IDS SEMESTER PROJECT

RECIPE RECOMMENDATION SYSTEM

NAME	ENROLLMENT
MARYAM ASHFAQ	01-135201-037
TAYYAB NAWAZ SATTI	01-135202-099
FATIMA SAFA	01-135202-027

1. Introduction

The digital age has significantly influenced how we access and prepare food. With a vast number of recipes available online, choosing the right one based on personal preferences and available ingredients can be overwhelming. To address this, we have developed a Recipe Recommendation System that helps users find recipes tailored to their needs. This system utilizes a dataset of recipes containing various attributes like ingredients, preparation and cooking time, cuisine type, and more, to provide personalized recommendations.

The dataset used in this project includes over 6000 recipes with the following attributes:

- TranslatedRecipeName: The name of the recipe in English or hindi.
- TranslatedIngredients: List of ingredients required for the recipe, translated into English.
- TranslatedInstructions: Step-by-step instructions for preparing the recipe, translated into English.
- PrepTimeInMins: Preparation time in minutes.
- CookTimeInMins: Cooking time in minutes.
- TotalTimeInMins: Total time required (prep time + cook time) in minutes.
- Servings: Number of servings the recipe provides.
- Cuisine: Cuisine type (e.g., Indian, South Indian).
- Course: Course of the meal (e.g., Main Course, Side Dish).
- Diet: Type of diet (e.g., Vegetarian, Non-vegetarian).
- URL: URL of the recipe for further details.

6000+ Indian Food Recipes Dataset (kaggle.com)

2. Problem statement

Finding the right recipe that matches a user's available ingredients, time constraints, dietary preferences, and cuisine choices can be challenging. The aim is to develop a recommendation system that simplifies this process by filtering and ranking recipes based on these criteria.

3. Scope

The scope of this project includes:

Data cleaning and preprocessing to handle missing or inconsistent data.
Feature extraction and engineering to represent recipes in a machine-readable format.
Implementing a content-based filtering algorithm to recommend recipes based on user preferences.
Developing a graphical user interface (GUI) to make the system user-friendly and accessible.

4. Objectives

The primary objective of this project is to build an efficient and user-friendly recipe recommendation system that:

- Filters recipes based on user-specified ingredients and preferences.
- Ranks and recommends recipes using content-based filtering.
- Provides detailed information about the recommended recipes, including instructions and URLs.

5. Methodology

The methodology of the Food Recommendation System project involves several steps, including data collection, preprocessing, feature extraction, algorithm implementation, GUI development, and evaluation. Let's break down each step in detail:

1. Data Collection:

The first step was selecting dataset containing information about recipes. This dataset should included attributes such as recipe names, ingredients, instructions, preparation and cooking times, cuisine types, dietary preferences, and URLs for additional details. In this project, the dataset used is 'IndianFoodDataset.xlsx', which contains over 6000 recipes with various attributes.

2. Data Preprocessing:

Data preprocessing involved cleaning and transforming the raw dataset into a format suitable for analysis and modeling. Tasks in this step include handling missing values, removing duplicates, standardizing text data (e.g., converting to lowercase, removing special characters), and handling categorical variables. In this project, the 'TranslatedIngredients' column is preprocessed to extract only ingredient names, and missing values are handled.

3. Feature Extraction and Engineering:

Feature extraction involves converting raw data into numerical or categorical features that is used for modeling. Feature engineering involve creating new features from existing ones or transforming features to improve model performance. In this project, text data from columns like 'TranslatedIngredients' and 'TranslatedInstructions' are combined into a single feature vector using techniques like **TF-IDF (Term Frequency-Inverse Document Frequency)** to represent the recipes.

4. Algorithm Implementation:

Once the dataset is prepared, an algorithm is implemented to generate recipe recommendations based on user preferences. **Content-based filtering** is a common approach used in recommendation systems, where recommendations are made based on the similarity between items (recipes) and user preferences. In this

project, **cosine similarity** is calculated between the feature vectors of recipes to determine their similarity, and the top similar recipes are recommended to the user.

5. GUI Development:

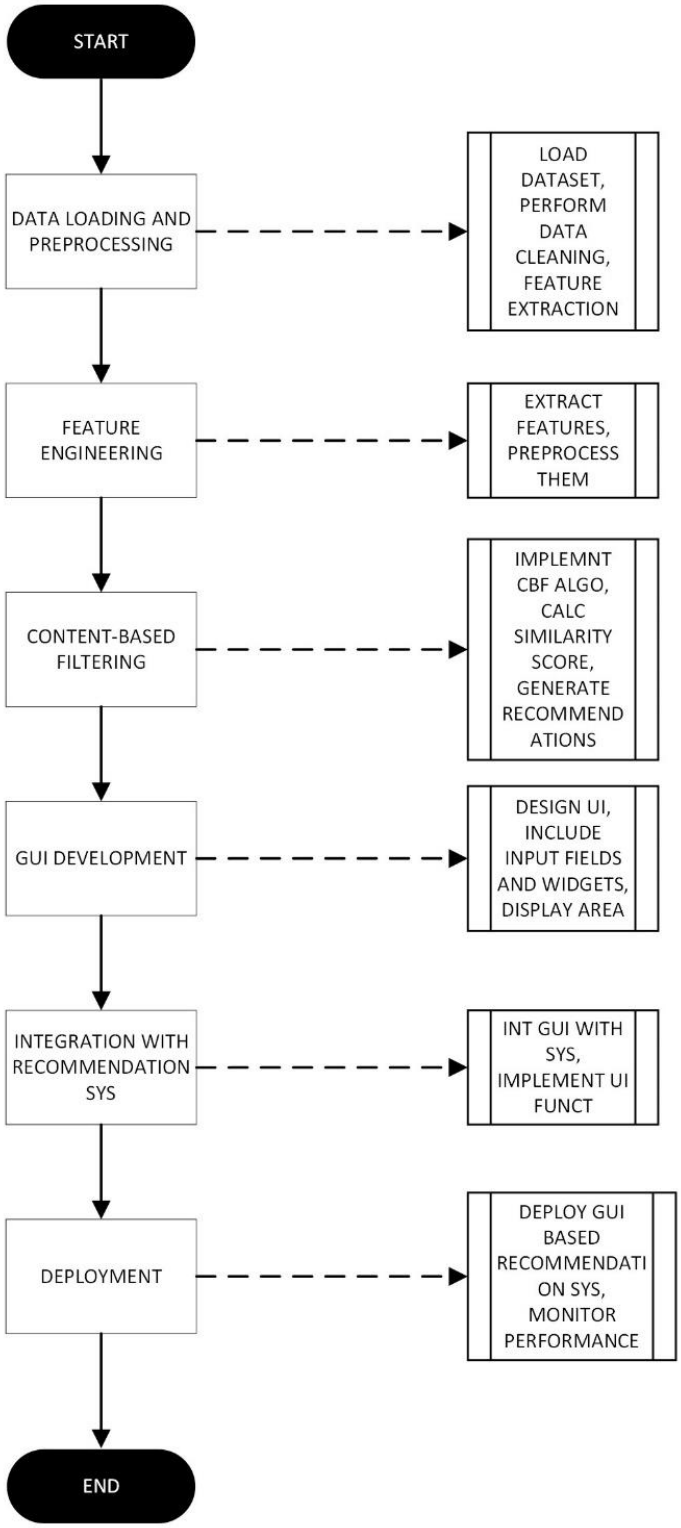
To make the system user-friendly, a graphical user interface (GUI) is developed to allow users to interact with the recommendation system easily. The GUI includes input fields for users to enter their available ingredients and buttons to submit inputs and view recommendations. In this project, the GUI is developed using the **tkinter library** in Python, providing windows for inputting ingredients, displaying recommendations, and copying recipe URLs.

6. Evaluation:

Finally, the system is evaluated to assess its performance and usability. Evaluation metrics may include accuracy, precision, recall, and user satisfaction. In this project, the system's effectiveness in providing relevant and personalized recipe recommendations is evaluated based on user feedback and the quality of recommendations.

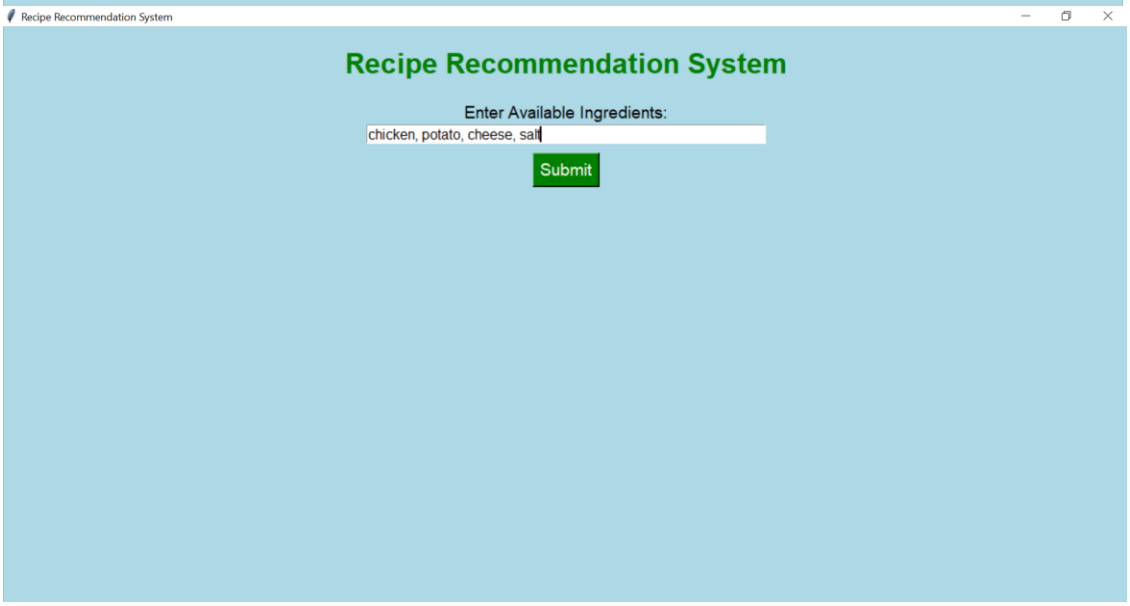
By following this methodology, the Food Recommendation System is developed effectively, providing users with tailored recipe recommendations based on their preferences and available ingredients.

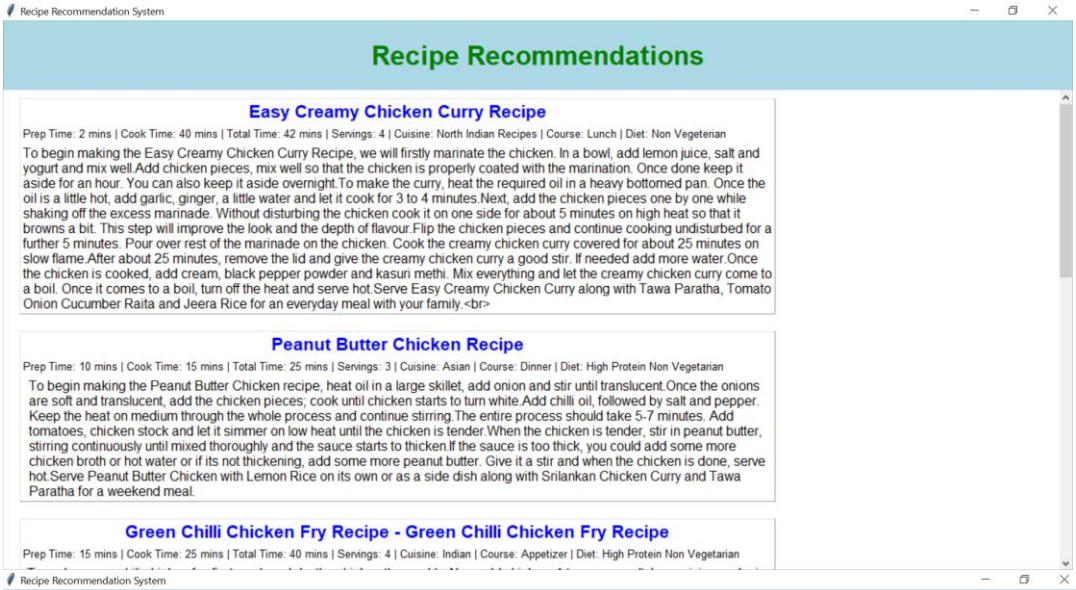
6. Flow chart


START
DATA LOADING AND PREPROCESSING
LOAD DATASET, PERFORM DATA CLEANING, FEATURE EXTRACTION
FEATURE ENGINEERING
EXTRACT FEATURES, PREPROCESS THEM
CONTENT-BASED FILTERING
IMPLEMNT CBF ALGO, CALC SIMILARITY SCORE, GENERATE RECOMMENDATIONS
GUI DEVELOPMENT
DESIGN UI, INCLUDE INPUT FIELDS AND WIDGETS, DISPLAY AREA
INTEGRATION WITH RECOMMENDATION SYS
INT GUI WITH SYS, IMPLEMENT UI FUNCT
DEPLOYMENT
DEPLOY GUI BASED RECOMMENDATION SYS, MONITOR PERFORMANCE
END

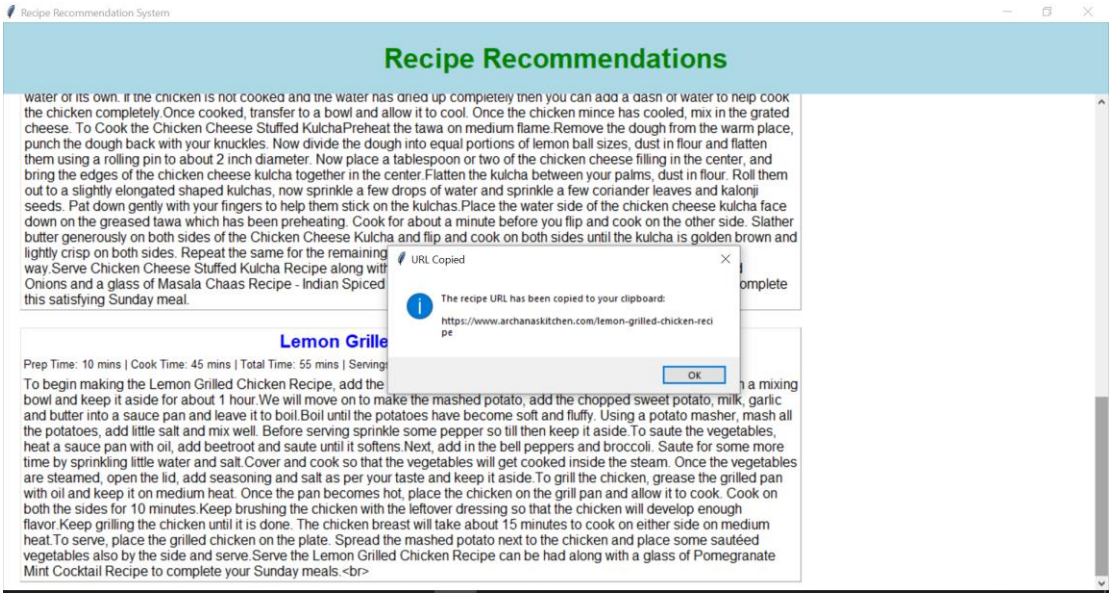
7. Code


Recipe Recommendation System
Recipe Recommendation System
Let's Get Cooking


Recipe Recommendation System
Recipe Recommendation System
Enter Available Ingredients:
chicken, potato, cheese, salt
Submit


Recipe Recommendation System
Recipe Recommendations
Easy Creamy Chicken Curry Recipe
Prep Time: 2 mins | Cook Time: 40 mins | Total Time: 42 mins | Servings: 4 | Cuisine: North Indian Recipes | Course: Lunch | Diet: Non Vegeterian
To begin making the Easy Creamy Chicken Curry Recipe, we will firstly marinate the chicken. In a bowl, add lemon juice, salt and yogurt and mix well.Add chicken pieces, mix well so that the chicken is properly coated with the marination. Once done keep it aside for an hour. You can also keep it aside overnight.To make the curry, heat the required oil in a heavy bottomed pan. Once the oil is a little hot, add garlic, ginger, a little water and let it cook for 3 to 4 minutes.Next, add the chicken pieces one by one while shaking off the excess marinade. Without disturbing the chicken cook it on one side for about 5 minutes on high heat so that it browns a bit. This step will improve the look and the depth of flavour.Flip the chicken pieces and continue cooking undisturbed for a further 5 minutes. Pour over rest of the marinade on the chicken. Cook the creamy chicken curry covered for about 25 minutes on slow flame.After about 25 minutes, remove the lid and give the creamy chicken curry a good stir. If needed add more water.Once the chicken is cooked, add cream, black pepper powder and kasuri methi. Mix everything and let the creamy chicken curry come to a boil. Once it comes to a boil, turn off the heat and serve hot.Serve Easy Creamy Chicken Curry along with Tawa Paratha, Tomato Onion Cucumber Raita and Jeera Rice for an everyday meal with your family.
Peanut Butter Chicken Recipe
Prep Time: 10 mins | Cook Time: 15 mins | Total Time: 25 mins | Servings: 3 | Cuisine: Asian | Course: Dinner | Diet: High Protein Non Vegetarian
To begin making the Peanut Butter Chicken recipe, heat oil in a large skillet, add onion and stir until translucent.Once the onions are soft and translucent, add the chicken pieces; cook until chicken starts to turn white.Add chilli oil, followed by salt and pepper. Keep the heat on medium through the whole process and continue stirring.The entire process should take 5-7 minutes. Add tomatoes, chicken stock and let it simmer on low heat until the chicken is tender.When the chicken is tender, stir in peanut butter, stirring continuously until mixed thoroughly and the sauce starts to thicken.If the sauce is too thick, you could add some more chicken broth or hot water or if its not thickening, add some more peanut butter. Give it a stir and when the chicken is done, serve hot.Serve Peanut Butter Chicken with Lemon Rice on its own or as a side dish along with Srilankan Chicken Curry and Tawa Paratha for a weekend meal.
Green Chilli Chicken Fry Recipe - Green Chilli Chicken Fry Recipe
Prep Time: 15 mins | Cook Time: 25 mins | Total Time: 40 mins | Servings: 4 | Cuisine: Indian | Course: Appetizer | Diet: High Protein Non Vegetarian


Recipe Recommendation System
Recipe Recommendations
Green Chilli Chicken Fry Recipe - Green Chilli Chicken Fry Recipe
Prep Time: 15 mins | Cook Time: 25 mins | Total Time: 40 mins | Servings: 4 | Cuisine: Indian | Course: Appetizer | Diet: High Protein Non Vegetarian
To make green chili chicken fry, first wash and dry the chicken thoroughly. Now add chicken, 1 teaspoon salt, lemon juice and mix in a bowl. Cover and keep it in the fridge for 30 minutes. Now in a mixer grinder, add green chili, ginger, garlic, coriander, onion and grind them. Keep it separately. Now in this mixer grinder add some water with tomatoes and grind it. Keep it separately. Now heat oil in a pan. After the oil is heated, add long, cinnamon, green chilli paste and cook it for 2 minutes. After 2 minutes, add the chicken and cook until the chicken changes its color. Now add tomato puree to it and cook it for 3 to 4 minutes. Add coriander powder, salt and cook for 5 minutes. After 5 minutes, add 1/2 cup of water and cook the chicken till the chicken is cooked well. Serve. Serve Green Chilli Chicken Fry Recipe along with Dal Fry and Rice for dinner.
Chicken Cheese Stuffed Kulcha Recipe
Prep Time: 30 mins | Cook Time: 20 mins | Total Time: 50 mins | Servings: 4 | Cuisine: North Indian Recipes | Course: Side Dish | Diet: Non Vegeterian
To begin making the Chicken Cheese Stuffed Kulcha Recipe, we will first make the dough. To make the dough for the Chicken Cheese Stuffed Kulchaln a mixing bowl, combine the whole wheat flour, all purpose flour, yeast, curd, sugar, salt, butter with your fingers. Next add water, little by little to knead the dough. We are looking for a smooth soft dough. Bring the kulcha dough together to form a smooth dough with a few drops of cooking oil. Cover the bowl with a damp cloth and rest it in a warm place for about 30-40 minutes or until the dough has doubled in size. To make the filling for the Chicken Cheese Stuffed Kulchaln a pan heat oil on medium heat, once the oil is hot add the garlic and saute for 30 seconds. Next add all the dry masalas, red chilli powder, cumin powder, black pepper powder, turmeric powder, garam masala powder, coriander powder and salt. At this stage add the minced chicken and cook until the chicken has fully cooked through. This will take a good 12-15 minutes. The chicken will release some water of its own. If the chicken is not cooked and the water has dried up completely then you can add a dash of water to help cook the chicken completely.Once cooked, transfer to a bowl and allow it to cool. Once the chicken mince has cooled, mix in the grated cheese. To Cook the Chicken Cheese Stuffed KulchaPreheat the tawa on medium flame.Remove the dough from the warm place, punch the dough back with your knuckles. Now divide the dough into equal portions of lemon ball sizes, dust in flour and flatten them using a rolling pin to about 2 inch diameter. Now place a tablespoon or two of the chicken cheese filling in the center, and bring the edges of the chicken cheese kulcha together in the center.Flatten the kulcha between your palms, dust in flour. Roll them out to a slightly elongated shaped kulchas, now sprinkle a few drops of water and sprinkle a few coriander leaves and kalonji seeds. Pat down gently with your fingers to help them stick on the kulchas.Place the water side of the chicken cheese kulcha face down on the greased tawa which has been preheating. Cook for about a minute before you flip and cook on the other side. Slather



```
import tkinter as tk
from tkinter import messagebox, scrolledtext
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity


file_path = 'IndianFoodDataset.xlsx'
df = pd.read_excel(file_path)


def preprocess_ingredients(ingredient_string):
    if isinstance(ingredient_string, str):

        ingredient_names = [re.sub(r"(\d+(\.\d+)?)\s*(\w+)?", "", ingredient).strip() for ingredient in
ingredient_string.split(',')]
        return ' '.join(ingredient_names)
    else:
        return ''

df['TranslatedIngredients'] = df['TranslatedIngredients'].apply(preprocess_ingredients)

df['combined_text'] = (df['TranslatedIngredients'] + ' ' +
                       df['TranslatedInstructions'] + ' ' +
                       df['Cuisine'] + ' ' +
                       df['Course'] + ' ' +
                       df['Diet'])
```

```
df['combined_text'] = df['combined_text'].fillna('')

# Vectorize the text data
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(df['combined_text'])
cosine_sim_matrix = cosine_similarity(tfidf_matrix, tfidf_matrix)


def get_recommendations(ingredients):
    ingredients = preprocess_ingredients(ingredients)
    combined_text = ingredients + ' ' + ' '.join([''] * 5)
    input_vector = tfidf_vectorizer.transform([combined_text])
    sim_scores = cosine_similarity(input_vector, tfidf_matrix)
    top_indices = sim_scores.argsort(axis=1)[0][-6:-1]
    recommendations = df.iloc[top_indices][['TranslatedRecipeName', 'PrepTimeInMins',
'CookTimeInMins', 'TotalTimeInMins', 'Servings', 'Cuisine', 'Course', 'Diet',
'TranslatedInstructions', 'URL']]
    return recommendations


class TitleWindow:
    def __init__(self, root):
        self.root = root
        self.root.title("Recipe Recommendation System")
        self.root.geometry("400x200")
        self.root.configure(bg='lightblue')

        title_label = tk.Label(root, text="Recipe Recommendation System", font=("Helvetica", 24,
"bold"), bg='lightblue', fg='green')
        title_label.pack(pady=50, padx=20)

        button = tk.Button(root, text="Let's Get Cooking",
command=self.open_ingredient_window, font=("Helvetica", 14), bg='green', fg='white')
        button.pack(pady=10)

    def open_ingredient_window(self):
        self.root.destroy()
        ingredient_window = tk.Tk()
        app = IngredientWindow(ingredient_window)
        ingredient_window.mainloop()


class IngredientWindow:
    def __init__(self, root):
        self.root = root
```

```
        self.root.title("Recipe Recommendation System")
        self.root.geometry("400x200")
        self.root.configure(bg='lightblue')

        title_label = tk.Label(root, text="Recipe Recommendation System", font=("Helvetica", 24,
"bold"), bg='lightblue', fg='green')
        title_label.pack(pady=20, padx=20)

        ingredients_label = tk.Label(root, text="Enter Available Ingredients:", font=("Helvetica",
14), bg='lightblue')
        ingredients_label.pack()

        self.ingredients_entry = tk.Entry(root, width=50, font=("Helvetica", 12))
        self.ingredients_entry.pack()

        submit_button = tk.Button(root, text="Submit", command=self.get_recommendations,
font=("Helvetica", 14), bg='green', fg='white')
        submit_button.pack(pady=10)

    def get_recommendations(self):
        ingredients = self.ingredients_entry.get()
        if ingredients:
            recommendations = get_recommendations(ingredients)
            if not recommendations.empty:
                self.root.destroy()
                recommendation_window = tk.Tk()
                app = RecommendationWindow(recommendation_window, recommendations)
                recommendation_window.mainloop()
            else:
                messagebox.showinfo("No Recommendations", "No recommendations found for the
provided ingredients.")
        else:
            messagebox.showwarning("Empty Input", "Please enter at least one ingredient.")


class RecommendationWindow:
    def __init__(self, root, recommendations):
        self.root = root
        self.root.title("Recipe Recommendation System")
        self.root.geometry("800x600")
        self.root.configure(bg='lightblue')

        title_label = tk.Label(root, text="Recipe Recommendations", font=("Helvetica", 24, "bold"),
bg='lightblue', fg='green')
        title_label.pack(pady=20, padx=20)
```

```
        canvas = tk.Canvas(root, bg='#ffffff', bd=0, highlightthickness=0)
        scrollbar = tk.Scrollbar(root, orient="vertical", command=canvas.yview)
        scrollable_frame = tk.Frame(canvas, bg='#ffffff')
        canvas.configure(yscrollcommand=scrollbar.set)
        scrollbar.pack(side="right", fill="y")
        canvas.pack(side="left", fill="both", expand=True)
        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")

        scrollable_frame.bind("<Configure>", lambda e:
canvas.configure(scrollregion=canvas.bbox("all")))

        for index, row in recommendations.iterrows():
            recipe_frame = tk.Frame(scrollable_frame, bg='#ffffff', bd=1, relief=tk.RIDGE)
            recipe_frame.pack(pady=10, padx=20, fill=tk.BOTH)

            recipe_name_label = tk.Label(recipe_frame, text=row['TranslatedRecipeName'],
font=("Helvetica", 16, "bold"), fg="blue", cursor="hand2", bg='#ffffff')
            recipe_name_label.bind("<Button-1>", lambda event, url=row['URL']:
self.callback(event, url))
            recipe_name_label.pack()

            details_label = tk.Label(recipe_frame, text=f"Prep Time: {row['PrepTimeInMins']} mins
| Cook Time: {row['CookTimeInMins']} mins | Total Time: {row['TotalTimeInMins']} mins |
Servings: {row['Servings']} | Cuisine: {row['Cuisine']} | Course: {row['Course']} | Diet:
{row['Diet']}", font=("Helvetica", 10), bg='#ffffff')
            details_label.pack(anchor=tk.W)

            instructions_text = row['TranslatedInstructions']
            instructions_text = instructions_text.replace('\n', '<br>')
            instructions_label = tk.Label(recipe_frame, text=instructions_text, font=("Helvetica",
12), justify="left", wraplength=900, bg='#ffffff')
            instructions_label.pack()

    def callback(self, event, url):
        self.root.clipboard_clear()
        self.root.clipboard_append(url)
        self.root.update()
        messagebox.showinfo("URL Copied", f"The recipe URL has been copied to your
clipboard:\n\n{url}")
        self.root.clipboard_clear()

if __name__ == "__main__":
    root = tk.Tk()
    app = TitleWindow(root)
    root.mainloop()
```

8. Conclusion

The Recipe Recommendation System effectively simplifies the process of finding suitable recipes based on user preferences. By integrating advanced text processing techniques and a content-based filtering algorithm, the system provides accurate and personalized recommendations. The GUI enhances user experience, making it easy to interact with the system and find the desired recipes quickly.

9. Future work

Future enhancements to the system could include Incorporating user feedback to improve recommendation accuracy. Adding support for multilingual inputs and outputs to cater to a broader audience. Integrating with external APIs to fetch the latest recipes and nutritional information. Expanding the dataset to include more diverse recipes and dietary preferences. Implementing collaborative filtering to recommend recipes based on user behavior and preferences of similar users.