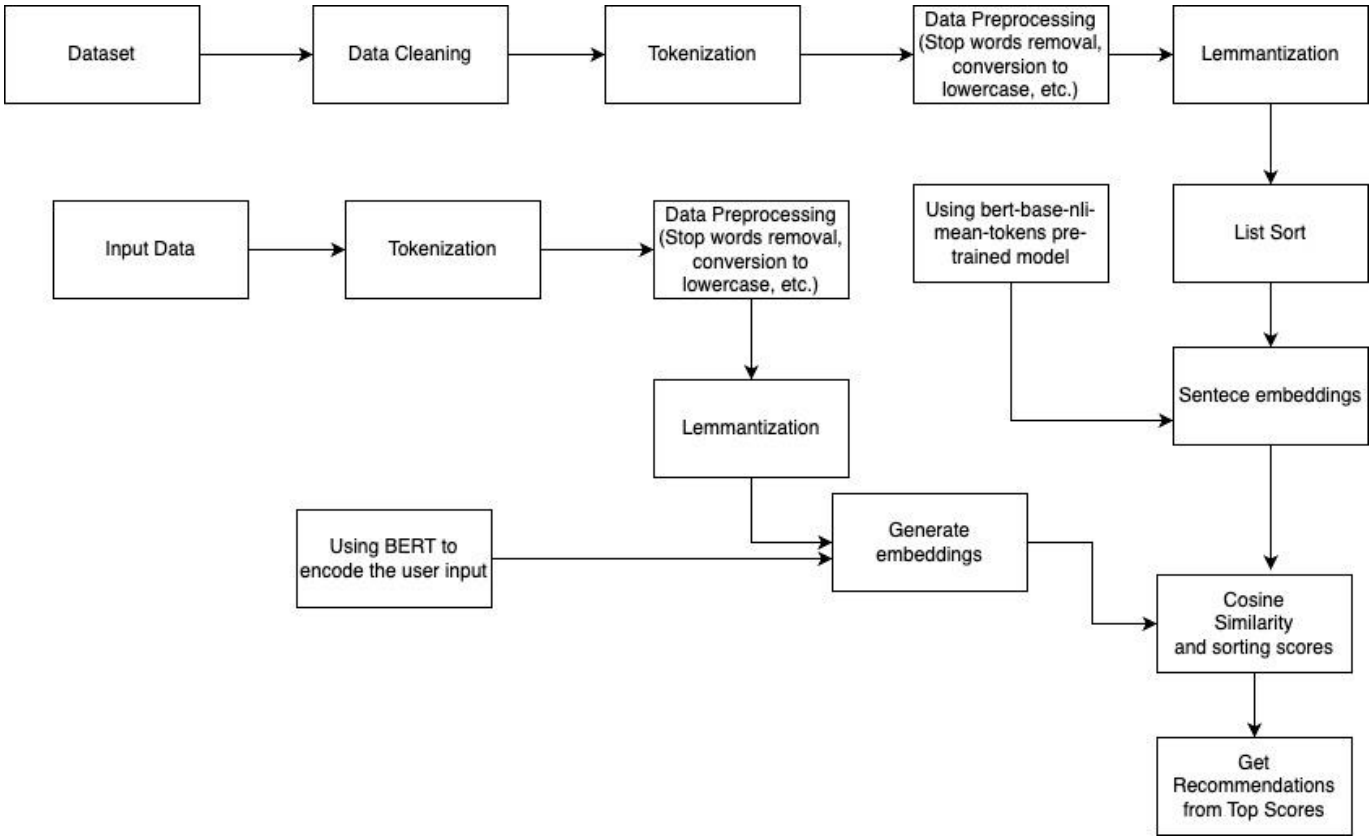

Figure 3.5.21 Model Architecture for Recipe Recommendation Using BERT

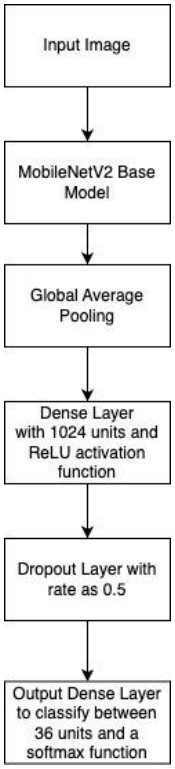

Figure 3.5.22 Model Architecture for Ingredient Recognition Using MobileNetV2

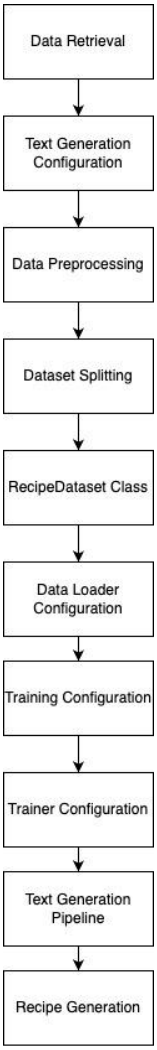

Figure 3.5.23 Model Architecture for Recipe Generation using GPT-2

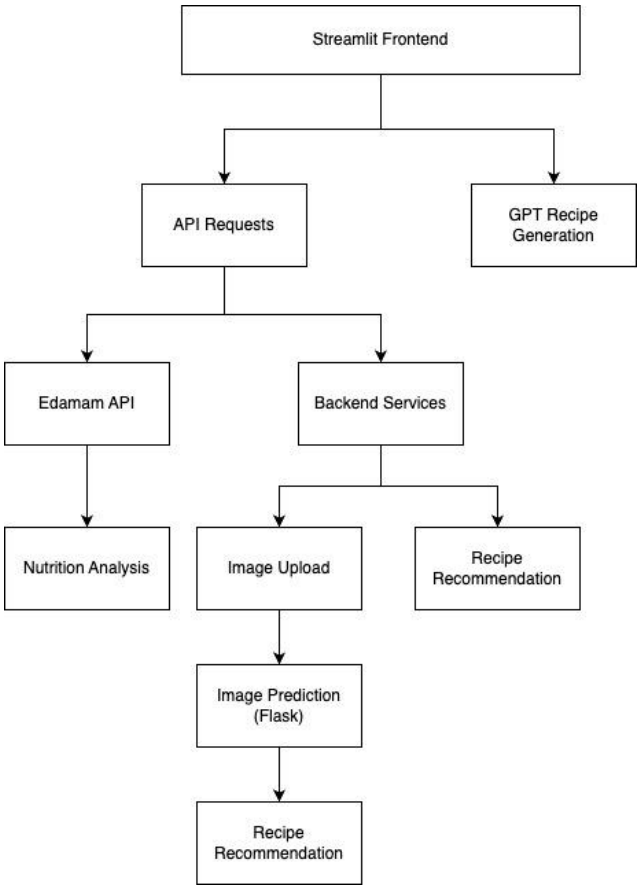

Figure 3.5.24 Model Architecture for Recipe Generation using GPT-2

Chapter 4

Results and Analysis

4.1 Performance Metrics:

A) Ingredient Recognition Models

Accuracy: This metric measures the overall correctness of the models' predictions by considering the ratio of correctly predicted instances to the total instances. It was used to assess the accuracy of ingredient recognition in images.

Precision: Precision is a crucial metric for image recognition and assesses the accuracy of the models' positive predictions. It measures how many of the instances classified as positive are true positives.

Recall: Recall evaluates the models' ability to identify all relevant instances within the dataset, measuring the ratio of true positives to the total number of actual positives.

F1-Score: The F1-Score, a harmonic mean of precision and recall, offers a balanced measure of the models' accuracy, considering both false positives and false negatives. It provides a comprehensive evaluation of performance.

B) Recipe Recommendation Models

Precision at K: Precision at K (where K = 10 in outry case) evaluates the accuracy of the top K predictions made by the recommendation models. It focuses on the most relevant recommendations within the top K predictions, ensuring the quality of the top suggestions.

4.2 Ingredient Recognition Models

A) Results

1) Custom CNN Model: The custom Convolutional Neural Network (CNN) model achieved an accuracy of 93%, indicating a strong overall performance. When examining precision, recall, and F1-scores for individual ingredient categories, the model demonstrated noteworthy results. It showcased particularly high F1-scores for some ingredients, underscoring its proficiency in recognizing these items. However, it faced challenges in the accurate identification of certain ingredients.

	precision	recall	f1-score	support
apple	0.78	0.88	0.82	8
banana	1.00	0.78	0.88	9
beetroot	1.00	1.00	1.00	10
bell pepper	0.64	0.90	0.75	10
cabbage	1.00	1.00	1.00	10
capsicum	0.83	0.50	0.62	10
carrot	1.00	0.90	0.95	10
cauliflower	1.00	1.00	1.00	10
chilli pepper	0.67	1.00	0.80	10
corn	0.77	1.00	0.87	10
cucumber	1.00	1.00	1.00	10
eggplant	1.00	1.00	1.00	10
garlic	1.00	0.90	0.95	10
ginger	1.00	1.00	1.00	10
grapes	1.00	0.90	0.95	10
jalepeno	0.83	1.00	0.91	10
kiwi	1.00	1.00	1.00	10
lemon	1.00	0.90	0.95	10
lettuce	1.00	1.00	1.00	10
mango	0.91	1.00	0.95	10
onion	0.77	1.00	0.87	10
orange	0.83	1.00	0.91	10
paprika	1.00	0.70	0.82	10
pear	1.00	1.00	1.00	10
peas	1.00	1.00	1.00	10
pineapple	1.00	1.00	1.00	10
pomegranate	1.00	1.00	1.00	10
potato	1.00	0.70	0.82	10
raddish	1.00	1.00	1.00	10
soy beans	1.00	0.90	0.95	10
spinach	1.00	1.00	1.00	10
sweetcorn	1.00	0.70	0.82	10
sweetpotato	1.00	0.90	0.95	10
tomato	1.00	1.00	1.00	10
turnip	1.00	1.00	1.00	10
watermelon	1.00	1.00	1.00	10
accuracy			0.93	357
macro avg	0.95	0.93	0.93	357
weighted avg	0.95	0.93	0.93	357

Figure 4.2.1 Performance Metrics for CNN

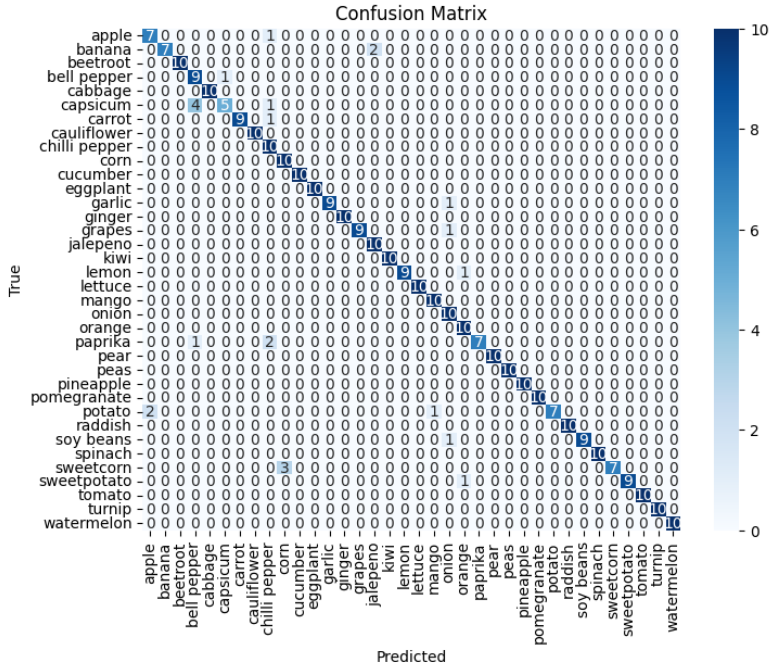

Figure 4.2.2 Confusion Matrix for CNN

2) MobileNetV2 Model: The MobileNetV2 model had an impressive accuracy of 95%. This model not only achieved high accuracy but maintained consistently strong precision, recall, and F1-score values. It excelled in recognizing a wide range of ingredients, with flawless F1-scores. The MobileNetV2 model demonstrated its robustness and proficiency in ingredient recognition.

	precision	recall	f1-score	support
apple	0.88	0.70	0.78	10
banana	1.00	0.67	0.80	9
beetroot	1.00	1.00	1.00	10
bell pepper	0.90	1.00	0.95	9
cabbage	1.00	1.00	1.00	10
capsicum	1.00	0.90	0.95	10
carrot	1.00	0.89	0.94	9
cauliflower	0.91	1.00	0.95	10
chilli pepper	1.00	0.89	0.94	9
corn	0.80	0.80	0.80	10
cucumber	1.00	1.00	1.00	10
eggplant	0.83	1.00	0.91	10
garlic	1.00	1.00	1.00	10
ginger	1.00	1.00	1.00	10
grapes	0.90	1.00	0.95	9
jalepeno	1.00	1.00	1.00	9
kiwi	0.91	1.00	0.95	10
lemon	0.71	1.00	0.83	10
lettuce	1.00	1.00	1.00	9
mango	1.00	1.00	1.00	10
onion	1.00	1.00	1.00	10
orange	1.00	0.89	0.94	9
paprika	1.00	1.00	1.00	10
pear	0.91	1.00	0.95	10
peas	0.91	1.00	0.95	10
pineapple	0.91	1.00	0.95	10
pomegranate	1.00	1.00	1.00	10
potato	0.89	0.80	0.84	10
raddish	1.00	1.00	1.00	9
soy beans	1.00	1.00	1.00	10
spinach	1.00	1.00	1.00	10
sweetcorn	0.89	0.80	0.84	10
sweetpotato	1.00	0.70	0.82	10
tomato	1.00	1.00	1.00	10
turnip	0.91	1.00	0.95	10
watermelon	1.00	1.00	1.00	10
accuracy			0.95	351
macro avg	0.95	0.95	0.94	351
weighted avg	0.95	0.95	0.94	351

Figure 4.2.3 Performance Metrics for MobileNetV2

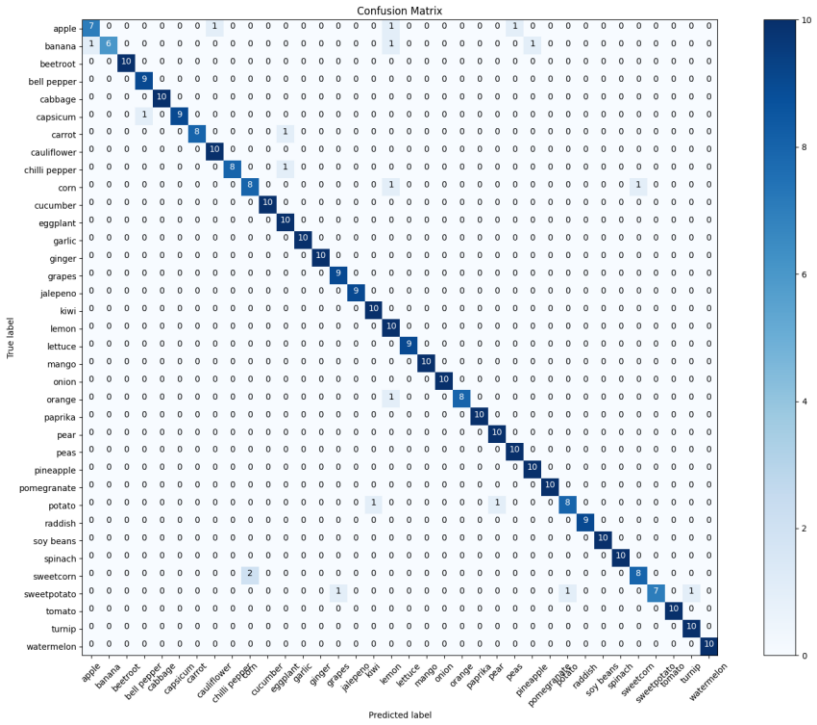

Figure 4.2.4 Confusion Matrix for MobileNetV2

B) Model Comparison: We conducted a comprehensive comparison of our two image recognition models, focusing on accuracy, precision, recall, F1-scores and the confusion matrices for each ingredient category.

1) Custom CNN Model: Our custom Convolutional Neural Network (CNN) model show-cases a commendable accuracy of 93%, reflecting its strong ability to identify a diverse array of ingredients. The model consistently delivers high precision, recall, and F1-scores, underscoring its proficiency. Notably, it excels in recognizing beetroot, cabbage, carrot, cucumber, and other ingredients, achieving impeccable F1-scores. However, it does face some challenges in the accurate identification of certain ingredients like apple and bell pepper, which is important to consider when assessing its performance.

2) MobileNetV2 Model: In contrast, the MobileNetV2 model emerges as the standout performer, surpassing the custom CNN model with an impressive accuracy of 95%. This model not only excels in recognizing ingredients but maintains consistently high recall and F1-score values. It demonstrates exceptional proficiency in recognizing a diverse range of ingredients, including

beetroot, garlic, lemon, and more, with flawless F1-scores. The model's ability to generalize across different ingredients and maintain high accuracy is a testament to its robustness.

Comparative Insights: When comparing the two models, the MobileNetV2 model clearly outperforms the custom CNN model. It showcases a notable edge in terms of accuracy and generalization across various ingredients. The MobileNetV2 model consistently delivers superior precision, recall, and F1-scores. It presents a compelling case for its application in ingredient recognition tasks.

To gain a deeper understanding of the recognition capabilities of both models, we also included confusion matrices that provide valuable insights into the specific strengths and challenges encountered by each model when classifying ingredients.

This comprehensive evaluation led us to the decision to proceed with the MobileNetV2 model for its remarkable performance and robustness.

4.3 Recipe Recommendation Models:

A) Results: In this section, we provide a comprehensive overview of the results achieved by various recipe recommendation models, with a primary focus on the precision at k (k=10) as the evaluation metric.

1) BERT Model using BERT Embeddings: This model, based on BERT embeddings, exhibited a modest performance with a precision at k of 0.1. The results were influenced by the model's ability to capture ingredient relationships and semantic meaning.

```
K = 10# mention it as actual no. of recipies
# Calculate P@K
correct_recommendations=[]
for recipe in actual_recipes:
    for i in rec['TranslatedRecipeName'][:K]:
        if recipe == i:
            correct_recommendations.append(True)
#correct_recommendations = [recipe in rec['TranslatedRecipeName'][:K] for recipe in actual_r
precision_at_K = sum(correct_recommendations) / K

print(f"Precision at {K} of Bert model using BERT embeddings : {precision_at_K}")
```

✓ 0.0s Python

```
Precision at 10 of Bert model using BERT embeddings : 0.1
```

Figure 4.3.1 Precision at k for BERT Model using BERT Embeddings

2) LSTM: The LSTM model, with a precision at k of 0.0, faced challenges in generating accurate recipe recommendations. Its ability to effectively capture ingredient relationships and context appeared to be less competitive when compared to other models.

```
K = 10 # mention it as actual no. of recipies
# Calculate P@K
correct_recommendations=[]
for recipe in actual_recipes:
    for i in rec['TranslatedRecipeName'][:K]:
        if recipe == i:
            correct_recommendations.append(True)
#correct_recommendations = [recipe in rec['TranslatedRecipeName'][:K] for recipe in actual_r
precision_at_K = sum(correct_recommendations) / K

print(f"Precision at {K} using LSTM: {precision_at_K}")
```

✓ 0.0s Python

```
Precision at 10 using LSTM: 0.0
```

Figure 4.3.2 Precision at k for BERT Model using LSTM

3) GENSIM (FastText): Similar to the BERT model using BERT embeddings, the GENSIM model achieved a precision at k of 0.1. While it displayed reasonable performance, there remains room for improvement in recommendation accuracy.

```
# Calculate P@K
    correct_recommendations = [recipe in recommendations[:K] for recipe in actual_recipes]
    precision_at_K = sum(correct_recommendations) / K
    print(f"Precision at {K} using GENSIM: {precision_at_K}")
```

✓ 0.0s Python

```
x = get_recommendations(input)
```

✓ 2.1s Python

```
Precision at 10 using GENSIM: 0.1
```

Figure 4.3.3 Precision at k for BERT Model using GENSIM

4) Word2Vec and TF-IDF: Outperformed other models with a precision at k of 0.2. This approach, which combines Word2Vec embeddings and TF-IDF weighting, excelled in comprehending ingredient relationships and delivering precise recipe suggestions.

```
K = 10 # mention it as actual no. of recipies
# Calculate P@K
correct_recommendations=[]
for recipe in actual_recipes:
    for i in result['TranslatedRecipeName'][:K]:
        if recipe == i:
            correct_recommendations.append(True)
#correct_recommendations = [recipe in rec['TranslatedRecipeName'][:K] for recipe in actual_recipes]
precision_at_K = sum(correct_recommendations) / K
print(f"Precision at {K} using Word2Vec using TF-IDF vectorization: {precision_at_K}")
```

✓ 0.0s Python

```
Precision at 10 using Word2Vec using TF-IDF vectorization: 0.2
```

Figure 4.3.4 Precision at k for BERT Model using Word2Vec and TF-IDF

5) BERT Model using Sentence Transformer: This model secured the highest precision at k, achieving a score of 0.5. Leveraging sentence embeddings derived from BERT, it excelled in capturing the semantic meaning of ingredients and consistently provided recommendations that closely matched user preferences.

```
K = 10 # mention it as actual no. of recipies
# Calculate P@K
correct_recommendations=[]
for recipe in actual_recipes:
    for i in result['TranslatedRecipeName'][:K]:
        if recipe == i:
            correct_recommendations.append(True)
#correct_recommendations = [recipe in rec['TranslatedRecipeName'][:K] for recipe in actual_recipes]
precision_at_K = sum(correct_recommendations) / K
print(f"Precision at {K} using BERT final model using sentence transformer model : bert-base-nli-mean-tokens: {precision_at_K}")
```

✓ 0.0s Python

```
Precision at 10 using BERT final model using sentence transformer model : bert-base-nli-mean-tokens: 0.5
```

Figure 4.3.5 Precision at k for BERT Model using Sentence Transformer

6) Logistic Regression: The Logistic Regression model yielded a precision at k of 0.0, indicating that it did not deliver accurate recipe recommendations. It struggled to encompass the complexity of ingredient relationships compared to more advanced models.

```
K = 10 # mention it as actual no. of recipies
# Calculate P@K
correct_recommendations=[]
for recipe in actual_recipes:
    for i in predicted_recipes[0]:
        if recipe == i:
            correct_recommendations.append(True)
#correct_recommendations = [recipe in rec['TranslatedRecipeName'][:K] for recipe in actual_recipes]
precision_at_K = sum(correct_recommendations) / K

print(f"Precision at {K} using Logistic Regression: {precision_at_K}")
```

✓ 0.0s Python

```
Precision at 10 using Logistic Regression: 0.0
```

Figure 4.3.6 Precision at k for Logistic Regression

Dataset Size and Its Impact: Our dataset comprises approximately 3,000 unique ingredients and 6,000 unique recipes. It's important to highlight that the ratio of unique ingredients to unique recipes in the dataset is almost 1:2, indicating that the amount of data may be insufficient for the recommendation models, especially those reliant on word embeddings, to achieve higher precision.

With a relatively small dataset, the models face limitations in effectively learning the semantic relationships and contextual understanding of ingredients. In particular, word embedding models require a substantial amount of data to capture the nuances of ingredient semantics, associations, and variations effectively. The relatively low number of recipes compared to the diversity of ingredients can result in a limited contextual understanding of ingredient relationships, making it challenging for the models to provide highly precise recommendations.

It's important to acknowledge that with a larger dataset, the models could potentially yield improved performance by having access to more varied and context-rich examples. In the context of our dataset, while some models demonstrated competitive performance, the dataset size remains a constraint that could be mitigated with an expansion in the number of recipes, enabling the models to better learn and comprehend ingredient relationships.

B) Model Comparison:

In this section, we offer an in-depth comparison of the various recipe recommendation models under consideration, using "precision at k" (k=10) as our primary evaluation metric. This metric enables us to assess how effectively these models provide recipe suggestions that closely align with users' preferences.

Model for Recipe Recommendation	Precision at k (k=10)
BERT Model using Sentence Transformer	0.5
Word2Vec and TF-IDF	0.2
GENSIM (FastText)	0.1
BERT Model using BERT Embeddings	0.1
LSTM	0.0
Logistic Regression	0.0

Table 4.3.1 Precision at k for all the Recipe Recommendation Models

BERT Model using Sentence Transformer: Emerging as the top-performing recommendation system, this model achieves an impressive precision at k of 0.5. Its remarkable ability to grasp the intricacies of ingredient semantics makes it the standout choice for ensuring highly accurate and user-centric recipe recommendations.

Word2Vec and TF-IDF: This model combination demonstrates commendable performance with a precision at k of 0.2, outshining several others. It exhibits a strong grasp of ingredient relationships, offering users precise and context-aware recipe recommendations.

GENSIM (FastText) and BERT Model using BERT Embeddings: Both models deliver a precision at k of 0.1, showcasing reasonable performance. However, there's room for improvement in terms of recommendation accuracy, suggesting the need for further model fine-tuning to capture intricate ingredient relationships effectively.

LSTM and Logistic Regression: Both models yield a precision at k of 0.0, indicating limited effectiveness in generating accurate recipe recommendations. These models face challenges in comprehending the

complexities of ingredient relationships and might require additional enhancements to be considered viable options.

The model comparison underscores the supremacy of the BERT Model using Sentence Transformer, boasting a precision at k of 0.5. Its exceptional proficiency in understanding ingredient semantics and delivering highly precise recipe recommendations solidifies its role as the optimal choice for our recipe recommendation system. This decision informs our selection of this model as the final recommendation engine for our project.

Chapter 5

Advantages, Applications and Limitations

5.1 Advantages:

- **Culinary Convenience:** The AI-powered culinary curator offers a significant advantage by simplifying the meal planning process. It provides users with tailored recipe recommendations based on the ingredients they have available. This not only saves time but also reduces food waste, as users can make the most of what's in their pantry or fridge. This feature aligns with the modern lifestyle characterized by busy schedules and a desire for convenient yet homemade meals.
- **Diverse Cuisine Options:** One distinctive feature of our project is its specialization in Indian recipes. While many recipe recommendation apps focus on international or generic cuisines, our system caters to users interested in the rich and diverse world of Indian cuisine. This focus provides a unique culinary experience for those looking to explore or enjoy Indian dishes, whether they are Indian residents or global food enthusiasts.
- **Integration of NLP and Computer Vision:** Our project stands out for its combination of Natural Language Processing (NLP) and Computer Vision technologies. It uses BERT, a state-of-the-art NLP model, for recognizing ingredients from text inputs. Simultaneously, it leverages MobileNetV2, a computer vision model, to identify ingredients from images. This dual approach offers versatility to users, allowing them to input ingredients in a way that suits their preference or accessibility, whether through text or images.
- **Nutritional Analysis:** In the era of growing health awareness, the nutritional analysis feature is a notable advantage. It empowers users to make informed decisions about their meals by providing insights into the nutritional value of recipes. Users can understand the calorie content, macronutrients, vitamins, and more, facilitating health-conscious choices. This feature contributes to healthier eating habits, a valuable benefit for those with dietary concerns.
- **Research Contribution:** Beyond its practical applications, our project contributes to the research field of AI-driven recipe systems. It conducts a comparative analysis of various models for ingredient recognition and recommendation. This academic exploration helps advance the understanding of AI approaches to culinary challenges. By assessing and contrasting different

models, our project adds to the knowledge base, potentially guiding future developments in the field.

5.2 Applications

- **Home Cooking:** The project is designed primarily for individuals who prefer cooking at home. It simplifies the process of deciding what to cook, which can be a daily challenge. By suggesting recipes based on available ingredients, the system encourages more people to engage in home cooking, fostering culinary skills and self-reliance in the kitchen.
- **Culinary Exploration:** Food enthusiasts find an exciting application in our project. They can use it to explore and experiment with Indian recipes, even if they are not familiar with Indian cuisine. It's an opportunity to broaden their culinary horizons, try new flavors, and create diverse and delicious meals.
- **Diet Planning:** A significant application is diet planning. Users with specific dietary preferences, such as vegetarian, vegan, or gluten-free, can easily find recipes that align with their dietary restrictions or choices. The system helps them maintain their dietary lifestyle and discover new dishes within their preferred dietary category.
- **Nutrition Monitoring:** Health-conscious users benefit from the system's ability to analyze the nutritional content of recipes. It is a valuable tool for those looking to monitor their calorie intake, macronutrients, and other nutritional aspects. This application promotes informed and healthy eating, contributing to improved well-being.

5.3 Limitations

- **Cultural Focus:** While our project excels in recommending Indian recipes, its cultural focus can be limiting for users seeking recipes from other cuisines. It may not provide as broad a range of recommendations for non-Indian cuisines, potentially narrowing its appeal to a specific audience.
- **Ingredient Recognition Challenges:** Recognizing ingredients from images can be challenging under suboptimal conditions. Factors such as lighting, image quality, and food presentation can impact the accuracy of ingredient recognition. This limitation may occasionally result in misidentified or incomplete ingredient lists.
- **Data-Dependent:** The effectiveness of our project is inherently linked to the availability and quality of recipe and nutritional data. The quality of recommendations and nutritional analyses is contingent on the completeness and accuracy of the underlying data sources.
- **User Engagement:** The accuracy and usefulness of the system depend on user engagement. Encouraging users to actively input data, including text or images of ingredients, is a potential challenge. Without sufficient user participation, the system's performance may be hindered

Chapter 6

Conclusion and Future Scope

In a fast-paced world marked by urbanization and technology reliance, the act of home cooking holds a cherished and vital place in our daily routines. It transcends cultural boundaries, strengthens family bonds, and offers a diverse culinary experience. However, in the midst of our modern, hectic lives, deciding what to prepare from the ingredients at hand can be a daunting task.

This culinary challenge has been the driving force behind our project. Our research addresses the contemporary challenges related to food and health. We introduce the Culinary Curator, a technological marvel at the confluence of artificial intelligence, natural language processing, and computer vision. This cooking assistant acts as a guiding light for individuals seeking culinary inspiration, especially when faced with the dilemma of what to cook with available ingredients. It also provides nutritional analysis, promoting healthier and more mindful eating choices.

The heart of this project lies in the extensive research and rigorous evaluation. The models we've selected, including 'BERT' for recipe recommendations, 'MobileNetV2' for ingredient detection from photos, and GPT-2 for recipe generation, are the culmination of meticulous research, ensuring the highest quality.

Our project offers a transformative solution that enhances culinary experiences, simplifies home cooking, and aligns perfectly with our contemporary, tech-driven lifestyle. It stands as a testament to the enduring significance of home-cooked meals and their ability to bring people together. The innovative approach of our project has the potential to revolutionize the way we perceive cooking, ensuring this beloved tradition thrives in the face of modern challenges.

Frontend Improvisations: Our project can benefit from continuous refinement of the user interface and overall user experience. Intuitive design and seamless navigation are essential to make the Culinary Curator even more user-friendly and accessible.

Enhancing Ingredient Recognition: To further boost the accuracy and capabilities of ingredient recognition, we can explore advanced computer vision techniques and models. This may involve fine-tuning existing models or integrating new ones to handle a wider range of ingredients and complex scenarios, such as regional variations.

Recipe Generation Models: In addition to GPT-2, we can consider implementing a variety of language models for recipe generation. This diversity can cater to different user preferences, allowing for the creation of recipes in various styles, cuisines, and dietary requirements.

Expanded Scope: To increase the project's scope, we can venture into additional functionalities, such as:

Allergen and Dietary Restrictions: Incorporating features that allow users to specify dietary restrictions or allergies to receive recipe recommendations tailored to their needs.

Multi-Cuisine Support: Extending the system's knowledge and recommendation capabilities to include a wider array of cuisines from around the world.

Ingredient Substitution: Providing suggestions for ingredient substitutions, ensuring that users can adapt recipes to what they have on hand.

Meal Planning: Enabling users to plan meals for a day, week, or special occasions and generating shopping lists based on selected recipes.

Community and User-Generated Content: Allowing users to share their recipes, modifications, and reviews, fostering a sense of community and collaboration.

References

[1] M. B. Vivek, M. N. Manju, and M. B. V. Vijay, "Machine Learning Based Food Recipe Recommendation System," 2023.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep Learning based Recommender System: A Survey and New Perspectives," IEEE Communications Magazine, vol. 57, no. 9, pp. 120-128, 2019.

[3] A. Dharawat and C. Doan, "Ingredient Extraction from Text in the Recipe Domain," 2022.

[4] M. S. Rodrigues, F. Fidalgo, and A. Oliveira, "RecipeIS—Recipe Recommendation System Based on Recognition of Food Ingredients," 2023.

[5] R. Patil, N. Shirsat, K. Bumb, and A. Satpute, "Identify Ingredients From Food Images and Generate Recipe," 2023.

[6] H. H. Lee, K. Shu, P. Achananuparp, P. K. Prasetyo, Y. Liu, E.-P. Lim, and L. R. Varshney, "RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System," arXiv preprint arXiv:2007.07788, 2020.

[7] T. Theodoridis, V. Solachidis, K. Dimitropoulos, L. Gymnopoulos, and P. Daras, "A Survey on AI Nutrition Recommender Systems," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 6, no. 1, pp. 81-96, 2022.

[8] H. Onyeaka, P. Tamasiga, N. Uju, T. Miri, U. C. Juliet, O. Nwaiwu, and A. A. Akinsemolu, "Using Artificial Intelligence to Tackle Food Waste and Enhance the Circular Economy: Maximising Resource Efficiency and Minimising Environmental Impact: A Review," arXiv preprint arXiv:2302.10659, 2023.

[9] Y. Tian, C. Zhang, R. Metoyer, and N. V. Chawla, "Recipe Recommendation With Hierarchical Graph Attention Network," arXiv preprint arXiv:2203.10795, 2022.

[10] K. Taneja, R. Segal, and R. Goodwin, "Monte Carlo Tree Search for Recipe Generation using GPT-2," arXiv preprint arXiv:2011.09522, 2020.

[11] J. Chen and C. Ngo, "Deep-based Ingredient Recognition for Cooking Recipe Retrieval," Proceedings of the 2016 ACM on Multimedia Conference, pp. 1373-1382, 2016.

[12] A. Mane, M. Bawankar, M. Goradia, and S. Chaudhari, "Image to Recipe and Nutritional Value Generator Using Deep Learning," arXiv preprint arXiv:2203.13713, 2022.

[13] M. Bolaños, A. Ferrà, and P. Radeva, "Food Ingredients Recognition Through Multi-label Learning," Lecture Notes in Computer Science, pp. 500-511, 2017.

[14] R. Maia and J. C. Ferreira, "Context-Aware Food Recommendation System," Proceedings of the 2018 ACM International Conference on Information and Knowledge Management, pp. 2519-2522, 2018.

[15] B. P. Majumder, S. Li, J. Ni, and J. McAuley, "Generating Personalized Recipes from Historical User Preferences," Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2109-2118, 2019.

[16] J. Webster and C. Kit, "Tokenization as the initial phase in NLP," 2023.

[17] D. Khyani and S. B. S, "An Interpretation of Lemmatization and Stemming in Natural Language Processing," 2023

[18] Kritik Seth, "Fruits and Vegetables Image Recognition Dataset," Kaggle 2020 [https://www.kaggle.com/kritikseth/fruit-and-vegetable-image-recognition]

[19] Kanishk Jain, “6000+ Indian Food Recipes Dataset”, Kaggle 2020 [https://www.kaggle.com/datasets/kanishk307/6000-indian-food-recipes-dataset]

Link to codes