BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE

DIPLOMA THESIS

Smart Flavor Pairing System for Culinary Innovation

Supervisor
Conf. Dr. Maria-Iuliana Bocicor

Author
Iancu Gheorghe-Aurelian

2024

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFROMATICĂ

LUCRARE DE LICENŢĂ

Sistem Inteligent de Combinare a Aromelor pentru Inovație Culinară

Conducător științific
Conf. Dr. Maria-Iuliana Bocicor

Absolvent
Iancu Gheorghe-Aurelian

2024

Abstract: Many of us have probably been in the situation where, when cooking, we feel like our dish is not complete but don't have an idea of what's missing. This thesis presents the development of a Smart Flavor Pairing System, driven by the personal motivation to enhance the taste of homemade dishes through an automated, scientifically based approach. To achieve this, various metrics like arithmetic mean, standard deviation and cosine similarity were used to quantify flavor compatibility, allowing us to perform a comprehensive analysis of how ingredients interact at a chemical level. An important part of this project involved creating a dataset for our specific requirement.

The exploratory nature of this research meant that the outcomes were questionable and it was challenging to determine the accuracy and correctness of the approaches only through statistical analysis. Practical testing is essential to validate the findings, but this requires significant resources and time. Future work could involve a more detailed implementation using advanced AI models to identify patterns within ingredient relations. Additionally incorporating factors such as ingredient quantity, nutritional values and cooking steps could further refine the recommendations.

Contents

Chapter 1

Introduction

Since the appearance of first humans, the necessity to eat has driven the evolution of cooking throughout history. From the discovery of fire to the development of modern culinary techniques, cooking has evolved continuously as people tried to enhance the flavors and nutritional value of their food. Creating the perfect dish is often a combination of intuition, experience and creativity, yet, even the most experienced cooks sometimes face those moments where something is missing, but they can't quite put the finger on what it is. This common cooking problem inspired this research. One day, while cooking at home, far from pleased of my dish's taste, I found myself thinking "What can I add to make it better?". This led me to consider the basic chemical compounds and flavors within the ingredients I was using. Each ingredient has a unique set of chemical compounds that contributes to its flavor profile and understanding these compounds could be the answer to our problem. This realization pointed to the need for an approach based on ingredient pairing, one that uses the science of flavor compounds to improve culinary creativity and innovation.

This thesis addresses the need for scientific and automated methods to pair ingredients based on their chemical flavor compounds, traditional cooking practices mainly relying on experience and trial and error. As a result, we propose the development of a Smart Flavor Pairing System. By using detailed chemical flavor data, this system should be capable to provide superior and innovative ingredient pairing recommendations. The goal is to combine computational methods with culinary arts to transform the way we select the ingredients within a recipe.

All this development will be approached from an experimental point of view. Our main challenge will be to quantify a recipe based on the number of shared flavor compounds between ingredients to understand how the addition of new ingredients influences the recipe. We will explore different methods for ingredient pairing and analyze their results to identify the most effective strategies for enhancing a dish.

The paper is structured in three chapters. The first chapter presents theoretical information and details about the research that has been done within the field of computational gastronomy. The second chapter presents our proposed solution and the experimental approaches that we tried. The third chapter presents the implementation of the application, its architecture and UI/UX details.

As a short review before we enter in the content of the paper, this thesis aims to assist those who wish to gain a better understanding of the concept of ingredient pairing and to find a way to improve recipes based on this concept.

1.1 Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author used Chat GPT in order to improve his grammar and provide a better flow to his ideas. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the thesis.

Chapter 2

Related Work

In this chapter, we have gone over the key ideas behind our application, like computational gastronomy,flavors, flavor pairing and recommendation systems. By looking at the latest research in these areas, we've set up a solid base for our project not only by showing the cool approaches others have taken but also pointing out where there's room for improvement, which is what we are aiming for. We consolidate this background to smoothly transition to the next chapter where we will present our proposed solution.

2.1 Computational Gastronomy

"Computational gastronomy is a field that combines technology and food to create culinary experiences that are unique and innovative" defines the concept of computational gastronomy. This field aims to use different metrics, formulas and training Artificial Intelligence models to improve the quality of the food we eat.[Tri23] Our point of interest in computational gastronomy is to give users the best recommendations of ingredients that they could add to their recipes based on the number of shared flavor compounds between ingredients.

Previous works present different perspectives and algorithms in this domain, illustrating different approaches and methodologies aimed to create the tastiest recipes. Approaches like recommendations based on the cooking instructions and ingredient tags [GPS+21], similarity between ingredients based on PMI scores(Point-wise Mutual Information) which is computed by comparing the probability to find two ingredients in the same recipe with the probability to find them in separate recipes, [TLA12] used to predict scores for unknown pairings [PKP+19] or to perform recommendations of new additions [GCM+22]. These approaches reflect some methods researchers have explored to enhance recipes.

2.1.1 Flavor Pairing

In our study we will take a different approach. The measure of compatibility between ingredients will be represented by how many flavor compounds a pair of ingredients share. Firstly, let's define the terms that we are going to work with. "Flavor is the sensory impression of a food or other substance and is resulted from the stimulation of the chemical senses of taste and smell". The smell plays a significant role in our perception of taste, estimating that up to 80% of what we taste is determined by our smell.[AS+22] The term compound refers to the chemical compounds present in the food that we eat. When this compounds come into contact with olfactory sensory neurons, the chemical signal is converted into neural impulses. As a result, neurochemical responses occur in numerous ares of the brain, resulting in the sense of smell.[KGB22]

In the previous paragraph it was proved that the chemical compounds within food aliments have a big influence on their taste. Now, let's see some examples of compounds with distinctive flavors.

1. **Menthol** is a chemical compound derived from mint plants with a strong minty odor and the chemical formula $C_{10}H_{20}O$ which can be seen in the Fig. 2.1

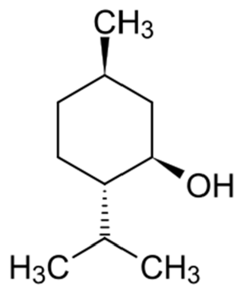


Figure 2.1: Menthol [FDB+15]

2. **Citral**, which is a combination of its two isomers "geranial" and "neral" has a lemony flavor. It has the $C_{10}H_{16}O$ chemical formula. [KGB22] The two isomers can be seen in the Fig. 2.2

3. **Vanillin** is found in vanilla beans and it is responsible for the characteristic vanilla flavor. Its chemical formula is $C_8H_8O_3$ and it can be seen in the Fig. 2.3

Multiple studies explored the relations between ingredients based on their flavor compounds. In their study Goel, Mansi and Bagler, Ganesh [GB22] explored the food pairing hypothesis proposed by Chef Heston Blumenthal, a master of molecular gastronomy, suggesting that ingredients sharing similar flavor compounds tend

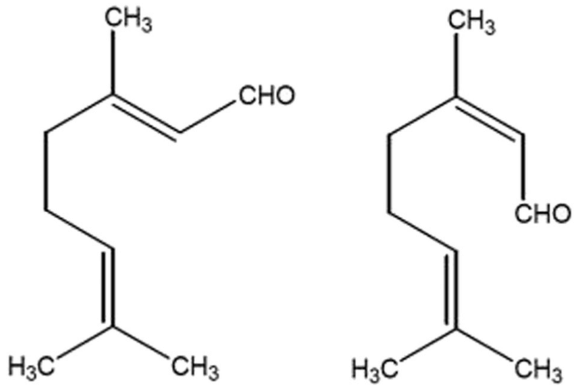


Figure 2.2: Citral [AVK+18]

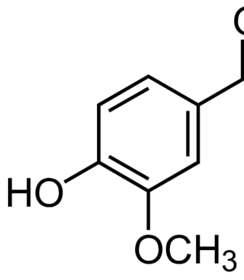


Figure 2.3: Vanillin [Gul20]

to pair well together. One of his first and most famous ideas of unusual combinations is white chocolate and caviar. [Blu08] The paper studies cuisines worldwide to identify such patterns in food pairings using Ahn et al.'s flavor network, network created from the same research reasons, showing that ingredients sharing flavor compounds are more likely to taste well together than the ones that don't.[AABB11] A visual representation of one of this network's approaches can be seen in Fig. 2.4 The paper states out that the hypothesis is not respected worldwide because by analyzing recipes from different cuisines, it was observed that cuisines like North American, Western European and Latin American respect the hypothesis while cuisines from East Asia show contrasting food pairing patterns.[GB22] In the same manner [JB+15] Indian cuisine has been analyzed with a similar idea to ours by trying to quantify the flavor sharing in a recipe using flavor pairings and their occurrences within recipes to show that spices are the reason Indian cuisine does not respect Blumenthal's hypothesis.

As a conclusion to this subsection, knowing all the information from above, we are trying a new approach of improving a recipe. We are trying to find a way to quantify the value of our recipe based on flavor compounds. This will allow us to modify the recipe by adding ingredients and observe how these changes affect it, in order to provide the bet possible improvements to the recipe.

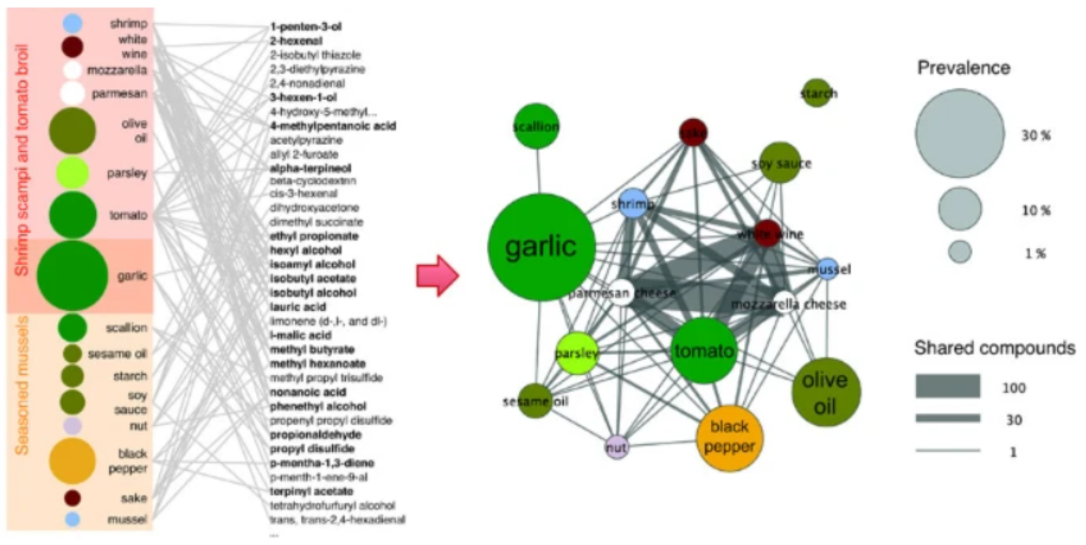


Figure 2.4: Flavor network based on ingredient-ingredient and ingredient-compound relations. [AABB11]

2.2 Recommendation systems

Along with the improvement algorithm, we'll implement a recommendation system for our users. Recommendation systems are software tools designed for providing suggestions of items that could be of interest to a certain user. [BDD$^+$12] When using an application, the user is exposed to a lot of data which represents much more information than a human can process. Because of such reasons, the recommendation's job is to filter the huge set of information based on each user's preferences. [ZPY$^+$23] We can say that recommendation systems are similar to search engines, but instead of the user actively searching for a product, the engine suggests items based on the user's previous interactions with the system. These interactions are used to create the user profile. [BFG11]

Big companies like Amazon, Netflix, Spotify, LinkedIn and many others use recommender systems to improve user experience, making their services more enjoyable and desirable. These systems are valuable across various domains. For example, they provide music and film recommendations on applications like Spotify and Netflix, suggest products that are similar or complementary to your purchases on Amazon, and offer social media recommendations with prompts like "You may also know" or "You may also like" on applications like LinkedIn.

Recommendation systems predict how much a user will like or dislike something based on various factors. Such factors are user preferences quantified by their implicit and explicit ratings, item attributes, user-item interactions, contextual information such as time, location or social context and user diversity. All this factors split the recommendation systems into three big categories [RD22].

2.2.1 Content-based filtering

Content-based recommendation systems are a popular and widely used approach when you want the recommendation to be predicted based on the user's personal preferences. When users give a positive rating to an item, the other items present in that item profile are aggregated together to build a user profile. The user profile then combines all the item profiles, whose items are rated positively by the user. Items present in this user profile are then recommended to the user. [RD22] For example, in the culinary domain, if a user likes a recipe from the Asian cuisine, it is very likely that the the system will consider recommending another Asian recipe. The entire process can be visualised in Fig. 2.5.

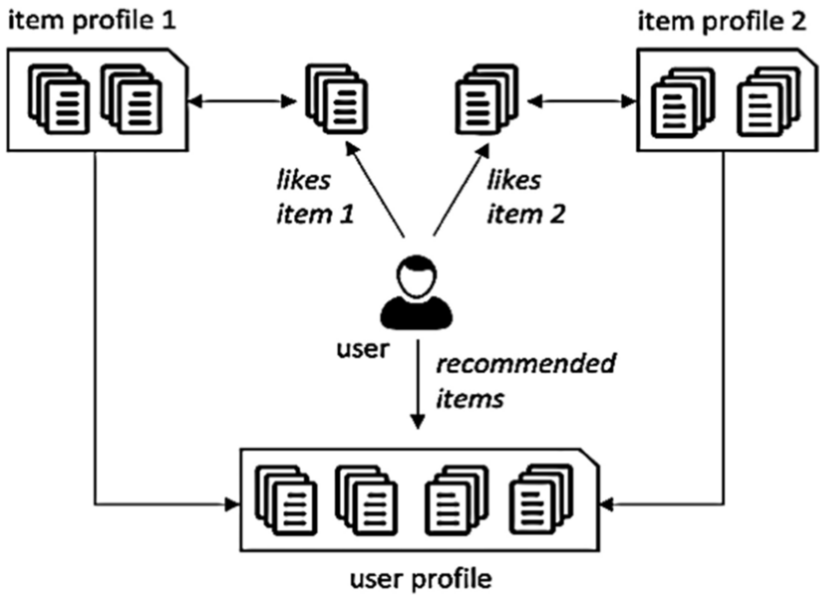


Figure 2.5: Content-based recommendation system [RD22]

An advantage of the content-based recommendations is that the prediction is only dependent on the content attributes. Additionally the "cold start" problem is not encountered because the recommendation can suggest new items to users without any information about user-item interactions.[MM23] One disadvantage of this approach is the necessity of detailed information of the item features for an exact prediction. [RD22]

2.2.2 Collaborative filtering

Collaborative filtering is used when you want your recommendations to be determined by the user relations. As the name says, there is a collaboration among the users, the predictions being done using the measure of similarity between them. The concept is pretty simple and feels a lot like natural language recommendations.

Let's suppose you are an user X and you like a recipe from a website. There is another user Y that also likes that recipe. Because you both liked the same recipe, it means that there is a compatibility between X and Y so the recommendation system will search in Y's list of preferences and recommend you something from that list.[RD22] This entire idea is quantified to groups of users that share similar preferences.

One of the main advantages of this approach is the independence of the item's feature. The entire idea behind this type of recommendation is the relation between users and because of that, the items are neglected. It is somehow considered that the similar user filtered the item information for us. An overview of the collaborative filtering can observed in Fig 2.6.

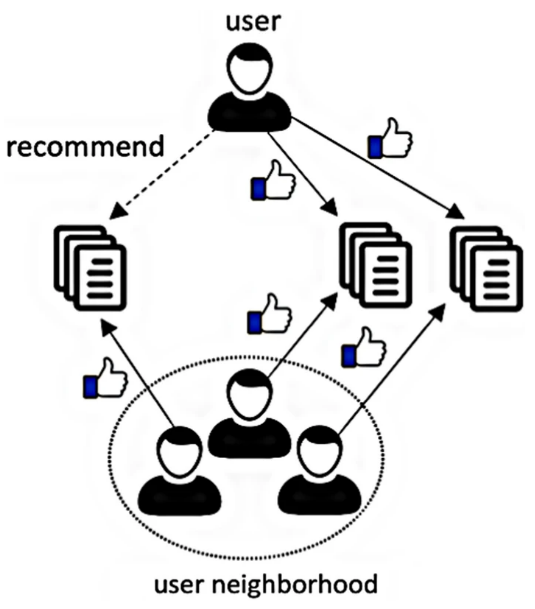


Figure 2.6: Collaborative recommendation system [RD22]

2.2.3 Hybrid approach

Hybrid recommendation systems combine multiple recommendation techniques to enhance the performance and avoid individual weaknesses. Approaches like "feature augmentation" represent one of the weak types of hybrid recommenders because one technique is still dominant, but it is supplemented by another. There are different ways in which we can combine the content-based filtering approach and the collaborative filtering approach to achieve the best hybrid recommender.[AK17] The visual representation of each combination can be seen in Fig. 2.7.

A. Implement two recommenders, one with the collaborative filtering approach and one with the content-based filtering approach and consider as recommendations their reunion.

B. Using content-based filtering characteristics within a collaborative filtering approach.

C. Creating a general model using the characteristics of both types and combining their result with a ML algorithm.

D. Using collaborative filtering characteristics within a content-based filtering approach.

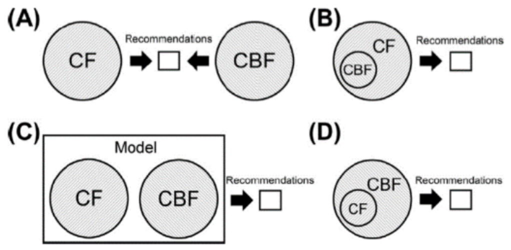


Figure 2.7: Methods of combining content-based filtering and collaborative filtering within a hybrid recommendation system. [AK17]

In this chapter we took a look at the most important theoretical concepts related to computational gastronomy, flavor pairing and recommendation systems. By understanding these information, we can now continue with the next chapter where we are going to present the solution to our idea.

Chapter 3

Proposed solution

In this chapter, we present the proposed solution for obtaining optimal flavor pairings based on the shared chemical compounds of ingredients. Our approach focuses on ingredients and their flavor profiles to provide accurate and innovative culinary recommendations. We will explain the specifics of the data sources utilized, the preprocessing steps taken to refine this data and the methodologies employed to achieve our goal.

3.1 Data preparation

One of the main challenges of this application was finding a dataset that works for the specific problem of this study. After long searches, something similar to what this paper is trying to accomplish could be found on this GitHub repository. [Lin] The repository contains multiple sets of data, however, the data about the recipes seemed to have a lack of details needed for the solution. The recommendation system requires comprehensive information about the recipes, such as name, ingredients, course, cuisine. The other type of recommendation, based on the number of common flavor compounds, also requires detailed information about ingredients, their types and the number of common flavor compounds shared by each pair of ingredients.

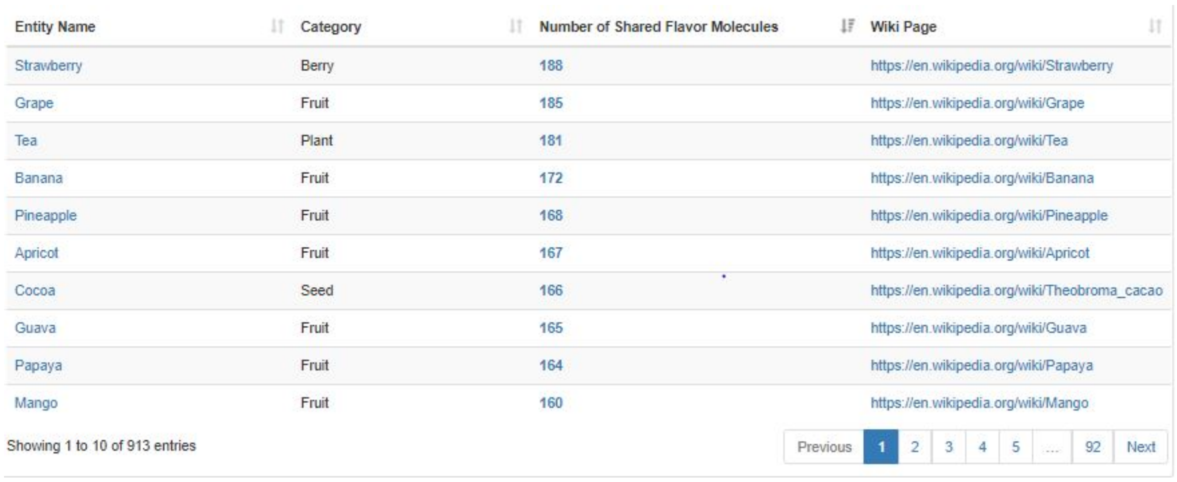
Unfortunately, a better dataset that suited all these requirements could not be found. Therefore a new dataset had to be created. It started from a dataset found on another GitHub repository [Yao] which is extracted from "Yummly". The data contains information about the recipes' name, rating, time of cooking, type of course, cuisine and ingredients. There were two main problems related to this dataset:

1. The ingredients were too ambiguous and way too detailed for our needs. Multiple ingredients that represented the same thing were listed separately. For example, "boneless skinless chicken breast halves", "chicken thighs" and "chicken"

all represent ways of saying that there was used chicken meat in that respective recipe. From the perspective of flavor compounds, there is no significant difference among these variations.

2. Additionally, the ambiguity in the form of our ingredients caused inconsistencies. Even thought a dataset with relations between ingredients and compounds was available from the GitHub repository [Lin], it could not be used because of the discrepancies between the ingredient names in our dataset and those in the ingredients-compounds dataset.

As a result of these problems, the data had to be processed. The goal was to refine the dataset so that the ingredients were not ambiguous, representing basic, simple, well-defined ingredients that could represent our recommendation of addition to a dish. To achieve this, we started with the dataset found at [Yao]. Additionally, there exists a paper [GST+18] that presents a web application called FlavorDB, where you can view ingredients and their relationships in a structured manner. This example from FlavorDB illustrates the type of ingredient detail we aimed for (see Figure 3.1). The reason for using simple, basic, well-defined ingredients is to ensure compatibility with the data that will be retrieved from FlavorDB, allowing us to accurately match ingredients and their flavor compounds.



Entity Name	Category	Number of Shared Flavor Molecules	Wiki Page
Strawberry	Berry	188	https://en.wikipedia.org/wiki/Strawberry
Grape	Fruit	185	https://en.wikipedia.org/wiki/Grape
Tea	Plant	181	https://en.wikipedia.org/wiki/Tea
Banana	Fruit	172	https://en.wikipedia.org/wiki/Banana
Pineapple	Fruit	168	https://en.wikipedia.org/wiki/Pineapple
Apricot	Fruit	167	https://en.wikipedia.org/wiki/Apricot
Cocoa	Seed	166	https://en.wikipedia.org/wiki/Theobroma_cacao
Guava	Fruit	165	https://en.wikipedia.org/wiki/Guava
Papaya	Fruit	164	https://en.wikipedia.org/wiki/Papaya
Mango	Fruit	160	https://en.wikipedia.org/wiki/Mango

Showing 1 to 10 of 913 entries

Previous 1 2 3 4 5 … 92 Next

Figure 3.1: Screenshot from FlavorDB application with Apple's Food Pairing Analysis [GST+18]

3.1.1 Ingredients mapping

This subsection's purpose is to describe the steps taken to transform the very ambiguous set of ingredients to a grouped one which we could rely on when trying to compute best additions for our dishes.

After removing from our recipes dataset the entities with empty data and with odd characters, we were left with 6285 recipes. From those recipes 3278 different ingredients were grouped out.

The first idea was to look at the ingredients lexical structure. We could observe 5 main problems:

1. **Existence of attributes:** The appearance of ingredients with specific attributes rather than their basic form (e.g., ″chopped walnuts″ instead of ″walnuts″, ″boiling water″ instead of ″water″, ″minced garlic″ instead of ″garlic″).

2. **Variations in form:** Different forms of the same ingredient listed separately either because of the multitude of details (e.g. ″boneless skinless chicken breast halves″, ″chicken thighs″), either because ingredients are in singular or plural forms inconsistently (e.g. ″tomatoes″, ″tomato paste″).

3. **Synonyms:** Multiple terms referring to the same ingredient (e.g. ″ravioli″, ″macaroni″, ″penne″, ″fettuccine″ all representing types of Pasta)

4. **Writing mistakes:** Errors in typing (e.g. ″jelli strawberri″ instead of jelly and strawberry).

5. **Brand-specific names:** Ingredients that appear with brand names rather than simple names (e.g. ″Skippy Creamy Peanut Butter″ instead of ″peanut butter″, ″Kewpie Mayonnaise″ instead of ″mayonnaise″).

6. **Compound ingredients:** Many of the ingredients were composed of multiple ingredients, and some of them were even entire dishes (e.g. ″prepared coleslaw″).

To solve these problems, we initially tried to group all the ingredients and split them by individual words. A dictionary was created to store each word along with the number of its appearances across all ingredients. The goal was to filter out the words that were relevant for us, keeping just the pure form of the ingredients necessary for our recommendation algorithm. Essentially, the aim was to have exactly the words that could be found on FlavorDB, allowing us to match the recipes dataset with the one containing flavor relations between ingredients.

As solution, we tried to identify the distinctive characteristics that the enumerated problems could bring. When looking at the ″Brand-specific names″ issue it was easy to observe that they start with capital letters. The question is ″Is it possible to have valid ingredient names with capital letters?″ (e.g. ″Beans″). To solve that, the capital words were filtered out, and the ones that could not be found in the dictionary in the non-capitalized form were removed. The ″Existence of attributes″

was partially solved using the "nltk" library which allows you to filter data based on their part of speech. The library was used to filter out only the nouns from our dictionary. Finally, matches of singular and plural forms related to the same word were created.

Despite these efforts, we could not be sure that the unwanted data was filtered out due to the various forms of attributes and the irregular plurals. Solutions for "Writing mistakes", "Compound ingredients" and "Variations in form" were not found. As a result, the data could not be filtered hard enough to provide an automated solution. To overcome this limitation, a CSV file with mappings for the 3000 ingredients was manually created. The file was structured in two columns: the actual name and the "translation". The ingredients that could not be translated were given a default value of "x".

From the point of view of the ingredients, 186 well-defined ingredients were mapped. Regarding the recipes, all recipes that contained at least one "x" were filtered out due to missing information. After all these processes, we were left with about 1300 recipes that can be cooked using the 186 ingredients.

3.1.2 Ingredient-ingredient relations

Now that we have some of our ingredients, we are going to use the FlavorDB tool [GST+18] to retrieve information about the relationships between our ingredients. A folder was created where for each one of the 186 ingredients a csv file was created named exactly how it is found on the FlavorDB website. Each one of those files contains three columns, representing the other ingredient from the ingredient-ingredient relation, the type of that ingredient and the number of flavor compounds they share.

We will now take a closer look at what this number of shared flavor compounds means from a mathematical point of view. Let's denote $I_n, n \in \{1, 2\}$, the nth ingredient from the ingredient-ingredient relation. We will also denote with FC_{I_n} the set of flavor molecules present in the I_n ingredient. Using this notations and denoting the number of shared flavor compounds with NSFC, we can compute it with the formula. 3.1. We are basically just taking the cardinal of the intersection set. A visual representation of the formula is presented in the Fig. 3.2.

$$NSFC_{12} = |FC_{I_1} \cap FC_{I_2}| \tag{3.1}$$

A concrete example of how the formula works will be analyzed on the onion-garlic relation. Information from FlavorDB [GST+18] will be used for that. In our example, Garlic contains 143 flavor molecules, while Onion contains 180 flavor molecules

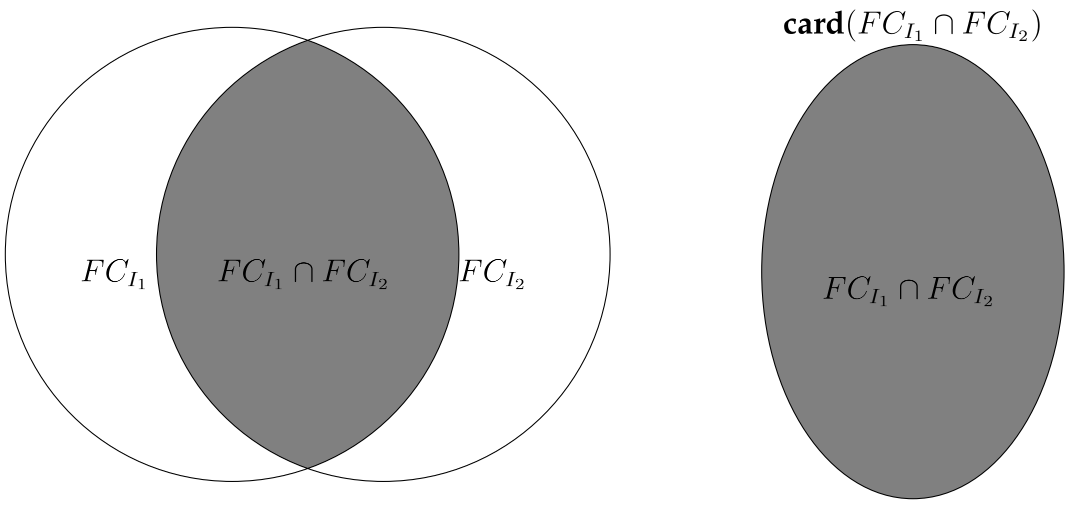
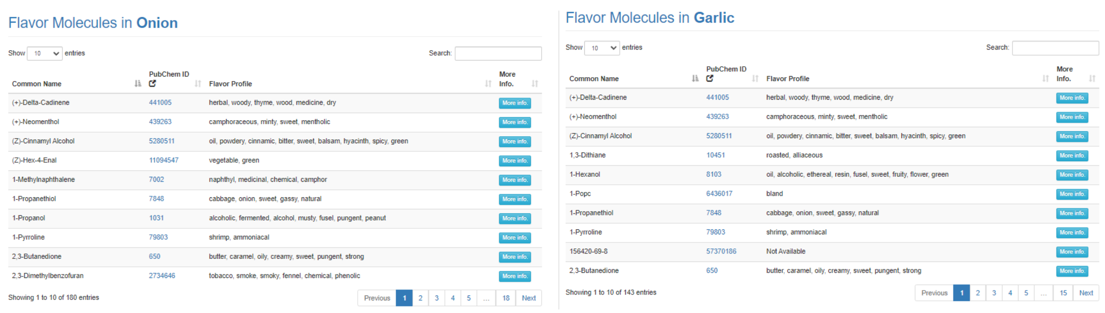


Figure 3.2: Venn diagrams representing the sets FC_{I_1} and FC_{I_2} and their intersection. The left diagram shows the full Venn diagram, while the right diagram highlights only the intersection.

Fig. 3.3. Out of those, 120 seem to appear in both of them, which means that the number of shared flavor compounds is 120. 3.4 Considering that they are a big percentage of the total molecules we would consider that a pretty good match. As a fact, based on this criteria, on FlavorDB, each one for the other appears to be the best match.



Flavor Molecules in **Onion**

Show 10 entries   Search:

Common Name	PubChem ID	Flavor Profile	More Info.
(+)-Delta-Cadinene	441005	herbal, woody, thyme, wood, medicine, dry	More info.
(+)-Neomenthol	439263	camphoraceous, minty, sweet, mentholic	More info.
(Z)-Cinnamyl Alcohol	5280511	oil, powdery, cinnamic, bitter, sweet, balsam, hyacinth, spicy, green	More info.
(Z)-Hex-4-Enal	11094547	vegetable, green	More info.
1-Methylnaphthalene	7002	naphthyl, medicinal, chemical, camphor	More info.
1-Propanethiol	7848	cabbage, onion, sweet, gassy, natural	More info.
1-Propanol	1031	alcoholic, fermented, alcohol, musty, fusel, pungent, peanut	More info.
1-Pyrroline	79803	shrimp, ammoniacal	More info.
2,3-Butanedione	650	butter, caramel, oily, creamy, sweet, pungent, strong	More info.
2,3-Dimethylbenzofuran	2734646	tobacco, smoke, smoky, fennel, chemical, phenolic	More info.

Showing 1 to 10 of 180 entries   Previous 1 2 3 4 5 … 18 Next

Flavor Molecules in **Garlic**

Show 10 entries   Search:

Common Name	PubChem ID	Flavor Profile	More Info.
(+)-Delta-Cadinene	441005	herbal, woody, thyme, wood, medicine, dry	More info.
(+)-Neomenthol	439263	camphoraceous, minty, sweet, mentholic	More info.
(Z)-Cinnamyl Alcohol	5280511	oil, powdery, cinnamic, bitter, sweet, balsam, hyacinth, spicy, green	More info.
1,3-Dithiane	10451	roasted, alliaceous	More info.
1-Hexanol	8103	oil, alcoholic, ethereal, resin, fusel, sweet, fruity, flower, green	More info.
1-Popc	6436017	bland	More info.
1-Propanethiol	7848	cabbage, onion, sweet, gassy, natural	More info.
1-Pyrroline	79803	shrimp, ammoniacal	More info.
156420-69-8	57370186	Not Available	More info.
2,3-Butanedione	650	butter, caramel, oily, creamy, sweet, pungent, strong	More info.

Showing 1 to 10 of 143 entries   Previous 1 2 3 4 5 … 15 Next

Figure 3.3: Onion and Garlic Flavor Molecules [GST+18]

Continuing with the dataset preparation, using a Python script and those files we managed to create one large dataset of relations between ingredients. Initially that file had a structure with four columns, representing the Ingredient1 which was taken from the name of the CSV, and the other three columns were taken from the respective CSV. Each ingredient file had around 900 rows of data, which meant that there were around 900 ingredient pairings for each ingredient. Having those repeating we grouped out around 800 new ingredients from their pairings.

Many of those new ingredients make no sense in our case and we had to filter them out. Among those reasons we can mention that some ingredients were actually complex dishes composed by multiple ingredients (e.g. cake), some ingre-

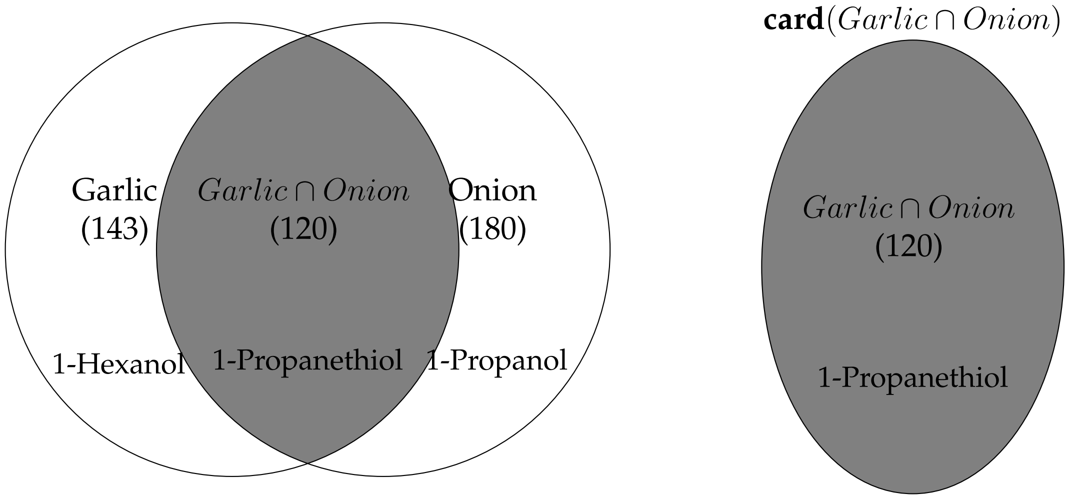


Figure 3.4: Venn diagrams representing the sets Garlic and Onion and their intersection.

dients were inedible, some were very specialized and only a small percentage of user would know how to use them(e.g. Narrowleaf Cattail), some were hard to procure or very rare(e.g. Black Bear meat), and some had no practical reasons in most recipes(e.g. Flour). After filtering the we were left out with 374 ingredients, 188 more than what we had before.

Using the new ingredients we extended the relations dataset with them being considered as the Ingredient1. The reason is related to the following implementations. Only the relations with those ingredients that were used to create the recipes were added because we want to add an ingredient to an already existent recipe which implies that we need only the relations with the ingredients within our recipes.

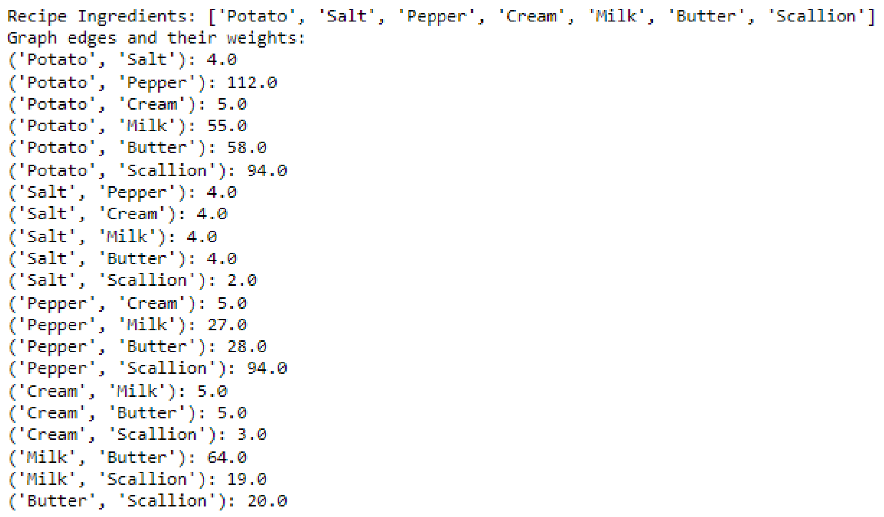
Now we have three well structured datasets, one dataset with ingredients, one with recipes and one with ingredients relations that contain the necessary information for checking our theory.

3.2 Quantification of flavor relations

In this section, we will discuss various approaches that have been attempted to solve our problem. Our goal is to identify the best possible ingredient to add to our recipe to enhance its flavor. These approaches are not necessarily solutions but rather exploratory methods to find optimal ingredient combinations. The primary factor used to determine the effectiveness of these combinations is the number of shared chemical compounds between ingredients.

3.2.1 Recipe Graph

As a first step of our research we tried to give each recipe a visual representation. We wrote a function that takes a recipe, the ingredients and their common flavor compounds and creates a graph using the networkx library. Each recipe has a representative graph where the edges represent the ingredients and the weights of the vertices between them represent the number of flavor compounds they share. Fig. 3.5 provides a visual representation of the ”Sour Cream Mashed Potatoes”’s flavor graph. As you can see the nodes are Cream, Pepper, Salt, Potato, Scallion, Butter, Milk and Cream and between each pair there is a vertex with the weight representing how many chemical flavor compounds they share.



```
Recipe Ingredients: ['Potato', 'Salt', 'Pepper', 'Cream', 'Milk', 'Butter', 'Scallion']
Graph edges and their weights:
('Potato', 'Salt'): 4.0
('Potato', 'Pepper'): 112.0
('Potato', 'Cream'): 5.0
('Potato', 'Milk'): 55.0
('Potato', 'Butter'): 58.0
('Potato', 'Scallion'): 94.0
('Salt', 'Pepper'): 4.0
('Salt', 'Cream'): 4.0
('Salt', 'Milk'): 4.0
('Salt', 'Butter'): 4.0
('Salt', 'Scallion'): 2.0
('Pepper', 'Cream'): 5.0
('Pepper', 'Milk'): 27.0
('Pepper', 'Butter'): 28.0
('Pepper', 'Scallion'): 94.0
('Cream', 'Milk'): 5.0
('Cream', 'Butter'): 5.0
('Cream', 'Scallion'): 3.0
('Milk', 'Butter'): 64.0
('Milk', 'Scallion'): 19.0
('Butter', 'Scallion'): 20.0
```

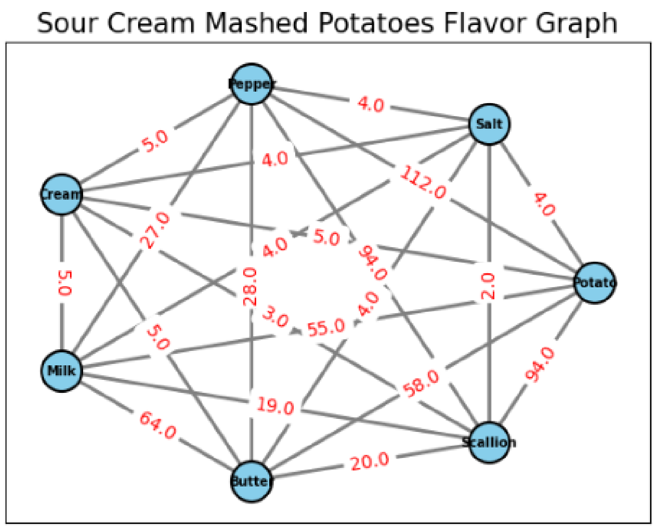


Figure 3.5: Sour Cream Mashed Potatoes Flavor Graph

At this point, the problem relies on adding a new node to the graph and analyze the behaviours of the changes. In order to do that we tried to quantify the value of those weights for the entire graph in order to give that a value. We will call that value FCS(Flavor Compatibility Score) and we will analyze the values of the FCS before and after each of the new ingredients were added to the graph.

3.2.2 Computing FCS using the arithmetic mean

Initially we tried to calculate the FCS using the most intuitive mathematical formula, the arithmetic mean. The value of the FCS could be calculated summing the weights of all the vertices in the graph and dividing it by their number.

Let's denote RE the set of all ingredients I_n within a recipe. Let's also denote sc_{ij}, the number of shared compounds between ingredient I_i and ingredient I_j. Using this notations, the formula for FCS is 3.2.

$$\text{FCS} = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sc_{ij}}{\frac{n*(n-1)}{2}}, n \in (1, (card(RE)) \text{ and } n \in \mathbf{N}, \tag{3.2}$$

Using the above formula we can calculate the FCS of our our recipe. In the case of the Sour Cream Mashed Potatoes recipe, the FCS value is 29.3333.... Now we want to analyze what happens with this score when we try to add new ingredients. According to Blumenthal's theory, ingredients that share more flavor compounds should taste better together. Considering that, we will try to see which ingredients raise the value of the FCS the most.

After trying the formula for all the ingredients additions we could extract some statistics and watch some behaviours. For our example the top 5 ingredients that raised the values of the FCS can be seen in the table 3.1. Starting from these recommendations and considering that we don't really have a way of testing the results because it requires resources and a lot of practical work in the kitchen, we tried to be as critic as possible with the interpretation of our results. Considering that, questions like "Who would add Papaya to Mashed Potatoes?" or "Why are there only fruits?" appeared. An overview perspective of the recommendations was taken by taking all the recommendations for all the recipes and grouping them within a dictionary of occurrences. We could see that out of all the ingredients only 57 of them got to be recommended, half of them being recommended less in less than 10 times. The reason for that is that the top ingredients appeared too many times and monopolized the recommendations. Top 5 ingredients 3.2, appeared over 300 times in recommendations, with the Black Currant occupying the first place with 624 aparitions. That's a lot considering that we have 1300 recipes. It means that it will be recommended in approximately 1 out of 2 recipes.

It seems like this approach would not solve our problem, but let's see what would be the problem and how we could solve it. Well, by looking at the ingre-

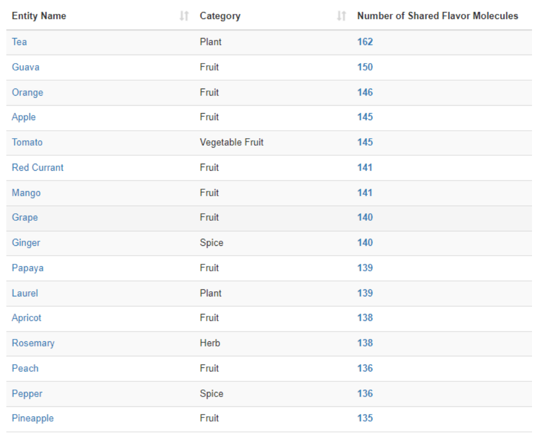
Ingredient	FCS
Papaya	68.29
Peach	65.86
Black Currant	65.57
Capsicum	65.29
Apple	65.14

Table 3.1: Best improvements of FCS using arithmetic mean.

Ingredient	Number of occurences
Black Currant	624
Papaya	604
Apple	416
Spearmint	411
Mandarin Orange	346

Table 3.2: Top 5 ingredients by how many times they were recommended using arithmetic mean.

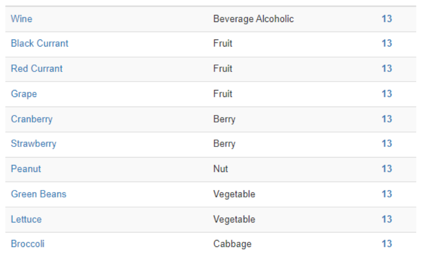
dients relationships we can see through FlavorDB [GST+18] that ingredients like Black Currant, Papaya and Apple that are highly recommended share over 100 flavor compounds with over 100 ingredients 3.6 and over 90 flavor compounds with over 300 ingredients, while ingredients like Chocolate don't share more than 13 flavor compounds with any ingredient. 3.7 Knowing this information, it makes a lot of sense to have such ingredients appearing in most of our recommendations because the formula fits perfectly for them.



Entity Name	Category	Number of Shared Flavor Molecules
Tea	Plant	162
Guava	Fruit	150
Orange	Fruit	146
Apple	Fruit	145
Tomato	Vegetable Fruit	145
Red Currant	Fruit	141
Mango	Fruit	141
Grape	Fruit	140
Ginger	Spice	140
Papaya	Fruit	139
Laurel	Plant	139
Apricot	Fruit	138
Rosemary	Herb	138
Peach	Fruit	136
Pepper	Spice	136
Pineapple	Fruit	135

Figure 3.6: Black Currant top flavor relations[GST+18]

The arithmetic mean doesn't seem to be a solution to our problem so we'll try to find another approach which gives us better recommendations.



Wine	Beverage Alcoholic	13
Black Currant	Fruit	13
Red Currant	Fruit	13
Grape	Fruit	13
Cranberry	Berry	13
Strawberry	Berry	13
Peanut	Nut	13
Green Beans	Vegetable	13
Lettuce	Vegetable	13
Broccoli	Cabbage	13

Figure 3.7: Chocolate top flavor relations [GST+18]

3.2.3 Computing FCS using standard deviation

Another approach similar with the arithmetic mean, but which is trying to solve the previous stated problem is the standard deviation.

In our case standard deviation measure the variability or consistency of the shared compounds between a potential new ingredient and the existing ingredients in a recipe. For each potential new ingredient, we create a list of shared compounds with each existing ingredient in the recipe and calculate its standard deviation.

Let $S = sc_{i1}, sc_{i2}, ..., sc_{ik}$ be the set of shared compound scores between the potential new ingredient I_{new} and each existing ingredient I_i in the recipe. The formula for the standard deviation σ_i of an ingredient I_i can be seen at 3.3, where:

- k is the number of existing ingredients in the recipe.
- sc_ij is the number of shared compounds between I_{new} and I_i.
- μ_i is the mean of the shared compound score, given by3.2.3

$$FSC = \sigma_i = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(sc_{ij} - \mu_i)^2} \tag{3.3}$$

$$\mu_i = \frac{1}{k}\sum_{j=1}^{k} sc_{ij} \tag{3.4}$$

Using the standard deviation we can interpret the result in 2 different ways:

- The standard deviation is low which indicates the new ingredient shares a similar number of compounds with many of the existing ingredients, suggesting uniform compatibility. In our case, this approach raises a problem similar to the last one where ingredients like Jalapeno and Prawn which have a very low number of flavor relations fit the formula very well and monopolize the recommendations.

- The standard deviation is high which indicates a wide range of shared compounds, suggesting that the new ingredient may pair well with some ingredients but not others, leading to less predictable improvements. In our case, the improvements seem to be better comparing them with the ones of the arithmetic mean. They are spread among a larger number of ingredients, 123 in this case and by eye it can be seen that the recipes with a salty flavor profile tend to have spices recommended and the recipes with a sweet flavor profile tend to have fruits, the fruits that used to monopolize the recommendation at the previous approach. Even with this improvements it is visible that some of the ingredients are recommended at a large scale and some are not even considered, in this case instead of fruits, there are 5 spices, Rosemary, Basil, Spearmint, Ginger and Oregano that appear in over 350 recommendations. It is a better approach than calculating the arithmetic mean, but it doesn't seem to be a sufficient solution.

3.2.4 Computing FCS using cosine similarity

Another approach that seemed to make sense was to compute the FSC using the cosine similarity. Cosine similarity effectively measure how similar two ingredient vectors are by comparing the angle between them. This is particularly useful for understanding the degree of similarity in flavor profiles between ingredients. We will use the cosine similarity to see how similar two ingredients are.

Using the cosine_similarity function from the sklearn library, we computed the matrix of cosine similarities where each entry(i,j) represents the cosine similarity between ingredient i and ingredient j. The formula for the cosine similarity between the ingredient I_{new} and I_i can be at 3.5.

$$\cos(\theta_i) = \frac{I_{\text{new}} \cdot I_i}{|I_{\text{new}}||I_i|} \tag{3.5}$$

In order to see how similar the new additions are with our recipe, we will perform the average of the cosine similarities between I_{new} and the rest of the ingredients of the recipe and that will be our FCS score. The formula can be seen at 3.6 where:

$$\text{FCS} = \frac{1}{k}\sum_{i=1}^{k} \cos(\theta_i) \tag{3.6}$$

- k is the number of existing ingredients in the recipe.
- $I_{\text{new}} \cdot I_i$ is the dot product of the vector for the new ingredient and the vector for the i-th existing ingredient.

- $|I_{\text{new}}|$ and $|I_i|$ are the magnitudes (Euclidean norms) of the vectors for the new ingredient and the i-th existing ingredient, respectively.

Cosine similarity seems like a great fit for our problem because:

- Cosine similarity measures the cosine of the angle between two vectors, which effectively quantifies how similar the flavor profiles of two ingredients are.
- It solves the sparsity problem focusing on the presence and absence patterns rather than absolute values.
- By focusing on the angle between vectors, the cosine similarity ensures that recommendations are performed based on the overall pattern of shared flavor compounds, leading to balanced and diverse suggestions.
- It solves the initial problem where some of the ingredients monopolized the recommendations by normalizing the data.

In the case of the cosine similarity the scores are going to look different. That's because using the cosine function, the values will rely in the interval [-1, 1], 1 being the perfect similarity and -1 being the perfect dissimilarity. For the Sour Cream Mashed Potatoes example, the top 3 scores can be seen in the table 3.3.

Ingredient	FCS
Sweet Pepper	0.402
Paprika	0.402
Beans	0.393

Table 3.3: Best improvements of FCS using cosine similarity.

Using this approach we observed a higher distribution of the recommended ingredients without obvious patterns, unlike the previous methods. While this is still and experimental approach and we cannot be certain that the recommendations are correct, the cosine similarity method appeared to yield the best results among the three approaches.

According to [PKK+21], "we found that the food ingredients in same categories tend to have similar chemical structures.". To improve the perspective of our recommendations we will provide the recommendation by type. We grouped the ingredients from our database into 9 major groups: Vegetable, Fruit, Spice, Animal Product, Dairy, Additives and Essences, Beverage, Cereal and Nuts and Seeds having the following distribution 3.8. We will consider grouping of recommendations based on the type of ingredients an upgrade because it gives us a perspective on the bigger picture of our recommendations. Grouping them in such a way makes a lot of sense considering the fact that food ingredients from the same category tend to have similar chemical compounds. In this way we can see which types represent better recommendations for our recipes.

Our example is pretty bad when we try to analyze the structured data based on types because the top scores of all the types rely within 0.3 and 0.4. Considering another example we can observe that for Peanut Butter Fudge groups like Additives and Essences provide much better recommendations than Nuts and Seeds. (See Fig. 3.9)

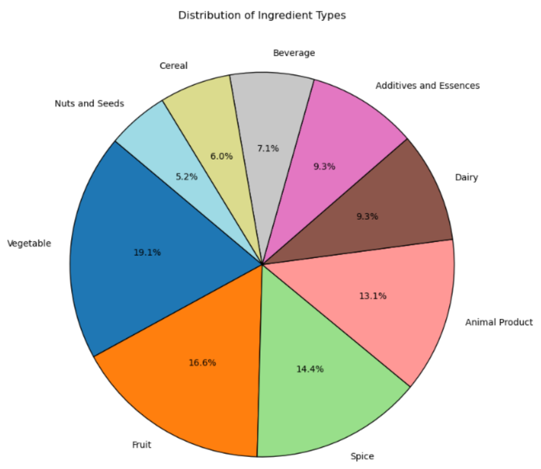


Figure 3.8: Distribution of ingredients by type

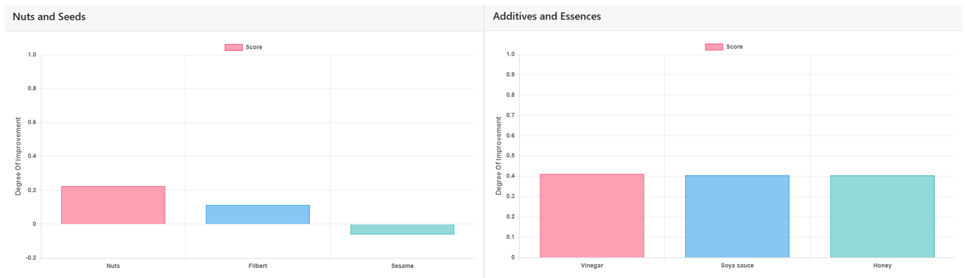


Figure 3.9: Peanut Butter Fudge Nuts vs Additives

As a conclusion to this chapter, we can say that finding a way to quantify the value of a recipe based on the flavor compounds is no easy task and requires higher-level computations and detailed information to improve the credibility of the improvements. Another perspective to enhance these improvements would be to consider different factors in the computations, similar to how we performed the recommendations based on the type of the ingredient. It is also important to note that practical experimentation is essential. While theoretical scores and graphical analyses are valuable, they must be validated through real-world testing to determine their actual effectiveness. Without practical application and testing, we cannot be certain that our solution are truly successful.

In the next chapter we will talk about how we implemented the application where you can see some practical utilities of what we stated in this chapter.

Chapter 4

Software Application

In the Software Application chapter, a close look will be taken at how the software for this study was created. The chapter will analyze the architecture and the technologies used to implement the application. An overview of the implementation and the user interface will also be provided for a better understanding. Its goal is to provide a clear presentation of how we got from our idea to the final product. We will understand how some of the flows of the application work using diagrams and also present the technical processes necessary to develop our application. Additionally, we will include screenshots to provide a visual representation of the application.

4.1 Architecture

In order to have a better understanding of the architecture of the application, we will use a series of UML (Unified Modeling Language) diagrams. "The UML is a collection of diagrams for specifying various aspects such as requirements and design of software systems." [A+12] UML diagrams are often used in scientific papers because they provide a detailed visual representation of how a software application works. It is easier and more pleasant to follow an image containing figures, arrows and labels than to understand the same thing using words. [DP06]

4.1.1 Use Case Diagram

Use case diagrams are a type of UML diagram used to show the relationships between the actors of an application and the use cases. Actors represent the different types of users using the application, while use cases represent the features available in the application that the user can interact with. Such diagrams are used to set the boundaries for each actor within our application, as each type of user may interact with various use cases within our system. [A+12]

In our case, the diagram will have only one actor, simply called "User", as we do not differentiate between different types of users. From the perspective of the use cases, our application provides a series of use cases representing its major features. The diagram can be seen in Fig 4.1

- Register: This feature allows users to create an account within the application. Users provide information such as their name, email, password(with confirmation), and the address to register.
- Login: Users owning an account can access the application using the email and the password.
- Logout: Users that are logged into the application can safely exit their account.
- View Recipe List: The list of recipes available in our database is nicely presented in a structured layout using grids, where each recipe is displayed as a card. Information such as name, cuisine, course, ingredients and cooking time is available for each recipe. Additionally, there is a button labeled "Improvements" that allows users to view recommended improvements based on shared flavor compounds for each of the recommended recipes.
- Pagination: This feature divides the list of recipes into smaller chunks, displaying only a portion of the data at a time. This approach prevents overcrowding the application interface and improves the performance by loading less data. Users can navigate through the pages to view more recipes.
- Search: Users can search and filter out recipes by their name. Only the recipes that match the user's preferences will be listed on the screen.
- Details: Users can check more details of a recipe by clicking the "Details" button. They will be redirected to the "Yummly" page of that specific recipe.
- Like Recipe: This feature is one of the most important for our application. It allows users to personalize their profiles by liking recipes that match their preferences. Users can also remove a like if they consider they don't like the recipe anymore. This process is done by clicking on the heart icon within each recipe card.
- View Favorites: Users can see their liked recipes filtered out on a separate page.
- View Recommendations: The list of recommendations based on the user's like is presented in a table format. Each recipe provides information similar to the "View Recipe List" use case.

- View Improvements: Users can see the available improvements for each recipe. The improvements are presented using charts and are grouped by score and the category of the ingredient. The charts allow the users to easily understand and compare the potential enhancements for their recipes based on different factors.

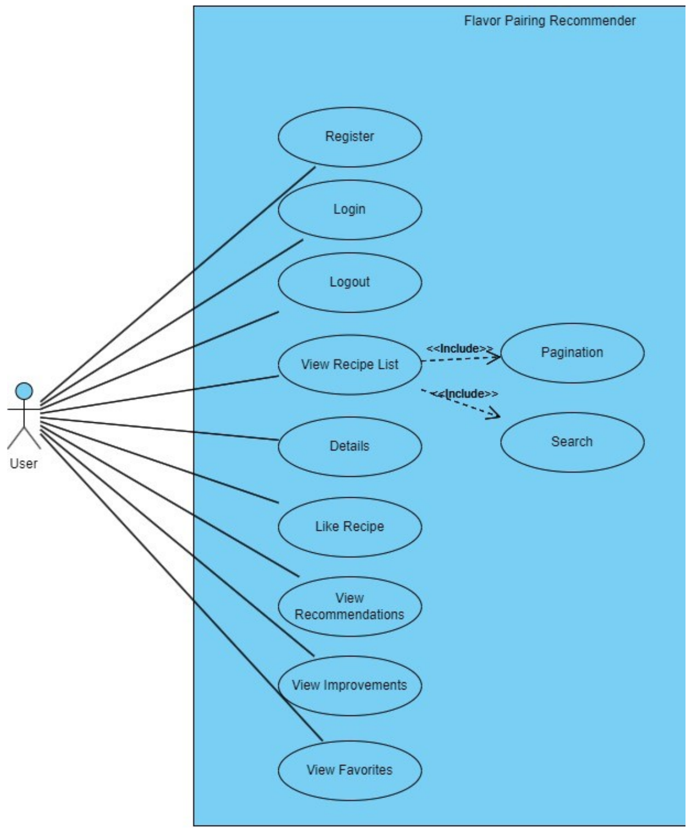


Figure 4.1: Use Case Diagram

4.1.2 Sequence Diagrams

In software engineering, a sequence diagram illustrates how objects interact over time within a system. They show the flow of events and the interactions between system objects. Such diagrams are useful for illustrating complex interactions between components. We will use such diagrams to provide a better, more detailed presentation of the most important use cases mentioned in the precedent subsection. It also illustrates how different features perform requests to the layers of our application. Our application follows a three-layer architecture consisting of presentation layer, business layer and data access layer.

Register Sequence Diagram

Fig 4.2 presents the flow of the "Register" use case. In order to have recommendations we need users and in order to have users we need accounts. To create an account a use has to navigate to the registration page, complete the form and submit it. If the data completed is valid a new account will be created. In the case of registration, the presentation layer is represented by the website itself and its interface, the business layer is represented by the server that's performing the requests and the data access layer is represented by the database. These attributions apply to nearly all our diagrams.

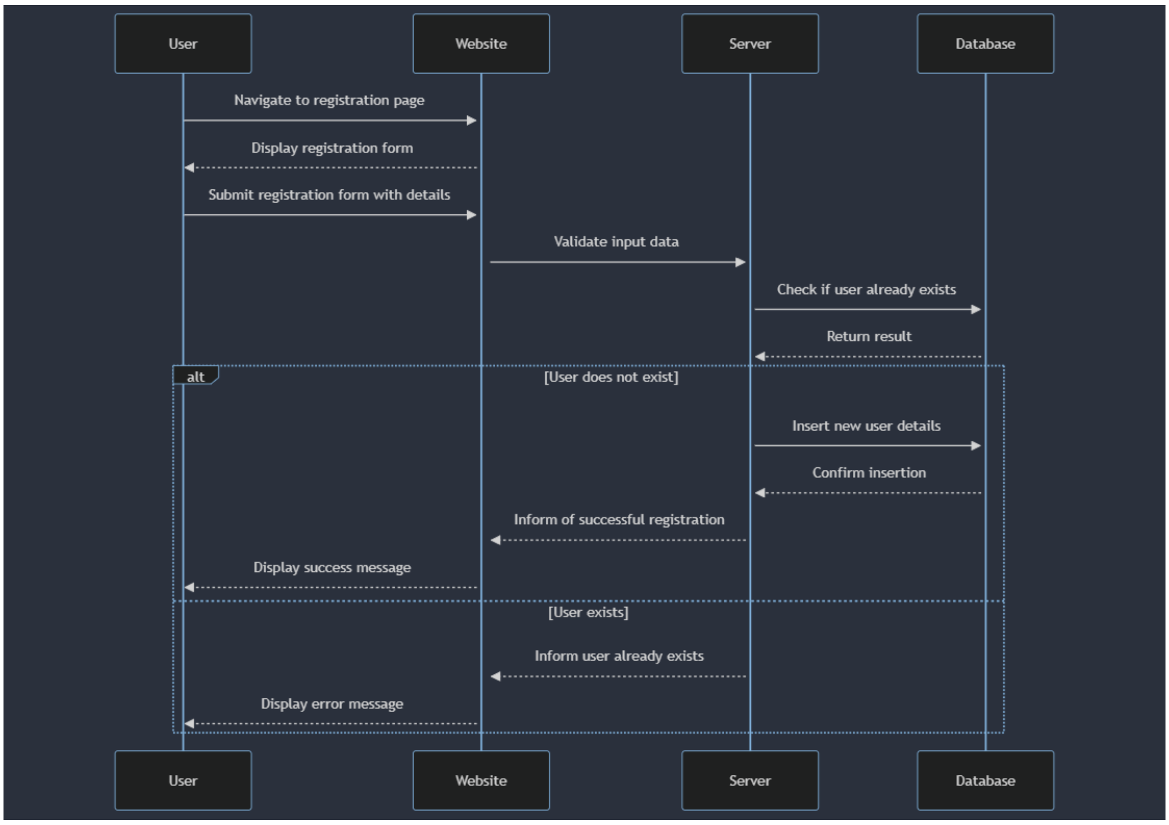


Figure 4.2: Register Sequence Diagram

Login Sequence Diagram

Fig 4.3 presents the flow of the ”Login” use case. The flow is intuitive and straightforward. Users that own an account can navigate to the login page, complete and submit the Login form. If there exists an account with those credential, they will be redirected to the main page of our application.

Like/Unlike Sequence Diagram

Fig 4.4 presents the flow of the ”Like Recipe” use case. The process begins on the recipes page, where the user can view the list of recipes. The user can click the like/unlike button for a specific recipe. If the button is clicked to like a recipe, a

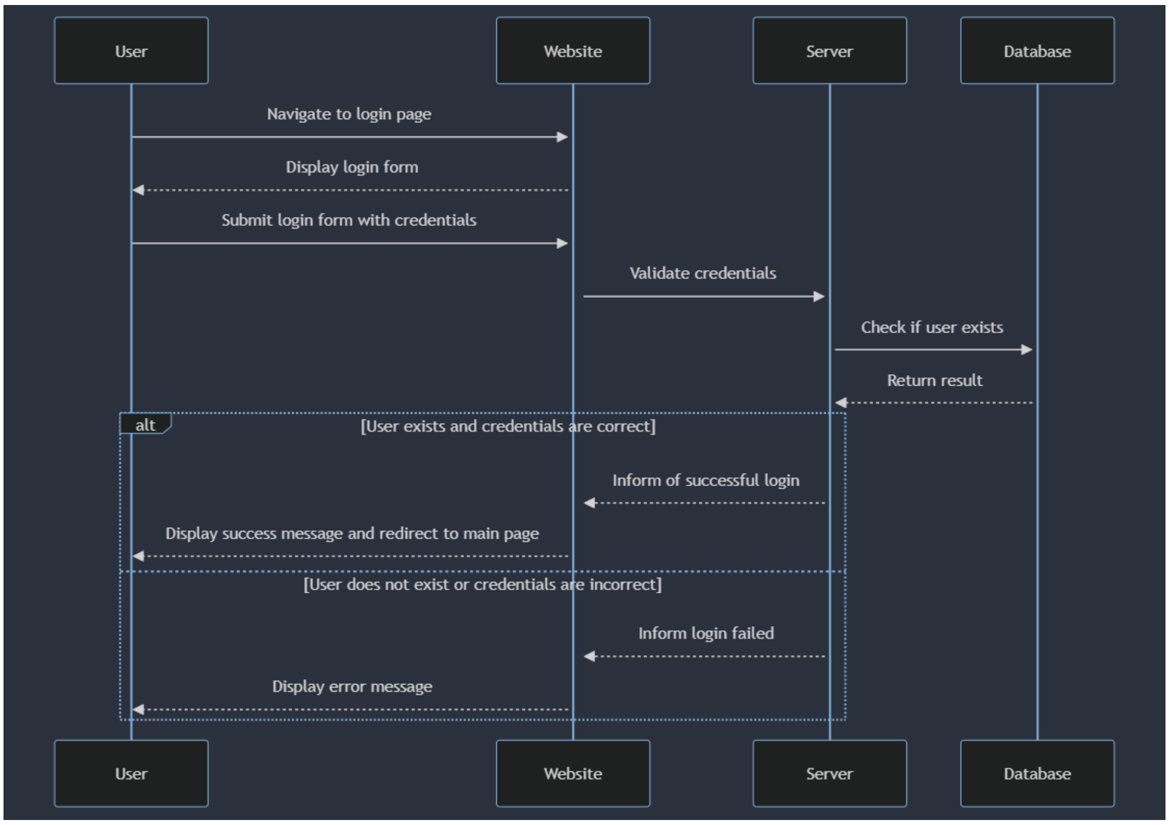

User
Website
Server
Database
Navigate to login page
Display login form
Submit login form with credentials
Validate credentials
Check if user exists
Return result
alt
[User exists and credentials are correct]
Inform of successful login
Display success message and redirect to main page
[User does not exist or credentials are incorrect]
Inform login failed
Display error message
User
Website
Server
Database

Figure 4.3: Login Sequence Diagram

request is sent to the server, which then updates the database to reflect the new like and confirms the update to the user by changing the like button to a full heart. On the other hand, if the button is clicked to unlike a recipe, the server removes the like entry from the database and confirms the update to the user by changing the like button to an empty hearth.

Favorites Sequence Diagram

Fig 4.5 presents the flow of the ”View Favorites” use case. The process begins when a user navigates to the favorites page. The website then requests the user’s liked recipes from the server. The server retrieves the favorite recipes from the database and returns them to the website. Finally, the website displays the list of favorite recipes to the user. This feature allows users to access faster and get a bigger picture of their preferences.

Recommendations Sequence Diagram

Fig 4.6 presents the flow of the ”View Recommendations” use case. The process begins when a user navigates to the recommendations page. The website then requests recommendations based on the user’s ID from the server. The server redirects this request to a Python REST API. The REST API which is connected to the same

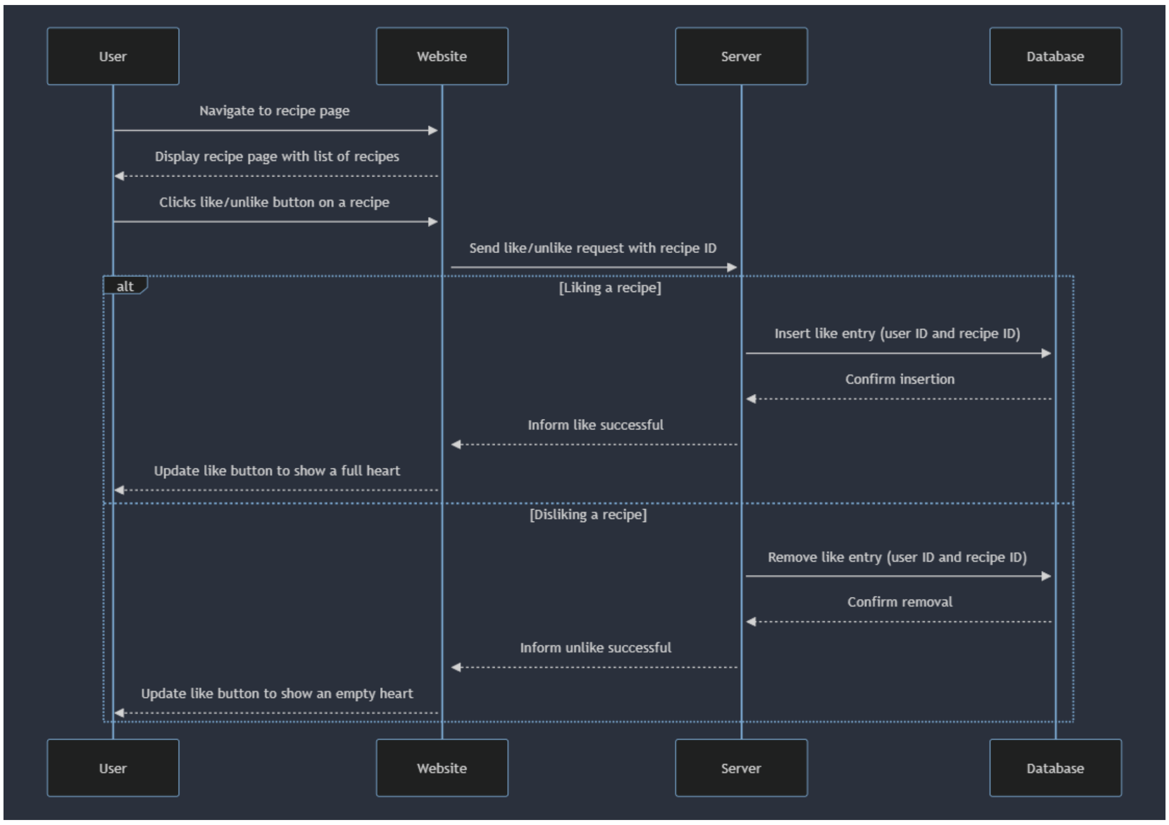

User
Website
Server
Database
Navigate to recipe page
Display recipe page with list of recipes
Clicks like/unlike button on a recipe
Send like/unlike request with recipe ID
alt
[Liking a recipe]
Insert like entry (user ID and recipe ID)
Confirm insertion
Inform like successful
Update like button to show a full heart
[Disliking a recipe]
Remove like entry (user ID and recipe ID)
Confirm removal
Inform unlike successful
Update like button to show an empty heart
User
Website
Server
Database

Figure 4.4: Like/Unlike Sequence Diagram

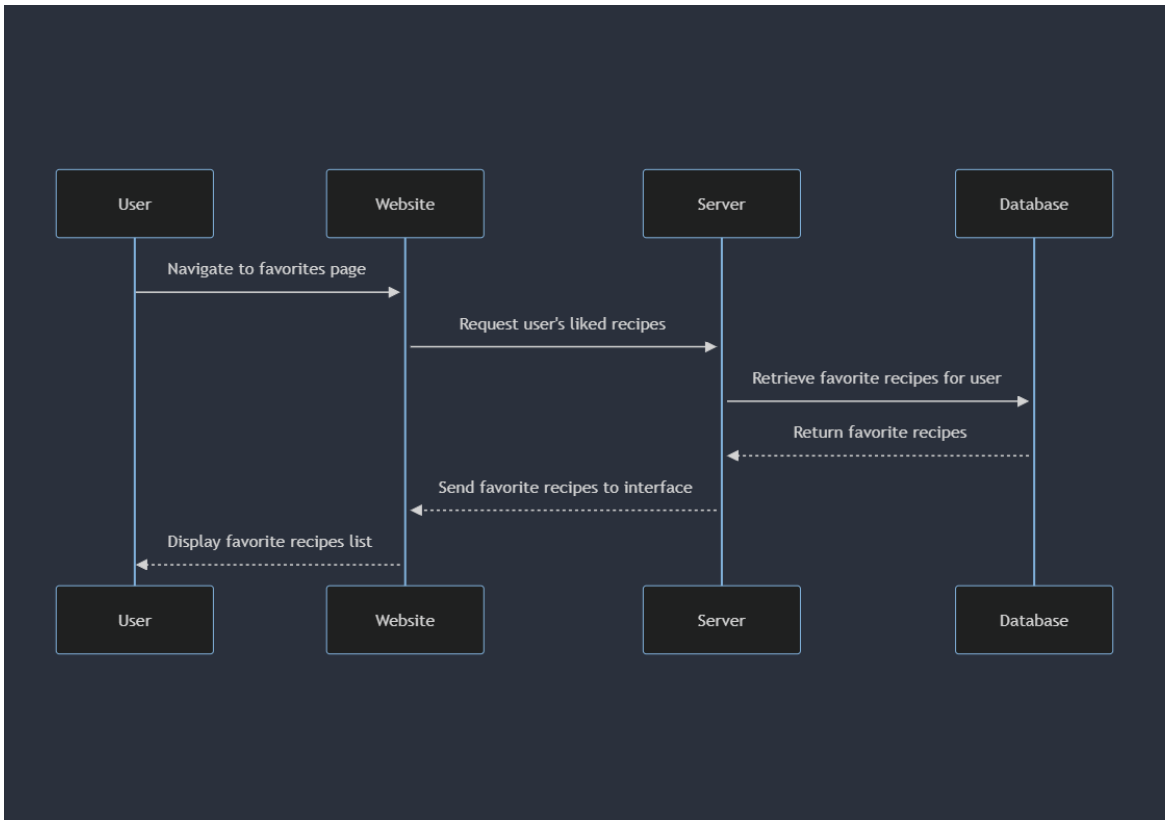

User
Website
Server
Database
Navigate to favorites page
Request user's liked recipes
Retrieve favorite recipes for user
Return favorite recipes
Send favorite recipes to interface
Display favorite recipes list
User
Website
Server
Database

Figure 4.5: Favorites Sequence Diagram

database as the server retrieves the user's likes from the database then performs a recommendation algorithm with specific Python libraries and sends the recommended items back to the server. Finally, the server sends these recommendations to the website, which displays the recommendations list to the user.

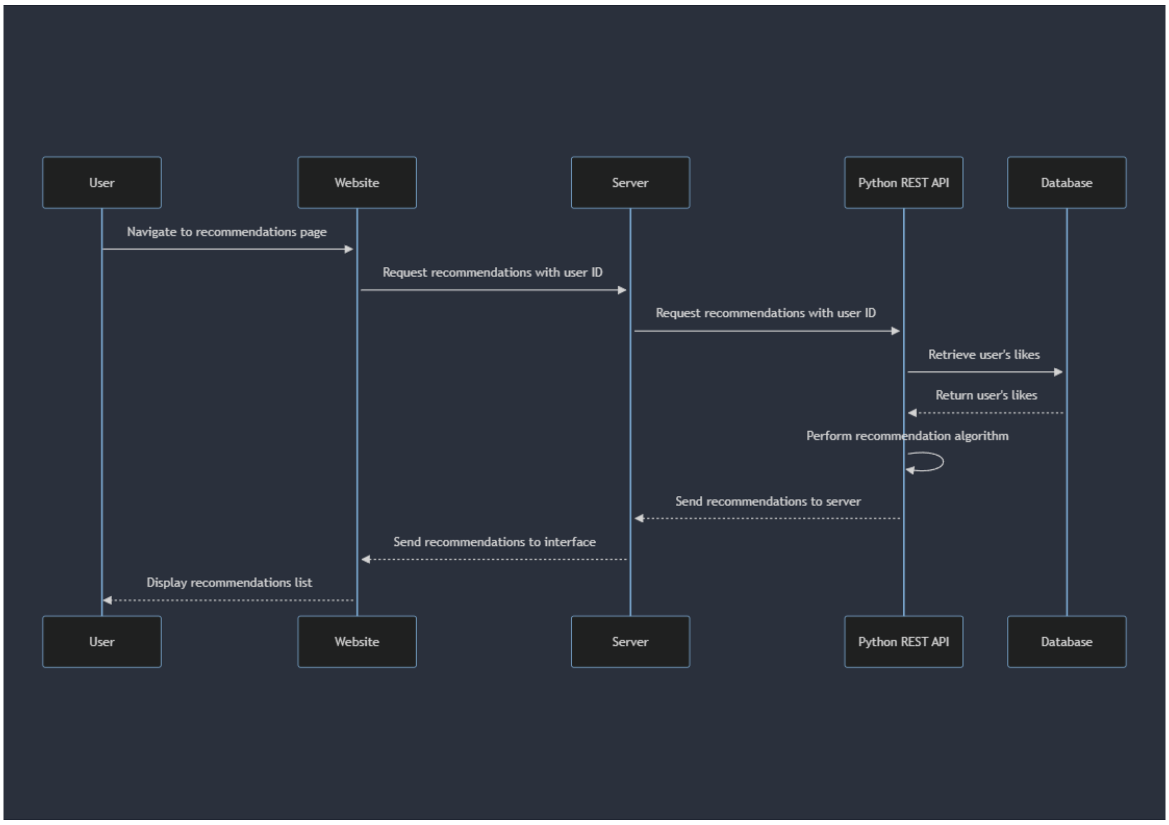

User
Website
Server
Python REST API
Database
Navigate to recommendations page
Request recommendations with user ID
Request recommendations with user ID
Retrieve user's likes
Return user's likes
Perform recommendation algorithm
Send recommendations to server
Send recommendations to interface
Display recommendations list
User
Website
Server
Python REST API
Database

Figure 4.6: Recommendations Sequence Diagram

Improvements Sequence Diagram

Fig 4.7 presents the flow of the "View Improvements" use case. The process begins when a user navigates to the recipes page. The website then requests the available recipes from the server. The server retrieves the recipe from the database and sends them back to the website which displays the recipes to the user. The user then proceeds to click on the "Improvements" button of a specific recipe. The website requests the improvements of the specific recipe to the server. The server retrieves the improvements from the database and sends the information back to the website which is display it after redirecting the user to another window.

4.1.3 Database Diagram

Fig. 4.8 presents the structure of the database that we are using for our application. It provides a visual representation of the tables that illustrate the four entities

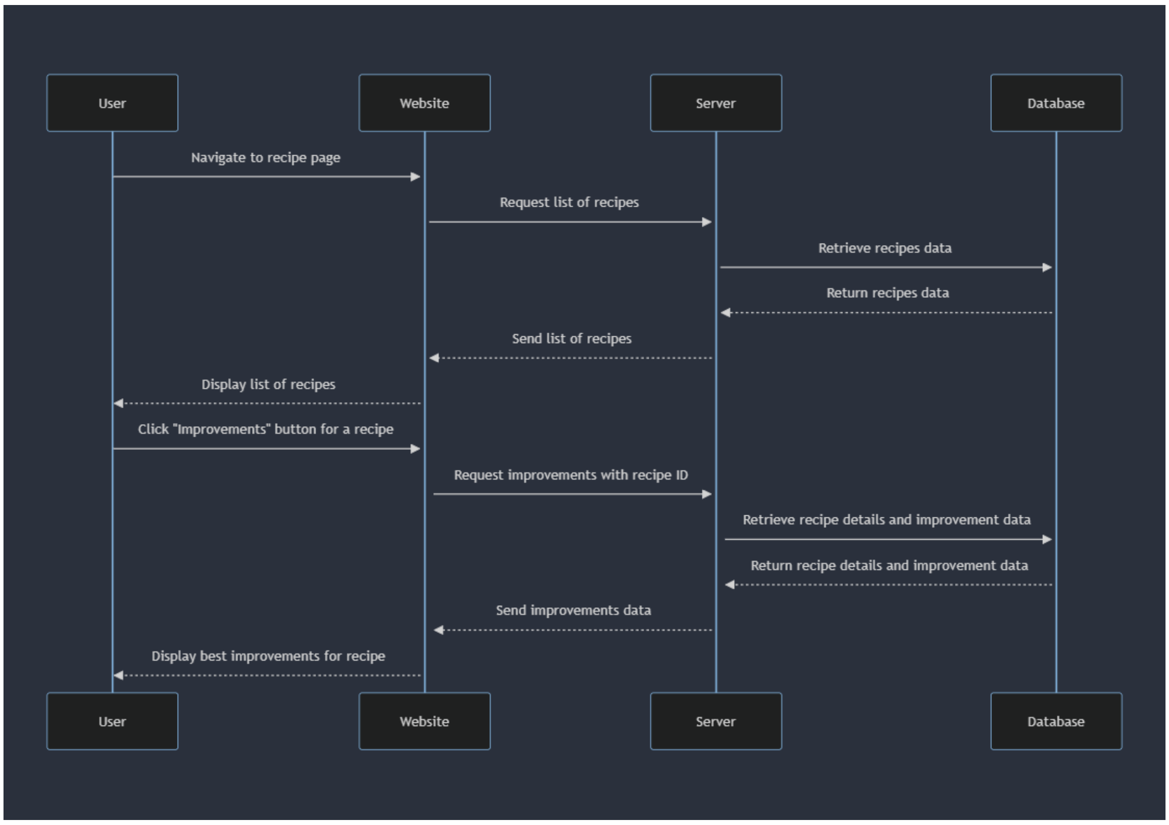

User
Website
Server
Database
Navigate to recipe page
Request list of recipes
Retrieve recipes data
Return recipes data
Send list of recipes
Display list of recipes
Click "Improvements" button for a recipe
Request improvements with recipe ID
Retrieve recipe details and improvement data
Return recipe details and improvement data
Send improvements data
Display best improvements for recipe

Figure 4.7: Improvements Sequence Diagram

that we are working with and the relationships between them. From a structural point of view, our application involves four types of objects: Users, Recipes, Likes and Improvements. The user and all their credentials are saved in the AspNetUsers table along with some other fields generated by the library used for implementing user creation. The recipes are stored in the Recipes table, which contains information about the name of the recipe, cooking time, course, cuisine, the list of ingredients and a link to ”Yummly” based on the of the recipe’s name. The likes of a User are stored in the UserLikes table, saving information about the user who liked a recipe and that specific recipe through foreign keys. The Improvements is directly related to the Recipes table because in this table are stored the improvements of each recipe. Information such as foreign key to the respective recipe, the ingredient that improves the recipe, its type and the degree of improvement are stored within.

From the relational perspective, the UserLikes table works as an intermediary in the many to many relationship between AspNetUsers and Recipes. The n:n relationship is split into two 1:n relationships, each user and recipe being related to multiple likes. There also exists a 1 to many relationship between Recipes and Improvements, a Recipe having multiple improvements.

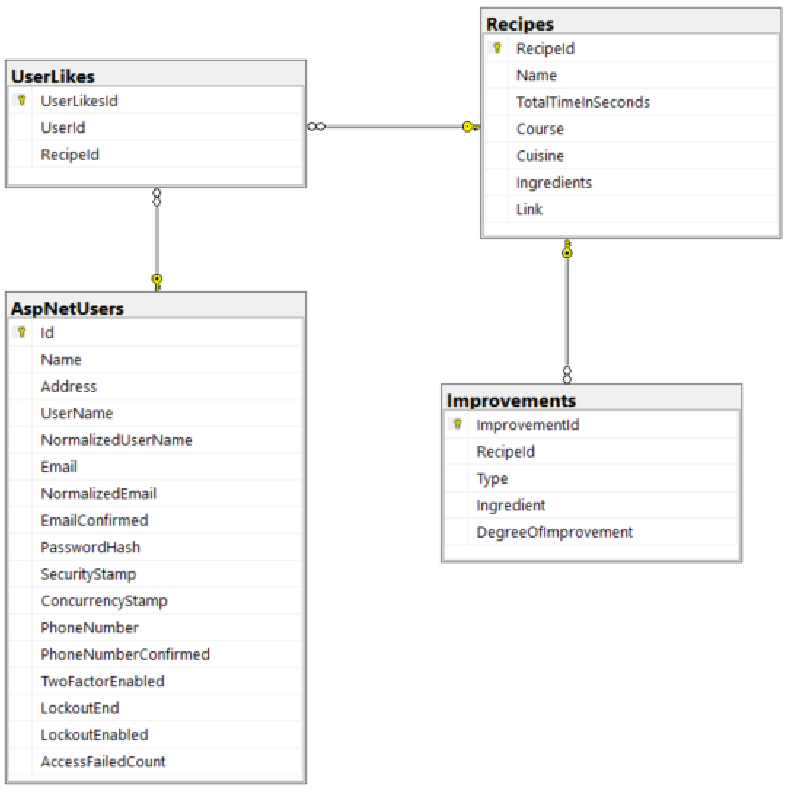


Figure 4.8: Database Diagram

4.2 Implementation

In the following subsections, we will provide an in-depth analysis of the various technologies utilized in the development of our application. We will also talk about some of he most challenging features and how we solved them. The implementation phase is crucial as it represents the foundation for the functionality and performance of the application. Understanding the technologies and methodologies used can offer powerful insights into the development process.

4.2.1 Technologies used

In this subsection, we will discuss all the technologies that played an important role in the creation of our application. Each technology selected based on specific criteria, including performance, scalabitlity, ease of integration and the author's level of knowledge. The following paragraphs will cover each technology in detail, explaining its purpose, advantages and how it was integrated into the application.

ASP.NET MVC

The flavor pairing recommender was developed using ASP.NET MVC (Model-View-Controller) which is a web application framework developed by Microsoft, which implements the MVC design pattern. It's part of the ASP.NET framework, which is a component of the .NET framework. We chose to use ASP.NET MVC to develop the application because it is a lightweight framework, allowing developers to build scalable, maintainable and testable web applications.

Using this approach of the .Net framework, MVC, we are not constrained to use a separate frontend framework as in the case of a REST API. Instead, we can develop both the frontend and the backend within our framework using models and controllers for the backend and views for the frontend.

MVC Design Pattern

Design patterns are basically design tools to improve existing code. They are highly dependent on the aspects of OOP because its principles of encapsulation, polymorphism, abstraction and inheritance extend their properties into patterns-based coding.[L+06]

As the name of the framework that we are using says, we will develop our application within the boundaries of the MVC design pattern. The main idea behind this design pattern is that it separates the application into three big components: Models, Controllers and Views, each one of this being responsible for a specific aspect of the application's functionality. The flow of the framework can be seen in the Fig. 4.9.

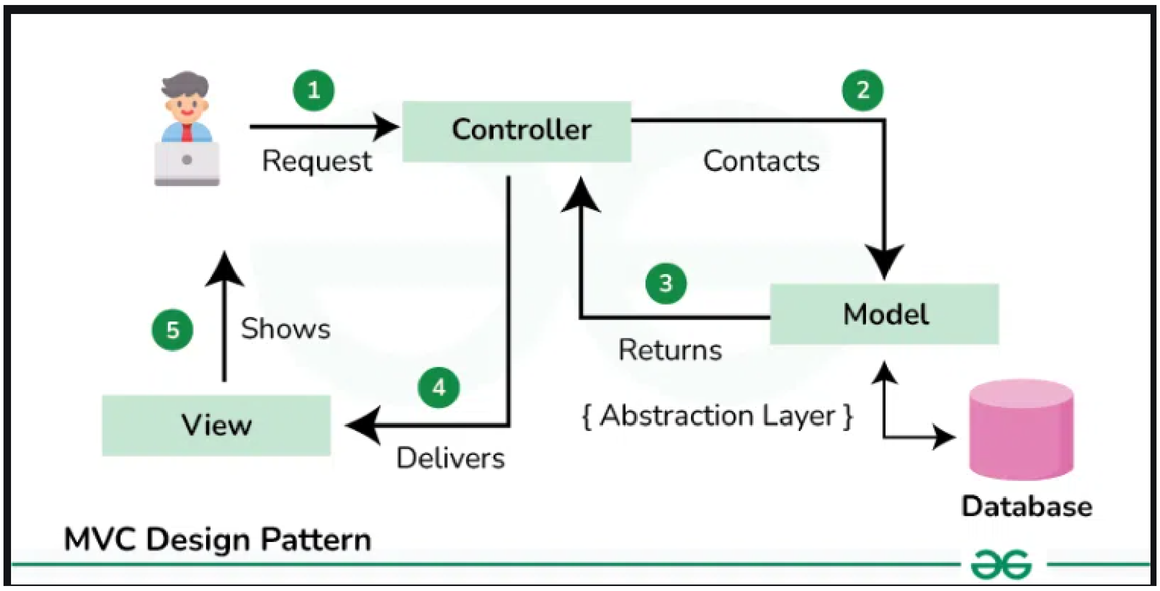


Figure 4.9: MVC Design Pattern flow. [Gee23]

When users are interacting with our application, they are performing requests to the controller which uses the information from the Model that's linked to a Database to process it and give it to the View in order to show it to the user.

Most web applications are split into two parts when talking about implementation: backend and frontend or client and server. In the next paragraphs we will talk about how this two parts were developed.

Backend

The backend or server is related to the business layer of our application and acts like a middle man between Frontend and the Database. In order to develop our application we actually need two servers, the original ASP.NET server which runs our web application and a Python REST API.

The ASP.NET server uses the C# programming language and lies within the .NET framework. This server acts as the foundation of our application, handling all the business business logic and the majority of database interactions.

The REST API uses the Python programming language and the Flask framework. Even thought ASP.NET provides a lot of useful libraries that you can work with, it is not as specialized as Python in the field of Machine Learning. The Python server was additionally created to perform computations on data using libraries like pandas, numpy and sklearn.

Frontend

The frontend or client is related to the presentation layer of our application. It is what the user interacts with within the application. In our case, the development of the frontend is done using the Razor views of the ASP.NET MVC framework. The main languages used are HTML for structure and content, CSS for presentation and styling and JavaScript for interactivity and behavior. Shortly said, the HTML puts the elements on your interface, the CSS makes them look prettier and the JavaScript gives them functionalities and interactivity. Razor views are powerful tools that allow you embed C# code within HTML, CSS and JavaScript code. Additionally, libraries like Bootstrap and jQuery were used to enhance the frontend development process. Bootstrap is a popular CSS library that provides pre-designed components, which helps you build application faster and that look better. jQuery is a JavaScript library which helps u use JavaScript on your website. It simplifies a lot of things like AJAX calls and DOM manipulation.

Database

The database is related to the data access layer of our application. Most of our application work using data. Well that data has to be stored somewhere easy to access within our application. In our case we used a SQL(Structured query language) database. A SQL database is a relational database which uses tables and relations between entities. Out of all the SQL databases we used MSSQL, Microsoft SQL Server because it fits perfectly within our needs. Being also created by Microsoft, we could say that the .NET framework has an affinity for this type of databases. It also provide advanced security features having authentication integration, high performance and scalability.

4.2.2 Details of implementation

In this subsection we will talk about some features that introduce some of the used libraries and tools along with the implementation of the most challenging features.

Mapping Models to Database

The connection between our application and our database is realised through an ORM(Object Relational Mapping). Object relational mapping represents a mechanism to bridge the semantic gap between object oriented programming languages and a relational database management system. [LHR16]

In the context of our application, we will use Entity Framework, an open-source ORM framework for .Net applications. It allows developers to work with the database using .Net objects, eliminating the need for most of the data-access code. In our application the mapping between the objects and database is done using IdentityDbContext, DataAnnotations and Entity Framework through migrations. A migration refers to the process of evolving the data model and database schema in response to changes in application requirements. This involves operations such as adding or modifying associations, renaming entities, adding, removing or updating entities. [VWV11] Basically migrations contain versions of your database. In Entity Framework you can change this versions using commands like Add-Migration for creating a new one, Remove-Migration for deleting one and Update-Database which applies the modifications from your Models to the database.

User Authentication

All the processes related to the User accounts is built using the ASP.NET Identity API. It provides a solution for managing user authentication and authorization and includes features for user registration, password recovery and account confirmation, multi-factor authentication and external login providers such as Google, Microsoft.

Within our application, we use ASP.NET Identity by inheriting our AppUser class from ”IdentityUser”, adding two additional fields, name and address and also by using the IdentityDbContext which is the base class for the Entity Framework database context used for Identity. By using those, several tables with information about users, roles, claims and tokens will be created in our database.(See Fig. 4.10)

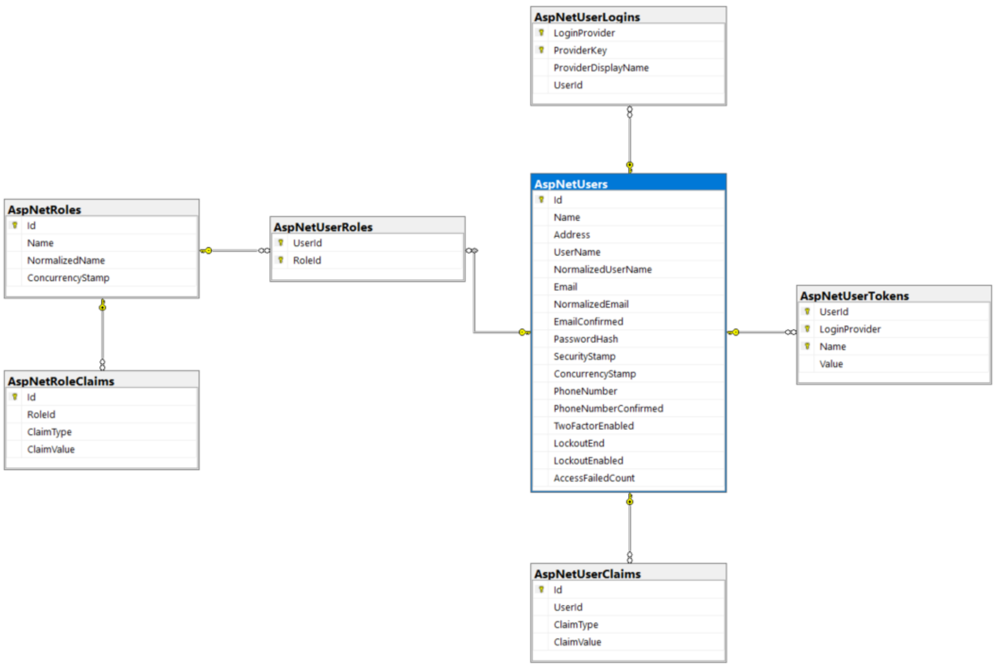


Figure 4.10: Identity tables and relations.

From a user role perspective, Identity provides various possibilities, but we kept it simple by not imposing any specific roles. All users are considered standard users and have the same access to application features. At this point, action related to the admin role can be performed directly through the database, but this aspect can be seen as a future work perspective.

Features like Registration, Login and Logout are used within the Identity API. Users can register using a Name, Email, a simple password that does not require capital letters or special characters with a minimum length of eight characters. The simplicity of the password is due to testing purposes, but it could easily be improved using the AddIdentity withing the Identity API. Users are also asked for the Password confirmation and for their addresses. The login is done using the right combinations of email and password and the logout allows you to exit the current session.

User liking system

User interactions, such as likes on recipes, are stored in the database. This allows the system to personalize recommendations and improve the user experience based on individual preferences.

The functionality uses the UserLikes table within the database to save the information. When a user clicks on the like button from the right corner of each recipe an ajax call is sent to the UserLikeController to save the information to the database. There are two cases for this functionality:

1. **Like**: When a user sees a recipe that isn't already liked, he can click on the empty heart triggering an ajax call to the ”addRecipeToUserProfile” function within the UserLikeController also sending the id of the liked recipe. The function gets the id of the current user using the UserManager API and the id of the liked recipes and creates a new UserLike object that will be persisted in our database.

2. **Unlike**: Quiet the opposite with the ”Like” functionality, when users click on the full hearth an ajax call to the ”removeRecipeFromUserProfile” function is triggered. The same retrievals are performed, but in this case, the like is removed from the database.

The Likes are stored in the database because when we render the list of recipes we also have to render the likes so that the user can see the likes from past sessions.

Recommendation system based on likes

To enhance user experience, we implemented a recommendation system based on user likes. The system uses the user interactions with the application to suggest similar items they might like.

This problem lies within the Machine learning field and in order to solve the problem we used some predefined Python libraries like ”pandas”, ”numpy” and ”sklearn”. This implementation imposes the existence of a Python REST API that will perform the computations for us.

The recommendations are presented to the user on the Recommendations Page. When the User tries to see his recommendations, the ASP.NET server sends a request to the Python REST API with the id of the current User in a JSON format. The REST API decrypts the user and retrieves the necessary data from the database. Both of the servers are connected to the same database so that they have access to the same data.

Now that we have the data we can use the libraries to perform some recommendations. In order to do that we'll create for each one of our recipe a ”soup” which is basically just a way of saying that we will take all the attributes that we are interested in and join them within a string. Our actual problem lies into the field of Natural Language Processing because we are trying to get recipes that have similar ″soups″. In our ″soup″ we will keep information about the name, the course, the cuisine and the ingredients of a recipe.

Having that soup we will use a tool from the scikit-learn library name CountVectorizer. Using the fit_transform function of the CountVectorizer on the ″soup″ column, we will create a count matrix, which is a sparse matrix representing the frequency of each word in the text data. Each row of the matrix corresponds to a recipe and each column corresponds to a unique word. The values in the matrix represent the number of times each word appears in each recipe. Another alternative would've been to use the TF-IDFVectorizer, but in our case Course and Cuisine are very likely to repeat and we do not want to down-weight them if there are multiple recipes from the same Cuisine or Course.

Using the sparse matrix created using the CountVectorizer, we will use the cosine similarity 4.1 to see how similar our recipes are with each other. Another matrix with similarity scores is created using the cosine_similarity function from sklearn library. The cosine similarity is a measure of similarity between two non-zero vectors that measures the cosine of the angle between them. It ranges from -1 to 1 where 1 means that the vectors are identical, 0 means the vectors are orthogonal and -1 means that vectors are diametrically opposed. The new matrix is a square matrix where each element represents the cosine similarity between two rows in the count matrix.

$$Cosine(x, y) = \frac{x \cdot y}{|x||y|} \tag{4.1}$$

In order to perform our recommendations, we will use the cosine similarity matrix. For each recipe that the user liked, we will retrieve the top five recipes with the biggest scores and those will be the recommendations for that recipe. Considering that some users might like multiple recipes and some users might like a small number of recipes, close to none, we are going to set the number of recommendations shown at a time to 10 and we'll impose the next conditions for our how we chose our recommendations:

1. If the user is at the beginning and he had no time to use our application and as a result he didn't like any of the recipes, 10 random recipes from the database will be provided for him.

2. If the user liked recipes, but not an enough number(e.g. if he liked only one recipe because it only gives five recommendations, or the special case in which

he liked two recipes, but their sets of recommendations are not disjunct) we will complete the list of recommendations with random recipes until 10.

3. If in the user's list of recommendations there are exactly 10 recipes, the user will see exactly those recipes.
4. If the user's list contains more than 10 recipes, 10 of those will be filtered out and shown to the user.

Recipe Improvements

The "Recipe Improvements" is the most important feature of our application. From the implementation point of view, similar to the recommendation system based on user likes, we used Python libraries to work with our data and to perform complex computations. The difference is that instead of performing real time request between the two servers, we only used Python server to perform the computations for the Improvements which were stored in the database from performance reason. With this approach, the algorithm is ran only once and for all the recipes, not everytime you want to access the information about the improvements of a recipe.

The computations are pretty straightforward, we retrieve the data from the related csv's and we create a matrix using pivot_table where at the position i,j will be the number of shared flavor compounds between ingredient i and ingredient j. Afterwards we create the cosine similarities matrix by applying the cosine_similarity function to the scaled matrix. Now, at the position i,j we will have the similarity score between ingredient i and ingredient j. Using this matrix we compute the FCS for the additional ingredients and select the best recommendations based on the nine previously mentioned types. All the information will be stored in the Improvement table providing information about the Id of the recipe because we need to know what recipe we are improving, the ingredient, its type and the value of FCS.

On the interface, each recipe has a separate page on improvements that can be accessed by clicking on the Improvements button of the corresponding recipe. Each page containing charts presenting the best improvements ranked based on different filters.

Verification and Validation

Verification and validation are critical components of the software development process, ensuring that the application functions correctly and meets the user's needs.

Validation is particularly important as it helps guide the user through the application, making the experience as likable and error-free as possible.

In our application, it is relatively difficult for users to make mistakes that require extensive validation. However, during the registration and login processes, validation becomes essential. For example, when users register, they must provide a valid email address, a password that meets the required criteria, a confirmation password that's identical with the initial password and other useful details. If any information is incorrect or incomplete, the system alerts the user with appropriate messages, guiding them to correct the errors. Similarly, during login, users are notified if their credentials do not match any existing accounts, alerting them to re-enter the correct information.

4.3 UI/UX

In this section we will provide information about the user interface and the user experience using a series of screenshots from the application. The screenshots have the role to illustrate how different features look in the eyes of a user.

4.3.1 Login page

The login page 4.11 of the application is designed with simplicity and a user-friendly interface. The page presents a clean layout with the application logo at the top left, followed by options to "Register" or "Login" at the top right. The main content is centered and includes fields like "Username" and "Password", along with a "Remember Me" checkbox. Below the login form, there is a link for users who do not have an account, encouraging them to register. The login button is clearly visible in turquoise, ensuring users can easily log in to access their accounts.

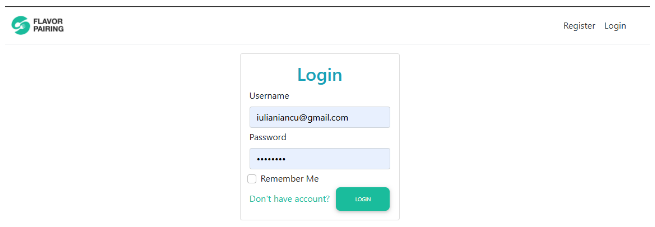
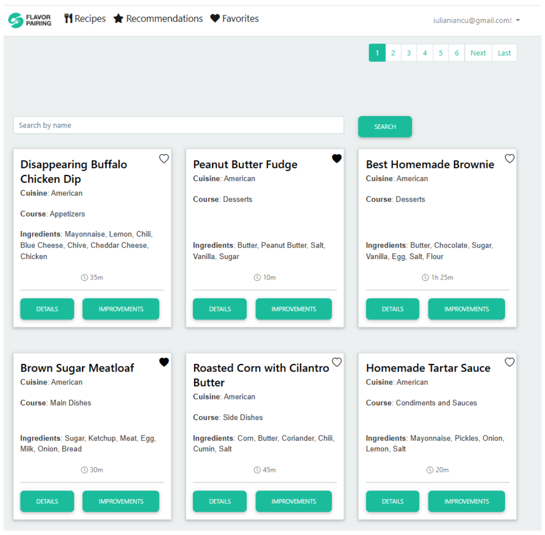


Figure 4.11: SS from application with Login page.

4.3.2 Recipes page

The recipes page 4.12 presents a collection of various recipes available within our database. Each recipe is displayed in a card format, providing a snapshot of key details such as the dish name, cuisine, course, ingredients and cooking time. Each card has two buttons, "Details" and "Improvements", allowing users to view more advanced details or to explore the list of available improvements. There is also a like/unlike button shaped as a hearth in the right corner of each card. The layout is designed to be intuitive, making it easy for users to browse and find recipes that interest them by using the search and the pagination.

The favorites page looks similar, the only difference being the content displayed. If here we present the recipes, on the favorites page there will be shown only the recipes of interest for the user.



Disappearing Buffalo Chicken Dip
Cuisine: American
Course: Appetizers
Ingredients: Mayonnaise, Lemon, Chili, Blue Cheese, Chive, Cheddar Cheese, Chicken
35m

Peanut Butter Fudge
Cuisine: American
Course: Desserts
Ingredients: Butter, Peanut Butter, Salt, Vanilla, Sugar
10m

Best Homemade Brownie
Cuisine: American
Course: Desserts
Ingredients: Butter, Chocolate, Sugar, Vanilla, Egg, Salt, Flour
1h 25m

Brown Sugar Meatloaf
Cuisine: American
Course: Main Dishes
Ingredients: Sugar, Ketchup, Meat, Egg, Milk, Onion, Bread
30m

Roasted Corn with Cilantro Butter
Cuisine: American
Course: Side Dishes
Ingredients: Corn, Butter, Coriander, Chili, Cumin, Salt
45m

Homemade Tartar Sauce
Cuisine: American
Course: Condiments and Sauces
Ingredients: Mayonnaise, Pickles, Onion, Lemon, Salt
20m

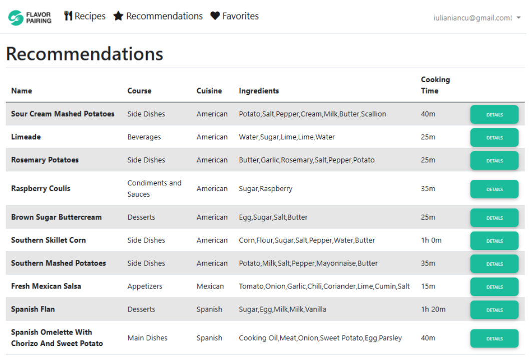
Figure 4.12: SS from application with the page containing the recipes.

4.3.3 Recommendations page

The recommendations page 4.13 presents a list of suggested recipes based on the user's preferences. The recommendations are displayed in a table with columns for the recipe name, course, cuisine, ingredients, and cooking time. This structures approach allows users to quickly compare different recommended recipes and select the ones they like. Each row also includes a "Details" button for users to access the complete recipe.

4.3.4 Improvements page

The improvements page provides detailed suggestions on how to enhance specific recipes. This page includes an analysis with a total of ten charts: one chart presents the overall best improvements, while the remaining nine charts present the



Recommendations

Name	Course	Cuisine	Ingredients	Cooking Time
Sour Cream Mashed Potatoes	Side Dishes	American	Potato,Salt,Pepper,Cream,Milk,Butter,Scallion	40m
Limeade	Beverages	American	Water,Sugar,Lime,Lime,Water	25m
Rosemary Potatoes	Side Dishes	American	Butter,Garlic,Rosemary,Salt,Pepper,Potato	25m
Raspberry Coulis	Condiments and Sauces	American	Sugar,Raspberry	35m
Brown Sugar Buttercream	Desserts	American	Egg,Sugar,Salt,Butter	25m
Southern Skillet Corn	Side Dishes	American	Corn,Flour,Sugar,Salt,Pepper,Water,Butter	1h 0m
Southern Mashed Potatoes	Side Dishes	American	Potato,Milk,Salt,Pepper,Mayonnaise,Butter	35m
Fresh Mexican Salsa	Appetizers	Mexican	Tomato,Onion,Garlic,Chili,Coriander,Lime,Cumin,Salt	15m
Spanish Flan	Desserts	Spanish	Sugar,Egg,Milk,Milk,Vanilla	1h 20m
Spanish Omelette With Chorizo And Sweet Potato	Main Dishes	Spanish	Cooking Oil,Meat,Onion,Sweet Potato,Egg,Parsley	40m

Figure 4.13: SS from application with the page containing the recommendations.

improvements by different types of ingredients that are suggested as an addition to the recipe.

The top section displays the overall best improvements for the "Disappearing Buffalo Chicken Dip", illustrated with a bar chart that compares different ingredients such as rice, beans and potato, which represent the best overall improvements based on their score. This provides a clear visual representation of which ingredients improve the most our recipe.

The lower section displays multiple charts similar to the first one the difference is represented by the analysed data. Instead of considering all the possible ingredients when performing the score for the improvements, we compute them separately based on the type of the ingredient and show the result in separate charts.

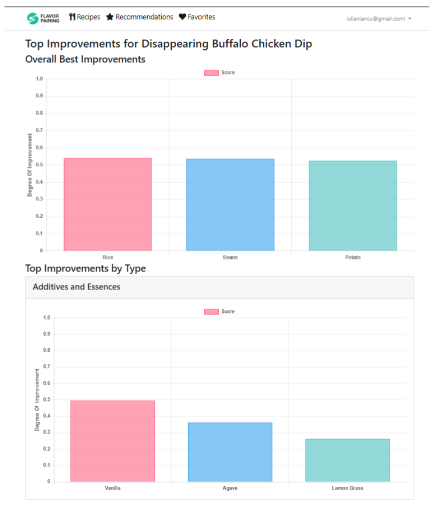


Figure 4.14: SS from application with the improvements of a recipe.

With these expressive images, we conclude the "Software Application" chapter. We have detailed the architecture, implementation and user interface of our application, providing a detailed overview of its development and functionality. The integration of various technologies and decision behind the major design choices were highlighted, illustrating the processes that brought our application to life.

Chapter 5

Conclusions

In conclusion, this paper has the purpose of finding an automated way of improving recipes based on the chemical compounds that give flavor to the ingredients. This search has been done using different metrics like arithmetical mean, standard deviation and cosine similarity, each of these metrics offering a unique approach to quantifying flavor compatibility, allowing for a comprehensive analysis of how ingredients interact at a chemical level.

Being an exploratory process, we did not know what results to expect and unfortunately, we could not determine the accuracy and correctness of our approaches only by analyzing some statistics. Practical testing is essential to validate our findings, but it would require significant resources and time to perform such experimentation. Without the practical testing, we can't be certain of the correctness of our results.

Future work could involve a more detailed implementation using advanced AI models to identify patterns within our ingredient relations. Another approach that would work would be to incorporate additional factors into the computations, such as the quantity of an ingredient, its nutritional values, cooking steps of the recipe and other elements that may influence the final result.

In summary, while this research presents a solid foundation for automated recipe improvements using flavor compounds, its true potential can only be realized through practical testing and further development. Adding AI models and additional factors may enhance the accuracy and applicability of our proposed approaches, making it a valuable tool for both scientific analysis and practical culinary use.

Bibliography

[A$^+$12] Fahad Alhumaidan et al. A critical analysis and treatment of important uml diagrams enhancing modeling power. *Intelligent Information Management*, 4(05):231, 2012.

[AABB11] Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. Flavor network and the principles of food pairing. *Scientific reports*, 1(1):196, 2011.

[AK17] Babak Joze Abbaschian and Samira Khorshidi. A review of hybrid recommender systems. *Ad Alta: Journal of Interdisciplinary Research*, 7(2), 2017.

[AS$^+$22] Ghada SR Al Saqqa et al. What to know about food flavor? a review. *Jordan Journal of Agricultural Sciences*, 18(1):1–15, 2022.

[AVK$^+$18] Md Fazle Alam, Supriya Varshney, Masood Alam Khan, Amaj Ahmed Laskar, and Hina Younus. In vitro dna binding studies of therapeutic and prophylactic drug citral. *International journal of biological macromolecules*, 113:300–308, 2018.

[BDD$^+$12] Cristian Bancu, Monica Dagadita, Mihai Dascalu, Ciprian Dobre, Stefan Trausan-Matu, and Adina Magda Florea. Arsys–article recommender system. In *2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 349–355. IEEE, 2012.

[BFG11] Robin Burke, Alexander Felfernig, and Mehmet Göker. Recommender systems: An overview. *Ai Magazine*, 32:13–18, 09 2011.

[Blu08] Heston Blumenthal. *The big fat duck cookbook*. Bloomsbury, 2008.

[DP06] Brian Dobing and Jeffrey Parsons. How uml is used. *Commun. ACM*, 49:109–113, 05 2006.

[FDB$^+$15] Irlan Almeida Freires, Carina Denny, Bruna Benso, Severino Matias de Alencar, and Pedro Luiz Rosalen. Antibacterial activity of essential oils and their isolated constituents against cariogenic bacteria: a systematic review. *Molecules*, 20(4):7329–7358, 2015.

[GB22] Mansi Goel and Ganesh Bagler. Computational gastronomy: A data science approach to food. *Journal of Biosciences*, 47(1):12, 2022.

[GCM+22] Mogan Gim, Donghee Choi, Kana Maruyama, Jihun Choi, Hajung Kim, Donghyeon Park, and Jaewoo Kang. Recipemind: Guiding ingredient choices from food pairing to recipe completion using cascaded set transformer. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pages 3092–3102, 2022.

[Gee23] GeeksforGeeks. Mvc design pattern, 2023. Accessed: 2024-06-06.

[GPS+21] Mogan Gim, Donghyeon Park, Michael Spranger, Kana Maruyama, and Jaewoo Kang. Recipebowl: A cooking recommender for ingredients and recipes using set transformer. *IEEE Access*, 9:143623–143633, 2021.

[GST+18] Neelansh Garg, Apuroop Sethupathy, Rudraksh Tuwani, Rakhi Nk, Shubham Dokania, Arvind Iyer, Ayushi Gupta, Shubhra Agrawal, Navjot Singh, Shubham Shukla, et al. Flavordb: a database of flavor molecules. *Nucleic acids research*, 46(D1):D1210–D1216, 2018.

[Gul20] Oshin Gulsia. Vanillin: One drug, many cures * oshin gulsia. *Resonance*, 25, 07 2020.

[JB+15] Anupam Jain, Ganesh Bagler, et al. Spices form the basis of food pairing in indian cuisine. *arXiv preprint arXiv:1502.03815*, 2015.

[KGB22] Reshna KR, Sreerag Gopi, and Preetha Balakrishnan. Introduction to flavor and fragrance in food processing. In *Flavors and Fragrances in Food Processing: Preparation and Characterization Methods*, pages 1–19. ACS Publications, 2022.

[L+06] Christopher G Lasater et al. *Design patterns*. Jones & Bartlett Publishers, 2006.

[LHR16] Martin Lorenz, Guenter Hesse, and Jan-Peer Rudolph. Object-relational mapping revised-a guideline review and consolidation. In *ICSOFT-EA*, pages 157–168, 2016.

[Lin] Lingcheng. Flavor network. `https://github.com/lingcheng99/Flavor-Network`. GitHub repository; accessed 22 May 2024.

[MM23] Vusumuzi Maphosa and Mfowabo Maphosa. Fifteen years of recommender systems research in higher education: Current trends and future direction. *Applied Artificial Intelligence*, 37(1):2175106, 2023.

[PKK$^+$21] Donghyeon Park, Keonwoo Kim, Seoyoon Kim, Michael Spranger, and Jaewoo Kang. Flavorgraph: a large-scale food-chemical graph for generating food representations and recommending food pairings. *Scientific reports*, 11(1):931, 2021.

[PKP$^+$19] Donghyeon Park, Keonwoo Kim, Yonggyu Park, Jungwoon Shin, and Jaewoo Kang. Kitchenette: Predicting and recommending food ingredient pairings using siamese neural networks. *arXiv preprint arXiv:1905.07261*, 2019.

[RD22] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59, 2022.

[TLA12] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. Recipe recommendation using ingredient networks. In *Proceedings of the 4th annual ACM web science conference*, pages 298–307, 2012.

[Tri23] Neha Tripathi. Computational gastronomy. 2023.

[VWV11] Sander Daniël Vermolen, Guido Wachsmuth, and Eelco Visser. Generating database migrations for evolving web applications. In *Proceedings of the 10th ACM international conference on Generative programming and component engineering*, pages 83–92, 2011.

[Yao] Conway Yao. Recipe-analysis. `https://github.com/conwayyao/Recipe-Analysis`. GitHub repository; accessed 22 May 2024.

[ZPY$^+$23] Zitong Zhang, Braja Gopal Patra, Ashraf Yaseen, Jie Zhu, Rachit Sabharwal, Kirk Roberts, Tru Cao, and Hulin Wu. Scholarly recommendation systems: a literature survey. *Knowledge and Information Systems*, 65(11):4433–4478, 2023.