UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DEVELOPMENT OF A WEB APPLICATION WITH AN INTEGRATED RECOMMENDATION SYSTEM FOR RECIPES

Diploma Thesis

Evangelia-Alkistis Lemonaki

Kleopatra Beka

Supervisor: Michael Vassilakopoulos

September 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DEVELOPMENT OF A WEB APPLICATION WITH AN INTEGRATED RECOMMENDATION SYSTEM FOR RECIPES

Diploma Thesis

Evangelia-Alkistis Lemonaki

Kleopatra Beka

Supervisor: Michael Vassilakopoulos

September 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΠΑΓΚΟΣΜΙΟΥ ΙΣΤΟΥ ΜΕ ΕΝΣΩΜΑΤΩΜΕΝΟ ΣΥΣΤΗΜΑ ΣΥΣΤΑΣΕΩΝ ΓΙΑ ΣΥΝΤΑΓΕΣ

Διπλωματική Εργασία

Ευαγγελία-Άλκηστις Λεμονάκη

Κλεοπάτρα Μπέκα

Επιβλέπων: Μιχαήλ Βασιλακόπουλος

Σεπτέμβριος 2022

Approved by the Examination Committee:

Supervisor	**Michael Vassilakopoulos** Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Athanasios Fevgas** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly
Member	**George Thanos** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

We want to express our gratitude to both our friends and family for encouraging us to continue with the pursuit of our dreams and providing us with unconditional support during this journey.

Furthermore, we thank Prof. Michael Vasilakopoulos for supervising this diploma thesis. Moreover, I would like to thank Lab. Teaching Staff Athanasios Fevgas and Lab. Teaching Staff George Thanos for their evaluation and contribution to our thesis.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarants

Evangelia-Alkistis Lemonaki and Kleopatra Beka

Diploma Thesis

DEVELOPMENT OF A WEB APPLICATION

WITH AN INTEGRATED

RECOMMENDATION SYSTEM

FOR RECIPES

Evangelia-Alkistis Lemonaki

Kleopatra Beka

Abstract

Due to the rapid technological development in recent years, an increasingly more significant percentage of the population uses the internet for convenience in everyday activities. One of these activities includes the process of learning and storing recipes. Whereas in the old days, a person had to manually write a recipe on a piece of paper (or a notebook) and keep them all stored together to stay organised, nowadays we only have to look online for any recipe of our choosing. They have a list of ingredients at their disposal and a step-by-step guide. Websites have integrated recommendation systems inside to help make this experience even more personalised to the user. Depending on the selected type (content-based, collaborative filtering, knowledge-based or hybrid), these systems provide content recommendations to the user using their implicit and or explicit feedback to generate them. In this thesis, we present the background of recommendation systems and the approach we took to develop both the recommendation systems themselves and the website that supports them, in detail. A dataset that includes fifteen thousand recipes and one hundred and eleven thousand reviews on these recipes was used for the development of the recommendation systems. For our purposes, we created three recommendation systems: two content-based and one with collaborative filtering. The first system finds similar recipes based on nutritional values related to the recipe of interest to the user. The second system, using cosine similarity, finds recipes similar to the specific recipe of interest to the user based on ingredients, title, keywords and category. Finally, the third system finds users with similar interests (in terms of recipes) and suggests other recipes. We calculate the similarity in terms of interests using the scores users gave to the recipes they evaluated.

Keywords:

Recommendation system, content-based, collaborative filtering,, knowledge-based, hybrid, recipes, website, Django, Python

Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΠΑΓΚΟΣΜΙΟΥ ΙΣΤΟΥ ΜΕ ΕΝΣΩΜΑΤΩΜΕΝΟ ΣΥΣΤΗΜΑ ΣΥΣΤΑΣΕΩΝ ΓΙΑ ΣΥΝΤΑΓΕΣ

Ευαγγελία-Άλκηστις Λεμονάκη

Κλεοπάτρα Μπέκα

Περίληψη

Λόγω της ραγδαίας ανάπτυξης της τεχνολογίας τα τελευταία χρόνια, ένα ολοένα και μεγαλύτερο ποσοστό του πληθυσμού χρησιμοποιεί το διαδίκτυο για την διευκόλυνσή του στις καθημερινές του δραστηριότητες. Μία από αυτές τις δραστηριότητες περιλαμβάνει τη διαδικασία εκμάθησης και αποθήκευσης συνταγών. Ενώ παλαιότερα, ένα άτομο έπρεπε να γράψει χειροκίνητα μια συνταγή σε ένα κομμάτι χαρτί (ή ένα σημειωματάριο) και να τις κρατήσει όλες μαζί για να παραμείνουν οργανωμένοι, σήμερα δεν έχουμε παρά να ψάξουμε στο Διαδίκτυο για οποιαδήποτε συνταγή της επιλογής μας και έχουμε στη διάθεσή μας τη λίστα των συστατικών και έναν οδηγό βήμα προς βήμα. Για να γίνει αυτή η εμπειρία ακόμα πιο εξατομικευμένη στους χρήστες, οι ιστότοποι έχουν ενσωματωμένα συστήματα συστάσεων. Αυτά τα συστήματα, ανάλογα με τον επιλεγμένο τύπο (βασισμένο στο περιεχόμενο, συνεργατικού φιλτραρίσματος, βασισμένο στη γνώση ή υβριδικό), παρέχουν προτάσεις περιεχομένου στον χρήστη χρησιμοποιώντας την έμμεση ή και την ρητή τους ανατροφορότηση για τη δημιουργία τους. Σε αυτή τη διατριβή, παρουσιάζουμε λεπτομερώς το ιστορικό των συστημάτων συστάσεων μαζί με την προσέγγιση που ακολουθήσαμε για την ανάπτυξη τόσο των ίδιων των συστημάτων συστάσεων όσο και του ιστότοπου που τα υποστηρίζει. Για την ανάπτυξη των συστημάτων χρησιμοποιήθηκε ενα σύνολο δεδομένων που περιλαμβάνει δεκαπέντε χιλιάδες συνταγές και εκατόν έντεκα χιλιάδες κριτικές πάνω στις συνταγές αυτές. Έχουν δημιουργηθεί τρία συστήματα συστάσεων: δυο βασισμένα στο περιεχόμενο και ενα συνεργατικού φιλτραρίσματος. Το πρώτο σύστημα βρίσκει παρόμοιες συνταγές με βάση τις διατροφικές αξίες, σχετικά με την συνταγή που ενδιαφέρει τον χρήστη. Το δεύτερο σύστημα, χρησιμοποιόντας ομοιότητα συνημιτόνου, βρίσκει συνταγές παρόμοιες με την συγκεκριμένη συνταγή που ενδιαφέρει τον χρήστη, ανάλογα με τα υλικά, τον τίτλο, τις λέξεις κλειδιά και την κατηγορία.

Τέλος, το τρίτο σύστημα βρίσκει χρήστες με παρόμοια ενδιαφέροντα (ως προς τις συνταγές), και προτείνει άλλες συνταγές. Η ομοιότητα ως προς τα ενδιαφέροντα υπολογίζεται απο τις βαθμολογίες που έχουν δώσει οι χρήστες στις συνταγές που έχουν αξιολογήσει.

Λέξεις-κλειδιά:

Συστήματα συστάσεων, βασισμένο στο περιεχόμενο, συνεργατικού φιλτραρίσματος, βασισμένο στη γνώση, υβριδικό, συνταγές, ιστότοπος, ιστοσελίδα

Table of contents

List of figures

List of tables

Abbreviations

e.g.	for example
etc.	Et cetera
aka	Also known as
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
EDA	Exploratory Data Analysis
KNN	K-Nearest Neighbor
NLP	Natural Language Process
SGD	Stochastic Gradient Descent
TF-IDF	Term frequency - Inverse document frequency
GUI	Graphical User Interface

Chapter 1

Introduction

In recent years, technological advancements have aided the market of readily available websites for solving the everyday needs of their users. One such demand is cooking. As a result, recipe websites make an increasing appearance on the web, overcrowding the space with exorbitant amounts of data. Consequently, users may become overwhelmed by the sheer volume and grow unable to retrieve the information they were searching for in the first place. To solve this problem, we have developed recommendation systems whose sole purpose is to figure out the user's preferences and remove the menial and time-consuming task of finding content to their liking. As recipe sharing through the internet becomes increasingly popular, the demand for such a solution as integrated recommendation systems becomes a necessity that every user-based website cannot do without.

For the systems to do their job correctly, they need to process a large amount of data (usually, they need at least 10000 items). For our case, we used data from two CSV files containing around 500,000 recipes from more than 300 different categories and almost 1,500,000 reviews. Food.com is a popular website for sharing recipes and reviews with users from all around the world. The recipe's data (for example, calories, fat content and other nutritional values) are solely provided by the recipe author.

1.1 Recipe Recommendation

This diploma thesis aims to solve the problem mentioned above that users experience nowadays. Notably, the way to solve the obstacle of overwhelming the user with information is by creating personalised recommendation systems. The recommendations provided through these systems will be tailored to each user on the website, taking into consideration the implicit and explicit feedback that the user generates while using the site to create them. This technique results in the user having ready-to-go content recommendations and do not have to search endlessly for something they might like in the vast amount of data. There are three recommendation systems in total, two of them are Content-Based and the other one is Collaborative Filtering. After their explanation and implementation, we evaluated them and interpreted the results.

1.2 Contribution

The contribution of this thesis can be summarized as follows::

1. Recommendation systems of various types have been studied in order to discern the ones that fit our needs.
2. A website has been implemented for the purpose of this thesis.
3. The recommendation systems that we have deemed necessary for our demands have been created and integrated with the website.
4. An evaluation on the accuracy of the recommendations has been conducted.

1.3 Bibliographic review

Book [1] is significantly informative and contains all the theories behind recommendation systems and their application. Apart from that, there are numerous examples to understand the concept fully. Furthermore, book [2] is equivalent to the one mentioned before, but it explains in more detail the comparison between different recommendation systems. Article [3] consists of information about Content-Based and Collaborative Filtering systems. The [4] is an article that explains the problems of each type of recommendation system and suggests a hybrid approach. Also, [5], and [6] suggest a hybrid system but the first one also contains general information about hybrid recommendation systems and some of the combinations. In [7], we have the differences between Memory-Based and Model-Based systems. Article [8] is a valuable source of information about all the types of recommendation systems and their subcategories. Also, it mentions the techniques of the systems and their limitations, like the cold start problem, that articles [9] and [10] explain more about it, its subcategories and its solutions. In [11], knowledge-based systems and their components are explained. After that, articles [12] and [13] interpret the types of feedback, implicit and explicit. Apart from explaining the types of feedback, the first one contains some information about predicting values from user feedback with machine learning algorithms and evaluating them. Articles [14], [15], [16], [17] and [18] describe what is evaluation on systems. They explain the basic general methods of evaluations, like novelty and serendipity, and [17] interprets the accuracy metrics.

As far as recommendation systems are concerned, in article [19], we have a recipe recommendation system that uses machine learning algorithms. More specifically, the dataset is created with web scraping from a website called 'yummy.com'. Then Term frequency - Inverse document frequency (TF - IDF) is used along with cosine similarity to compute the scores of the ingredient pairs. TF - IDF is a statistical measure that evaluates how compatible a word is with a document in a group of documents. On the other hand, cosine similarity measures the similarity between two vectors of an inner product space. We used a function similar to TF - IDF, called CountVectorizer, and cosine similarity only in our Content-Based part of the hybrid system we made. Another interesting approach is the one in article [20] that is based on Natural Language Process (NLP) using the nutrition data of the recipes.

In article [21], we have a recommendation system similar to ours, which is a hybrid recommendation system using both Content-Based and Collaborative Filtering approaches. It proposes two hybrid strategies utilising the content of recipes and rating information on recipes. The first approach is with K-Nearest Neighbor(KNN), and the second is with Stochastic Gradient Descent (SGD). Apart from the Collaborative filtering technique, both use content information from recipes. The dataset used here is also from the 'food.com' website. The algorithm we used in our Collaborative-Filtering part of the project is called BaselineOnly, and it is simpler than these mentioned above. Still, according to the RMSE values, this one gave the best RMSE for our dataset. Apart from that, its implementation was much easier using the surprise library from python, and the Content-Based part of our project gives better results with cosine similarity. Also, in article [22], a hybrid model combines Content-Based and Model-Based Collaborative Filtering to propose a recipe recommendation system based on various ingredients. Moreover, it could find ingredients that are similar to make different recipes.

1.4 Thesis Organization

In Chapter 2 we discuss the the background of recommendation systems, their types and problems, along with the evaluation metrics.

In Chapter 3 we reference the various tools used to create our website.

In Chapter 4 we discuss the making of the website, both front-end and back-end.

In Chapter 5 includes the experimental evaluation of the recommendation systems.

In Chapter 6 we summarise the thesis and discuss possible future work.

Chapter 2

Background

In this chapter, we will explain what recommendation systems are, when someone can use them, and their types in detail. Moreover, feedback, ratings and their categories, which show the user's experience, are analyzed. Finally, the evaluation metrics of recommendation systems are described along with mathematical formulas, where necessary.

2.1 Introduction Recommendation Systems

The principal use for recommendation systems is to use collected data to determine what a user may like based on what they have already interacted with and provide recommendations. A secondary use is to supply the user recommendations for new content even if they would not typically try them. Recommendation systems help users cope with the problem of information overload by presenting them with the most engaging content for them and offering innovation. Recommendation technology is an effective solution to the problem of searching for information that arose along with the continuous increase in data.

2.2 Types of Recommendation Systems

When discussing recommendation systems, the two most popular types are content-based and collaborative filtering. Another type not as well-known is knowledge-based. Last but not least, we have hybrid recommendation systems that use a combination of techniques from each type.

2.2.1 Content-Based Recommendation Systems

Content-Based recommendation systems, as the name suggests, utilise the content users consume to provide recommendations. Specifically, each item in the database that the user has previously interacted with is divided into attributes. Afterwards, we handle those attributes to create recommendations based on the amount of similarity between items. For example, if a user interacts only with chicken-based recipes, the system will recommend them solely chicken-based recipes similar to those with high ratings from the user. Generally, if someone is interested in a specific item, the system will suggest content as similar to that item as possible. In contrast with Collaborative filtering systems, Content-Based ones do not have any problems with new users and items. This type of recommendation system works in two completely different methods and a variety of models and algorithms. The first uses the vector spacing method [3] and the second uses a classification model. There are some fundamental advantages to a Content-Based recommendation system. To begin with, it does not require data about other users. Additionally, it suggests niche things to the user, based on their preferences, that only a few other users may find engaging. Furthermore, when new items are added, they appear on the list of recommendations even if they are not rated yet. However, there are also disadvantages; for example, the suggestions are limited to the user's current interests, so there is no variety of subjects recommended and only a small amount of novelty. [1] [4] In Figure 2.1 there is an example of Content-Based filtering.

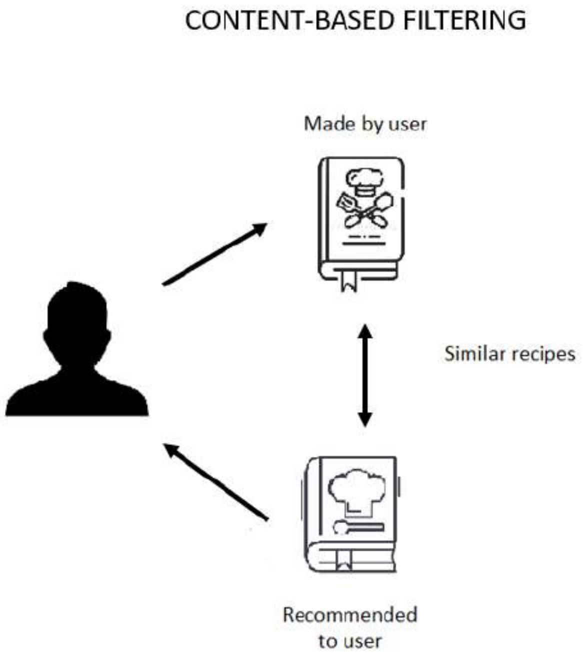


Figure 2.1: Content-Based Filtering example

2.2.2 Collaborative Filtering Recommendation Systems

Collaborative filtering is the most acknowledged category of recommendation systems. It uses the ratings of multiple users to recommend content. Computers nowadays allow us to process a massive amount of data so we can use the opinions of hundreds of users to come up with the best possible personalized recommendations. The main challenge of this type of recommendation system is that the 'ratings' matrix is sparse, as there are tons of recipes, for example, but a user only rates a few of them. In Figure 2.2 we present a Collaborative Filtering example. Mainly there are two methods used in collaborative filtering, Memory-Based and Model-Based. [1]

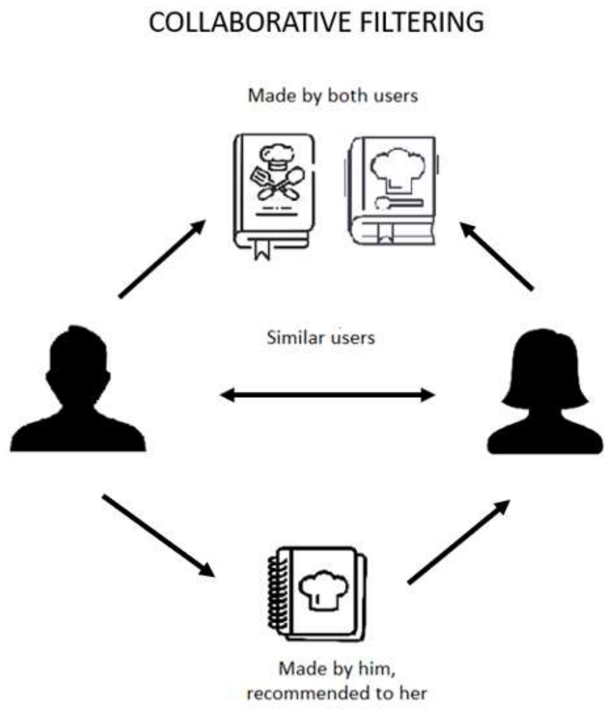


Figure 2.2: Collaborative Filtering example

2.2.2.1 Memory-Based method

The Memory-Based method is one of the earliest created approaches and is also known as neighbourhood-based collaborative filtering. This method generates recommendations by locating sets of users, known as neighbours, similar to the target user. It uses the entire matrix of the user-item ratings to find suggestions. Regardless, two fundamental problems occur, sparsity and scalability. The issue of sparsity arises because users do not rate many items, which results in a sparse matrix. Scalability, on the other hand, happens because the memory-based approach can not handle vast amounts of data (users and items). [7] The Memory-Based method is easy to implement but does not produce satisfactory results with sparse 'ratings' matrices. The method is divided into two subcategories: User-based and Item-based Collaborative filtering.

The first assumes that a user will be interested in an item if other users similar to them like this item as well. Generally, users are categorised as similar when they have many 'liked' items in common. So, the k more similar users to person A we find, the better the predictions for their unrated recipes, for example, will be. [1]

The second category is similar to the Content-Based method. In short, if we have a target item A and a user B, this method discovers sets of similar items to A and recommends them to user B. [1]

2.2.2.2 Model-Based method

In contrast, the Model-Based method groups different users into classes based on their rating patterns. [7] Machine learning and data mining methods are primarily used for predictions. Some of them are the Bayesian network, decision trees and rule-based models. [1] The model-based approaches are usually time-consuming to build and update and cannot cover as diverse a user range as the memory-based ones do.[7]

2.2.3 Knowledge-Based Recommendation Systems

Knowledge-Based recommendation systems are different from the others we discussed above. It uses other techniques to recommend content based on domain knowledge. A specific user will get recommendations based on their profile and feedback from other users will not be included in most cases. [11] This is the main difference between this recommendation system and Content-Based/Collaborative Filtering because these two give suggestions based on the past actions of the user and his peers. Knowledge-Based systems are separated into two subcategories, Constraint-Based and Case-Based recommendation systems. In Constraint-Based, users specify their needs or constraints (for example, limits) on the item attributes. In a Case-Based system, however, there are specific cases that are selected by the user as targets (or anchor) points. In Figure 2.3 the process of the system is observed. [1]

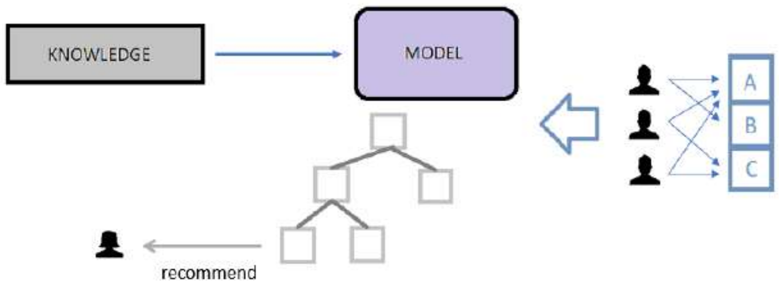


Figure 2.3: Knowledge-Based example

2.2.4 Hybrid Recommendation Systems

The recommendation systems specified above have many advantages. However, they contain enough problems that there was a demand for a different solution. The solution to that is Hybrid recommendation systems. These recommendation systems combine two or more of the individual systems mentioned in the previous chapters. In Figure 2.4, there is a hybrid system example which combines three different recommendation techniques. The result of merging the various aspects of recommendation systems is better performance with fewer drawbacks for any individual one. [1] Usually, Collaborative Filtering is combined with another type of recommendation system. For example, EntreeC is a hybrid recommendation system that combines Knowledge-Based and Collaborative Filtering. The knowledge-Based method helps the Collaborative Filtering because it creates an entry point where the target user rates an item so the item cold start problem does not exist anymore.[5] Some of the techniques of hybrid systems are:

- Weighted: A linear combination of predictions by the various methods of the recommendation system used is computed to come up with the final result.
- Switching: The hybrid system switches among the available methods based on the requirements each time.
- Mixed: The final result here is the results of each method added together without any combination. [6]
- Feature Combination: The outcome of the collaborative-filtering method is used as a

feature to build a Content-Based system over the augmented feature set.

- Cascade: In the cascade case, one system of recommendations refines the results given by another system.
- Feature augmentation: The results of one recommendation system are utilised as input features for the other system.
- Meta-level: The model used for a recommendation system is utilised as input for another system. [1]

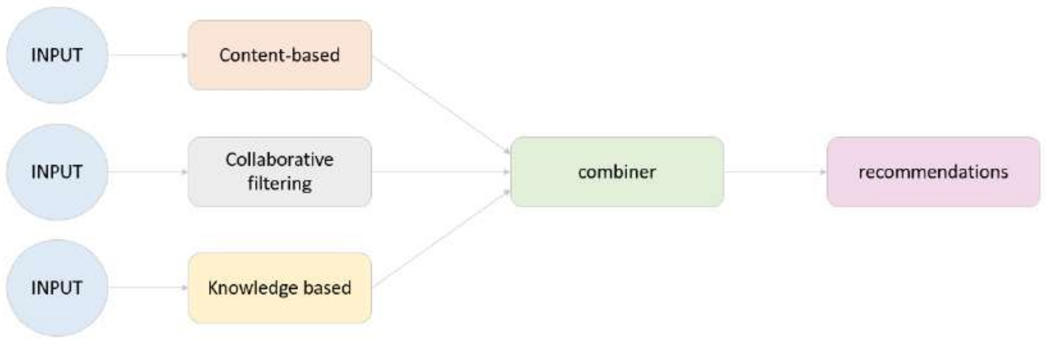


Figure 2.4: Hybrid system example

2.3 Problems of Recommendation Systems

A content-Based system has considerable disadvantages. This system is based solely on the objective information of items. Thus, it does not consider the subjective attributes of the item. Moreover, another problem is overspecialization since it suggests only items similar to those the user has rated highly. As a result, the user does not get any recommendations about other categories of items that they have not yet tried but may find interesting. Furthermore, the quality of the suggestions is not up to par unless there is enough interaction with the target user. [4] [11]

For Collaborative-Filtering recommendation systems, the main concerns are sparsity and scalability. The item-user ‘ratings’ matrix has many items, and the user’s ratings are too few compared to the number of items, which means the matrix is sparse. As for the problem of scalability, the memory-based approach can not handle large amounts of data (users and items). [7] Another problem is the item cold-start problem, which occurs when we add new items, and there is no feedback for them, so it is not recommended to the users. [4] Furthermore, an additional concern is the ‘grey sheep’ that occurs when we classify users into two or more different groups, making the similarity equivalent and the recommendations unreliable. [11]

The Cold Start problem occurs when there is an insufficient number of ratings and feedback to generate quality recommendations. [8] There are three types of cold start problems:

- New community: The lack of data (ratings) when creating a new recommendation system for making reliable recommendations.
- New item: When adding a new item with no ratings, the users will not get recommendations for this item.
- New user: When users register for the first time, they have not rated any items, so the system can not generate personalized recommendations about them. Apart from that, even if the user has rated only a few items, it is entirely possible that this will not be enough for the system to make satisfying suggestions causing the users to stop using the application. [9] [10]

Content-Based and Collaborative-Filtering recommendation systems suffer from the cold start problem, either ‘new item’, ‘new user’, or both. That is because they need past actions of users to produce recommendations. [6]

2.4 Comparison of Recommendation Systems

Each one of the recommendation systems mentioned above has advantages and disadvantages, and the results differ depending on the type of data. Specifically, a particular method may not produce satisfactory results for one kind of data because of the algorithm in use. Furthermore, it is challenging to generate recommendations for a user that does not interact with any content. Collaborative filtering works very well in this case. However, there are other concerns, such as the item cold-start problem and the issue that if a user likes only a specific category of items, there might be other available users with the same taste. A content-Based system does not have these types of problems and works well with new content. [2]

2.5 Feedback

Feedback and ratings are mentioned many times in this chapter by now. These ratings show the level users like or dislike a specific item.

2.5.1 Ratings

2.5.1.1 Continuous rating values

Ratings can be continuous values. Usually, in cases like this, the users have to select a value in a range (for example, 0 to 100). However, this method is not often used.

2.5.1.2 Interval-based ratings

Interval-based ratings are a method of feedback that is more in use than Continuous rating values. In Interval-based ratings, a discrete set of numbers in order are used to quantify positive or negative ratings. For example, a 5-point rating scale might look like this set −2,−1, 0, 1, 2, in which the lowest value of rating (−2) indicates an extreme dislike, and the highest value of rating (2) indicates a strong like to the item. Figure 2.5 contains an example of a 5-star rating method.

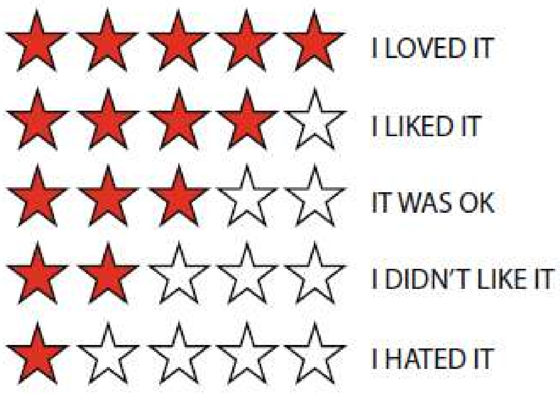


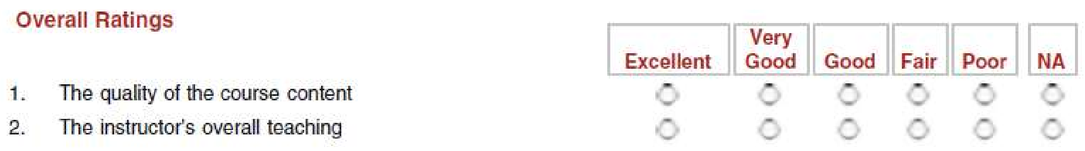
Figure 2.5: Interval-based rating example [1]

2.5.1.3 Forced Choice rating system

Consider a 5-star Interval-based method with the following set of numbers 1, 2, 3, 4, 5. If number 4 corresponds to 'really like' and number 3 corresponds to 'like', there are two choices for dislike and three for like, so we have an unbalanced rating scale as a number that corresponds to 'neutral' does not exist.

2.5.1.4 Ordinal ratings

If ordered categorical values are used like Strong dislike, Dislike, Neutral, Like, Strong like we have the Ordinal rating case. The name for the category comes from the fact that these attributes are ordinal. In Figure 2.6 an example of Ordinal rating used in Stanford University course evaluation forms is illustrated.



Overall Ratings

		Excellent	Very Good	Good	Fair	Poor	NA
1.	The quality of the course content	○	○	○	○	○	○
2.	The instructor's overall teaching	○	○	○	○	○	○

Figure 2.6: Ordinal rating example [1]

2.5.1.5 Binary ratings

For Binary ratings, there are only two options to rate an item; positively or negatively. The choices can be like or dislike (as seen on YouTube), 0 and 1, or unspecified values.

2.5.1.6 Unary ratings

Unary ratings allow the users to specify whether they like an item, but there is no option to express a negative rating. Unary ratings are often used, especially in implicit feedback data sets. For example, if a user buys an item, it means they liked it, but if they do not buy it, this does not mean that they did not like it. [1]

2.5.2 Implicit

Implicit feedback is the type of feedback the users are urged to make on their own, for example, rating a movie on IMDB. This feedback seems to be the most straightforward route for the recommendation system to understand what users like. However, this notion is inaccurate, as the feedback is inherently noisy and harder to interpret. [12] [13] That is why sometimes users rate an item in a way that does not correspond to their natural reaction and feelings about this specific item. Consider some users that declare they like vegan food recipes, but they have made chicken soup many times. Which one of these reflects their true preferences? Definitely the second one.

2.5.3 Explicit

The system generates explicit ratings by monitoring the users' actions. For instance, when users buy an item, it is possible that they are interested in this item. That means that the users would rate with a high value this item. Furthermore, if users ask for more information about an item or check it out on a website, it possibly means they like it. Because of the problem mentioned above in implicit feedback, explicit ratings are becoming more popular. [1]

2.6 Evaluation Metrics of Recommendation Systems

This section will analyse various ways of evaluating a recommendation system. We implement the evaluation process using two different methods, in general, online and offline.

The user's reactions are measured in online methods concerning the presented recommendations. In order to achieve this, active users are essential. For instance, in an online evaluation of a recipe recommendation system, one might measure the percentage of users clicking on recommended recipes. These methods are known as A/B testing and what they actually do is measure the direct impact of the recommendation system on users. However, because of the necessity of active users, this method is usually not feasible in research. [1]

On the other hand, offline evaluation methods are used much more frequently with static data resources and analogous evaluation metrics to estimate numeric effectiveness measures that can be tuned for and compared. In this type of evaluation, the existing ratings are divided into train set and test set, with the choice of where a rating is placed, train or test, as an attribute of (u, i) combined and not as individuals(u or i). The training set has rating data that are used as an input field to the recommendation system to build a model of user-item interplays. Then, a score has to be set for each item of the train set affiliated with the target users or to each item in the other set for the users, as mentioned earlier. [18]

The following issues are significant when designing evaluation methods for recommendation systems:

- **Evaluation goals:** Although accuracy metrics are commonly used for evaluating recommendation systems, the image of the users' experience may not be total. Even if they are the most valuable part of the evaluation, many other objectives such as innovation, trust, coverage and serendipity are also essential for the users' experience.

- **Experimental design issues:** Even when we use accuracy as the metric, it is critical to design the tests so that the accuracy is not overestimated or underestimated. For instance, if the same set of defined ratings is used both for model development and for accuracy evaluation, the accuracy will be highly overestimated.

- **Accuracy metrics:** This one is the most significant evaluation part of the evaluation. Recommendation systems can be assessed either as the prediction accuracy of a rating or the accuracy of ranking the items. So, common metrics like the mean absolute error (MAE) and mean squared error (MSE) are regularly used. The estimation of rankings can be accomplished using numerous methods, like utility-based computations, rank-correlation coefficients, and the receiver operating characteristic curve. [1]

2.6.1 Coverage

As reported by [14], the coverage of a recommendation system is a measure of the field of items over which the system can make recommendations. The term coverage was mainly related to two approaches: (i) the percentage of the items for which the system can make a suggestion, and (ii) the percentage of the accessible items which successfully are at least once recommended to a user [14] [17]. Therefore, even though various authors differ regarding terminology, we choose the definition from [14] and refer to (i) as prediction coverage and (ii) as catalogue coverage.

Prediction coverage is mainly based on the recommendation engine and its input. If we have a collaborative filtering system, the inputs are item ratings. The system will be capable of making suggestions for items for which it obtained enough input. In our case, it will generate reliable recommendations if we have enough ratings. If we define the available items and I_p the items that the system can make predictions for, a fundamental measurement for prediction coverage can be given by:

$$Prediction\,coverage = \frac{|I_p|}{I}$$

Assuming that r(x) gives the usability of item x, a weighting factor in the computation of the coverage is introduced for taking the corresponding usefulness of items in a recommendation list into account:

$$Weighted prediction\, coverage = \frac{\sum_{i \in I_p} r(i)}{\sum_{j \in I} r(j)}$$

Catalog coverage can be a highly valued measure for systems that suggest lists of items (e.g. top 5 most appropriate items) since the above coverage does not consider this. Catalog coverage is generally measured on a set of recommendation sessions; for instance, by examining for a specified period, the recommendations returned to users [14]. If we consider I_L^j as the set of items included in the list L returned by the j^{th} suggestions noticed during the measurement time. N is the total number of suggestions observed during the measurement time, and let 'I' be the set of the entire accessible items. The measurement of catalog coverage is as follows:

$$Catalog\, coverage = \frac{|\bigcup_{j=1...N} I_L^j|}{|I|}$$

If B^j refers to the set of items that may be valuable to be returned in suggestion j, the computation of the Weighted Catalog coverage is given as follows:

$$Weighted Catalog\, coverage = \frac{|\bigcup_{j=1...N} (I_L^j \cap B^j)|}{|\bigcup_{j=1...N} B^j|} [14]$$

2.6.2 Serendipity

Serendipity is equal to 'lucky discovery', which means that we measure how obvious the recommendations the system makes are. Serendipity is something bigger than a novelty, which depends on how much the user suspects what recommendation results he will see. For example, consider a user who makes recipes from Category A, and the recommendation system suggests new recipes from Category B. These recipes will be new to the user. However, these categories might relate to each other, so these recommendations will not be serendipitous. However, if these categories are unrelated, the recommendations are more unexpected and serendipitous. [1] Consider PM as a set of suggestions made by a primitive prediction model and RS as the recommendations made by a recommender system. Elements from RS that do not appear in PM are denoted as unexpected recommendations. So, we calculate the unexpected set of recommendations with the following equation:

$$UNEXP = RS \backslash PM$$

Although, the unexpected recommendations are not at all times useful. So, u(RS\RM) is used to describe the usefulness of the recommendations mentioned before. If $u(RS_i) = 1$ the prediction is useful, else if $u(RS_i) = 0$ it is not. N is the overall number of data in UNEXP. The following equation determines the serendipity: [15]

$$SRDP = \frac{\sum_{i=1}^{N} u(RS_i)}{N}$$

2.6.3 Novelty

Novelty has to do with the percentage of the recommendations a recommendation system makes that the target user was unaware of or has not seen again. The unexpected recommendations help users discover whether they have other preferences they have not discovered yet. Moreover, it helps with recipes, for example, or categories they dislike. In some types of recommendation systems, like Content-Based methods, the suggestions are generally relevant to each other because some features are combined, and their similarity leads to the final suggestions. However, these recommendations are too obvious, so there is no novelty here. The most efficient way of measuring novelty is to ask the users if they have ever seen or thought of the recommended items. However, this online method is usually complex because it requires large databases with users. However, some offline methods are available, considering we have timestamps together with the ratings. Novelty recommendation systems' target is to suggest items that a user is more likely to choose in the future. [1]

2.6.4 Privacy

The privacy issue comes with Collaborative Filtering systems, in which a user shares his preferences over items with the system to get some valuable recommendations. However, these preferences must stay private; that is, no one else can use the recommender system to gain information about the preferences of a particular user. It is inappropriate for a recommendation system to reveal private information about users, not even for a single one. Last but not least, privacy may come at the expense of the accuracy of the suggestions. So, it is significant to examine this trade-off thoroughly. [17]

2.6.5 Accuracy Metrics

The accuracy must be measured over the test set to conclude that a recommendation system makes satisfying and efficient recommendations. [1] First of all, to begin the process, a recommendation system must have a data set including several items (I). Also, some of these items have to be rated implicitly or explicitly in one of the two ways mentioned in section Feedback. Then, the system chooses a number of these items and suggests them to the target user in a specific form, e.g. ordered list. The most critical issues here are:

- the set of the selected items
- in which order these items are(in case of ordered list)

To measure the accuracy of the recommendation system, the thing that is taken most into account is how close the utility of the shown object is considering the preferences of the target user. Generally, accuracy is the most known metric in Artificial Intelligence, and the following equations can describe it: [16]

$$accuracy = \frac{number\,of\,good\,cases}{number\,of\,cases}$$

$$accuracy = \frac{number\,of\,successful\,recommendations}{number\,of\,recommendations}$$

In many cases, the difference between the actual and predicted value of the recommendation system is calculated to measure the accuracy. This value is called error,

$$error = real\,value - predicted\,value$$

2.6.5.1 Root Mean Squared Error (RMSE)

This metric is the most popular one used to measure predicted ratings' accuracy. The system sets up predicted ratings $\hat{r}_{ui}$ for a test set T containing user-item pairs (u,i). Apart from that, the real ratings r_{ui} are known because the users have previously rated the items explicitly or implicitly. The RMSE value among the predicted and actual ratings is calculated by:

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{(u,i)\epsilon T}(\hat{r}_{ui} - r_{ui})^2}$$

One feature of the RMSE is that it disproportionately penalises significant errors through the squared term within the summation. [17]

2.6.5.2 Mean Absolute Error (MAE)

Mean Absolute Error calculates the absolute value of the difference between the predicted value and the actual value. For example, let 'i' be the first prediction in which the algorithms predict a value much bigger than the actual and ii the second prediction with a predicted value much smaller than the actual. These two predictions will cancel each other out, resulting in a small error value. However, only positive values will be calculated using the absolute value so that the error will be the actual addition of the two predictions' errors. The equation for MAE is the following:

$$MAE = \sqrt{\frac{1}{|T|}\sum_{(u,i)\in T}|\hat{r}_{ui} - r_{ui}|}$$

Mean Absolute Error, compared to Root Mean Squared Error does not disproportionately penalizes large errors. [17]

2.6.5.3 Disadvantages of measuring Errors

The main problem with the above metrics is that usually, most users do not consider as significant the fact that recommendation systems can predict their rating with decimal precision. Instead, most likely, they are interested in the list of items the system returns. Moreover, not much weight is given to whether the system can predict the rating for items users have rated low.

Using these measures has the problem that all ratings are considered essential. If we look at the problem from the scope of top-k recommendations and not from the prediction of the rating, the issue is if the recommendations are good enough. On the other hand, the items not included in the top-k list are almost indifferent. So, the measures mentioned in the next section are more important than these.

2.6.6 Information retrieval measures

In many cases, applications do not have recommendation systems that predict the rating the user will give to an item. However, it attempts to recommend items common to these the target user likes. For example, if a user adds a recipe to 'favourite recipes', the system tries

	Recommended	Not recommended
Used	True-Positive (TP)	False-Negative (FN)
Not used	False-Positive (FP)	True-Negative (TN)

Table 2.1: Classification of the possible result of a recommendation of an item to a user.

to suggest similar recipes that the user might add to favourites in the future. So, we measure whether the user actually adds the predicted items to liked items or not.

A common technique is that a specific user is selected from a data set, and some of the user's ratings are removed from the data set. Then, the recommendation system has to predict some items the user might like and compare them to those removed. After that, there are four possible outcomes for the recommended and hidden items, as shown in Table 2.1

- **True Positive (TP) :** An item is recommended and used by the user.
- **False Positive (FP) :** An item is recommended but the user has not used it.
- **True Negative (TN) :** An item is not recommended and the uses has not used it.
- **False Negative (FN) :** An item is not recommended but the user used it.

Since the data is not collected during the recommendation system evaluation, all unused items must be set as indifferent and useless to the target user, as if the user would never be interested in any of those. However, this is not always true because a user may not have discovered the existence of an item or a category of items, so they may be helpful after they have been listed in the recommendation results. In this case, the number of 'FP' (false positive) is overestimated.

The number of examples that fall into each cell in the table can be counted, and the following quantities may be computed:

$$Precision = \frac{\#TP}{\#TP + \#FP}$$

$$Recall(TruePositiveRate) = \frac{\#TP}{\#TP + \#FN}$$

$$FalsePositiveRate(1 - Specificity) = \frac{\#FP}{\#FP + \#TN}$$

Usually, we foresee trade-offs among these quantities. Besides, longer suggestion lists improve recall, but the precision may decrease. In applications where the number of returned recommendations is fixed, the best measure is Precision at N. On the other hand, if the recommendation lists are not fixed, curves comparing precision to recall or true positive to false positive are computed to evaluate the algorithms. The first type is the precision-recall curve, and the other is the Receiver Operating Characteristic or ROC curve. Although both curves measure the percentage of recommended items, precision-recall curves emphasize the proportion of chosen suggested items. In contrast, ROC curves emphasize the proportion of items not chosen that end up being recommended. [17]

Chapter 3

Development Environment & Tools

To develop the web application that supports the development of recommendation systems, we needed to find the tools that would help us most in our endeavour. After carefully considering all available means at our disposal, we selected the following languages and tools.

3.1 Python

For developing our thesis, we used Python programming language and, to be more specific, version 3.9.0. Python is an OSI-approved open source license, making it freely usable and distributable, even for commercial use [18]. It is an object-oriented language created by Guido van Rossum and released in 1991. Even though Python has been around for over thirty years, it is still one of the most popular programming languages. The reason why is because Python can be used for several things, such as:

- create web applications
- connect to database systems
- perform complex mathematical calculations
- software development

making it versatile. Another reason why Python has become so widely used is because of its simple syntax and readability, making it one of the easiest programming languages for beginners and experienced programmers alike to grasp and use. This, along with the vast libraries available with ready-to-use functions, makes Python a valuable asset in a programmer's toolkit.

3.2 Django Framework

For designing the website, we used Django Framework. Django is a free, open-sourced Python Framework, released publicly on 2005, and its principal use is to make the process of creating a website more accessible. That means that Django focuses on the reusability of its content (aka DRY; Don't Repeat Yourself) with ready-to-use functions that allow the user to concentrate on developing the web application.

Django has several features that make it appealing when choosing an approach to creating a website.[23] Such as:

- **Super Fast:** designed to help developers create web apps quickly.
- **Fully loaded:** user authentication, content administration, site maps, RSS feeds etc., are all taken care of by Django.
- **Reassuringly secure:** SQL injection, cross-site scripting, cross-site request forgery and clickjacking are some of the issues with security that Django helps overcome with its secure user authentication.
- **Exceedingly scalable:** Django helps with website traffic demand.
- **Incredibly versatile:** Build anything from content management systems to scientific computing platforms.

Furthermore, Django follows the MVT (Model-Views-Template) architectural pattern. The 'Model' assists with handling the data that the developer works with; more often than not, the data are from databases. The 'View' deals with the appearance and is the layer that the user sees and interacts with. Finally, the 'Template' handles the logic and functionality of the web application.

3.3 Bootstrap

To design the website more efficiently, we used Bootstrap version 5. Bootstrap is currently the most popular HTML, CSS and JavaScript Framework available for developing responsive websites released on 2011 [24]. Bootstrap is a front-end framework that includes base templates for web page content that is regularly used. That includes templates for:

- Cards, card groups and card decks.
- Buttons
- Modals
- Tables
- Image carousels
- Navigation bars
- Forms

along with many others. Those ready-to-use templates from Bootstrap have alternatives for a bit of a selection, for example, buttons have eight colours to choose from and tables have a dark theme apart from the default light theme. However, if the developer needs more customisation, that is achieved using CSS. Furthermore, one of the benefits of using Bootstrap while designing a web page is that it helps achieve a uniform look throughout.

3.4 Anaconda

Anaconda is a Python and R language distribution released on 2012. Anaconda helps simplify the process of downloading and managing packages, and virtual environments [25]. Its principal use is for solving complex scientific problems using machine learning. Apart from being a package manager, Anaconda includes the Anaconda Navigator, a graphical user interface (GUI) that allows developers to access valuable tools from a user-friendly desktop application. The applications that are available through Anaconda Navigator are the following:

- JupyterLab
- Jupyter Notebook
- QtConsole

- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

3.5 PostgreSQL & pgAdmin 4

For managing our database, we used PostgreSQL. PostgreSQL is an open source object-relational database system that extends the SQL language, released on 1996 [26]. It was initially designed for Unix-type systems; however, as its popularity grew, they developed it for other operating systems like Windows. The main uses of PostgreSQL are:

- General use database for applications and websites.
- Geo-spatial and analytics applications.

PostgreSQL has multiple features that make it compelling for database management, with more added with every significant release. Bellow, we see a list of a few of them:[26]

- **Data Types :** everything from integers and strings to date/time and arrays, JSON files and custom types.
- **Data Integrity :** with features like primary and foreign keys, explicit and advisory locks etc.
- **Concurrency, Performance:** with features like Multi-Version concurrency Control (MVCC), Table partitioning, indexing etc.
- **Reliability, Disaster Recovery :** a reason why PostgreSQL is popular is its readability. With that, features like Write-ahead Logging (WAL) and tablespaces help.
- **Security :** PostgreSQL's authentication system is one of the most useful features.
- **Extensibility:** extensions like PostGIS (for geo-spatial databases and geographical information) add to the functionality of the database.

- **Internationalisation, Text Search :** PostgreSQL provides support for international characters and is case and accent-insensitive.

To help with the management of the database, we used pgAdmin4. PgAdmin is a GUI tool that is free and open-source. [27] The software has the look and feel of a desktop application, making it easy for the developer to handle databases in PostgreSQL even if they have little to no prior knowledge. PgAdmin is available on both website and desktop application. It assists in managing the schemas and relationship visualisation among them.

Chapter 4

Design & Implementation

A large amount of data is necessary to implement the content-based and collaborative filtering recommendation systems. This chapter will present the data used, the process of creating the recommendation systems, and the website application they work on.

4.1 Dataset

The dataset we used is 'Recipes.csv' and 'Reviews.csv' from Food.com, found on Kaggle.[28] It consists of two csv files and the first one, 'Recipes', contains the following elements:

- **RecipeId :** A unique id for each recipe
- **Name :** The name of each recipe
- **AuthorId :** The unique id of the author who wrote the recipe
- **AuthorName :** The name of the author who wrote the recipe
- **CookTime-PrepTime-TotalTime :** The cooking time, the preparation time and the summation of these two values
- **DatePublished :** The date the recipe was published
- **Description :** The description of the recipe written by the author
- **Images :** Some images of the result of making the recipe
- **RecipeCategory :** The category the recipe belongs to

- **Keywords :** Some keywords about the recipe
- **RecipeIngedientQuantities :** The quantities for each ingredient of the recipe
- **RecipeIngedientParts :** The parts of the ingredients
- **AggregatedRating :** The aggregated rating for each recipe
- **ReviewCount :** How many reviews each recipe has
- **Calories, FatContent, SaturatedFatContent, CholesterolContent, SodiumContent, CarbohydrateContent, FiberContent, SugarContent, ProteinContent :** The amount of the above variables that each recipe contains
- **RecipeServings :** The servings of each recipe
- **RecipeYield :** The yield of each recipe
- **RecipeInstructions :** The instructions to make the recipe

The second file, 'Reviews', contains the following elements:

- **ReviewId :** The unique id of each review
- **RecipeId :** The id of the reviewed recipe
- **AuthorId :** The id of the author who wrote the review
- **AuthorName :** The name of the author who wrote the review
- **Rating :** The rating of the author for the recipe
- **Review :** The review of the recipe
- **DateSubmitted :** The date the review was submitted
- **DateModified :** The date the review was modified

Because the dataset with the recipes was too large, we deleted some recipes. To begin with, we deleted the rows with 'null' and 'character(0)' as the value. Subsequently, we only kept the recipes reviewed at least once.

For a better understanding of the dataset 'reviews', we performed exploratory data analysis (EDA), with the following results:

- **Distribution of the number of ratings:**

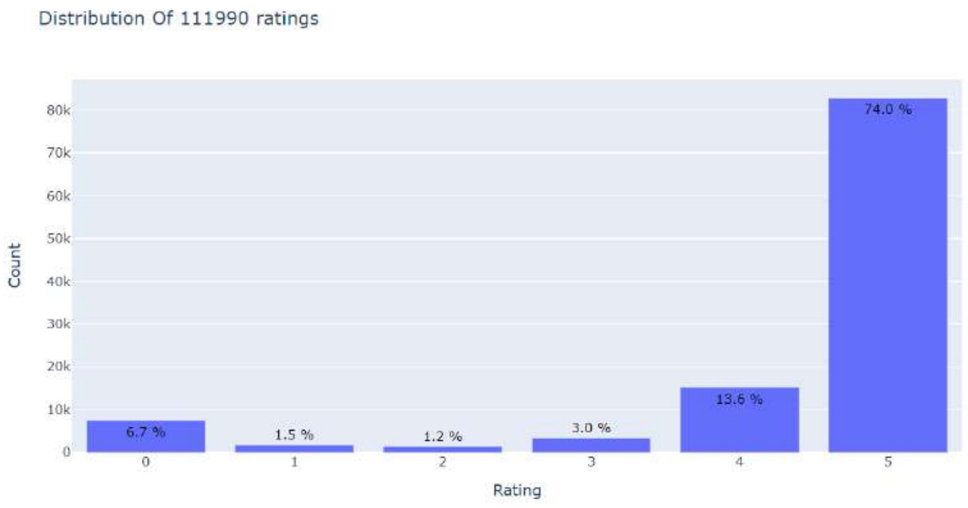


Figure 4.1: Distribution of ratings

In Figure 4.1, we observe that 74% of all ratings in the data are five (5), and very few are less than this. In general, low ratings show that the recipes are not that good.

- **Distribution Of Number of Ratings Per Recipe:**

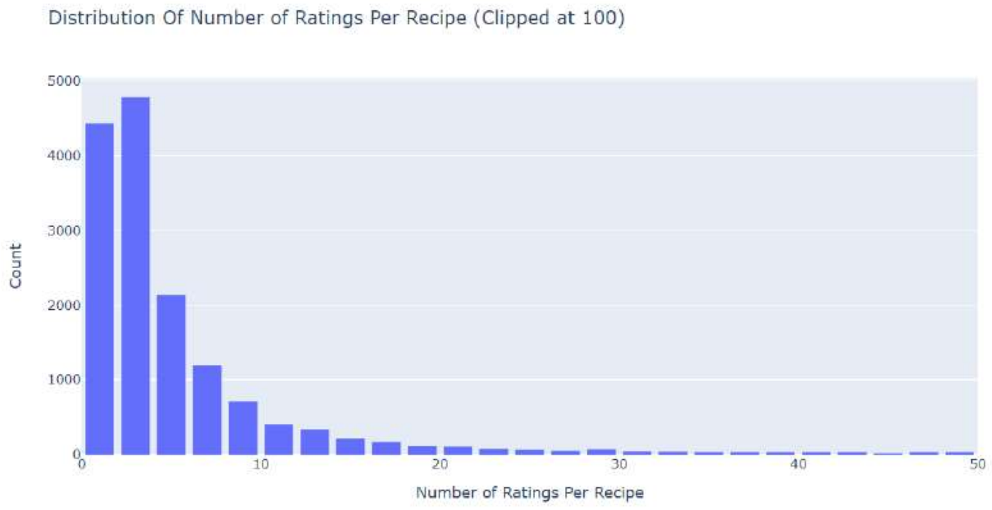
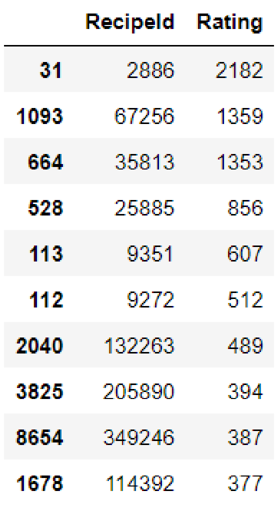


Figure 4.2: Distribution of Number of Ratings Per Recipe



	RecipeId	Rating
31	2886	2182
1093	67256	1359
664	35813	1353
528	25885	856
113	9351	607
112	9272	512
2040	132263	489
3825	205890	394
8654	349246	387
1678	114392	377

Figure 4.3: Most Rated Recipes

Here, in Figure 4.2, we see that most recipes receive less than seven (7) ratings, and very few recipes have many ratings, although the most rated recipe has received 2,182 ratings, as we observe in Figure 4.3.

- **Distribution of Number of Ratings Per User:**

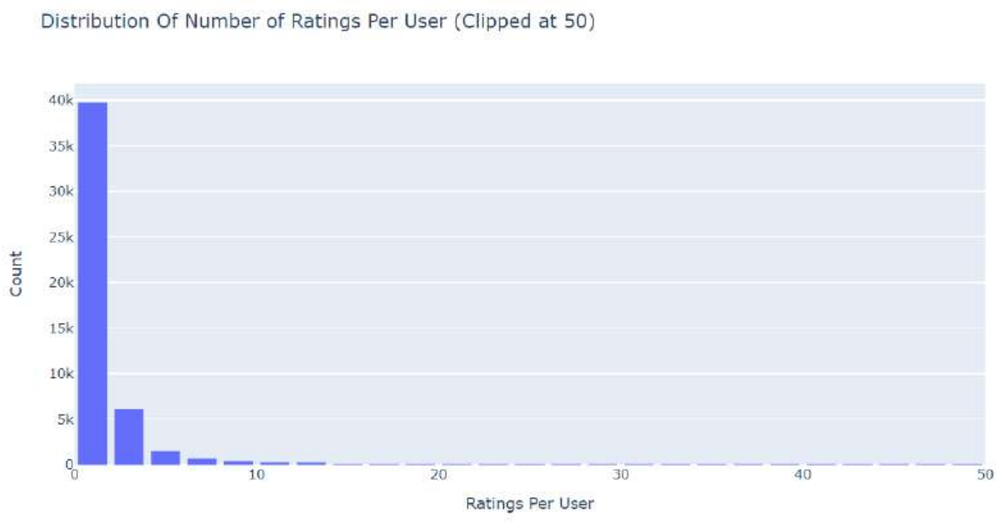
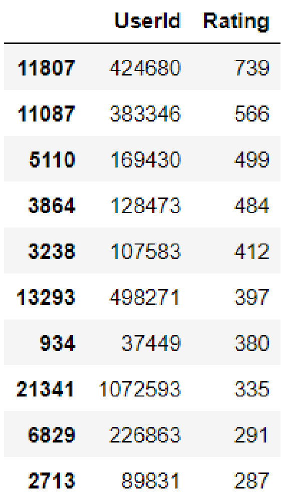


Figure 4.4: Distribution Of Number of Ratings Per User



	UserId	Rating
11807	424680	739
11087	383346	566
5110	169430	499
3864	128473	484
3238	107583	412
13293	498271	397
934	37449	380
21341	1072593	335
6829	226863	291
2713	89831	287

Figure 4.5: Users With Most Ratings

We noticed in Figure 4.4, that most users gave less than five (5) ratings, and very few gave many ratings, although the most productive user has given 739 ratings as we see in Figure 4.5.

4.2 Front-End

In this section, we will talk about the look of our website and what a user can do once they make a profile, along with screenshots for visual aid.

4.2.1 Login & Registration of a User

The first page a user comes across on the website is the login page 4.6. If they already have a profile, the user can fill in their credentials to continue using the website.

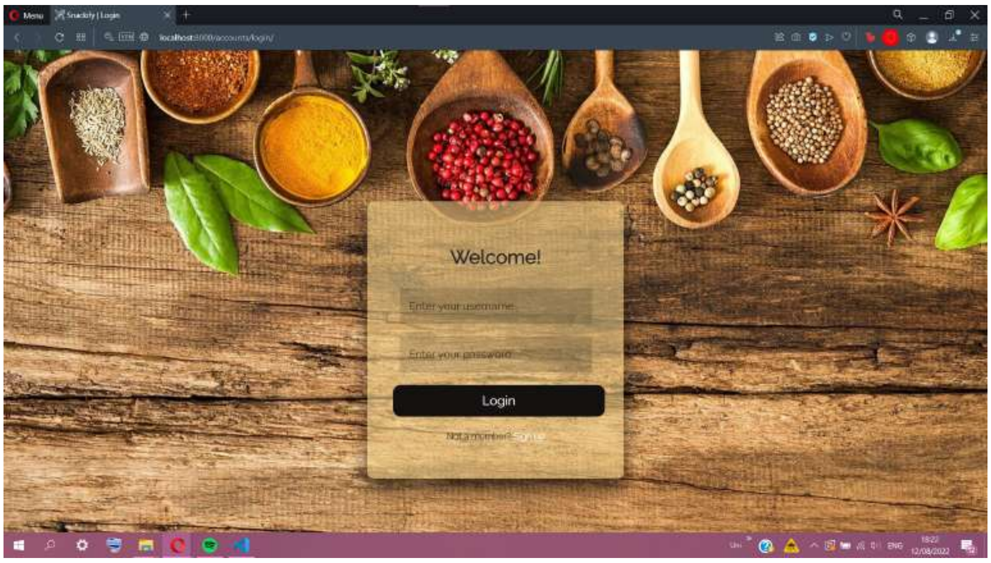


Figure 4.6: Login Form.

If the user is not registered, there is a sign-up prompt below the login button where the user can create a profile. The form requires the user's full name 4.7, username, email 4.8 and password 4.9. As soon as the registration is finished, the user can use their username and password to log in.

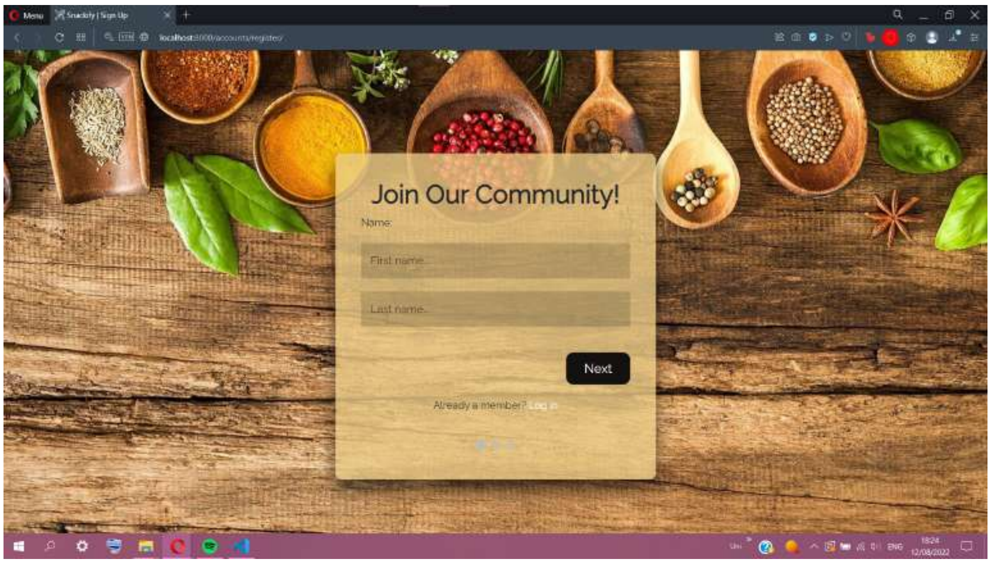


Figure 4.7: Registration Form - Step 1.

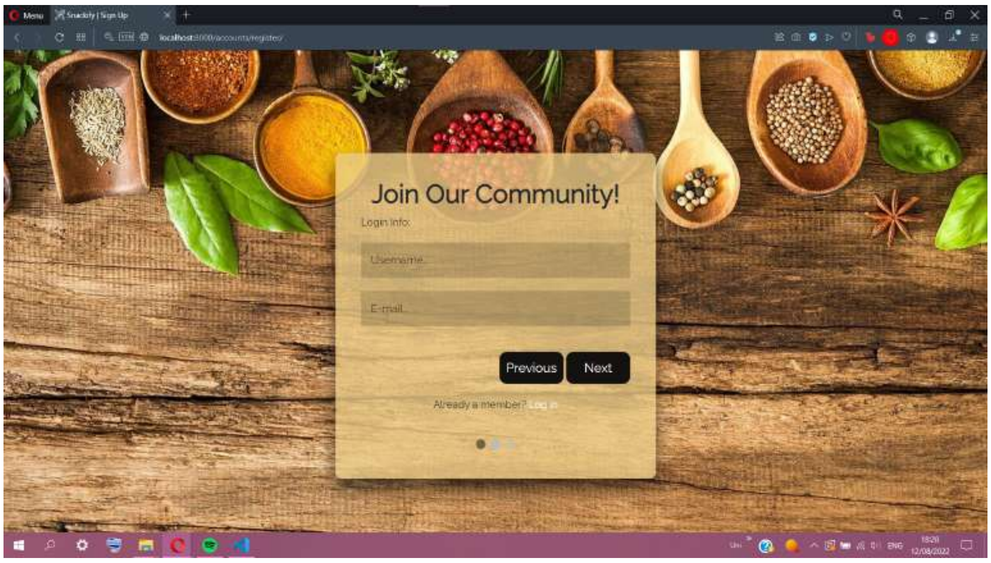


Figure 4.8: Registration Form - Step 2.

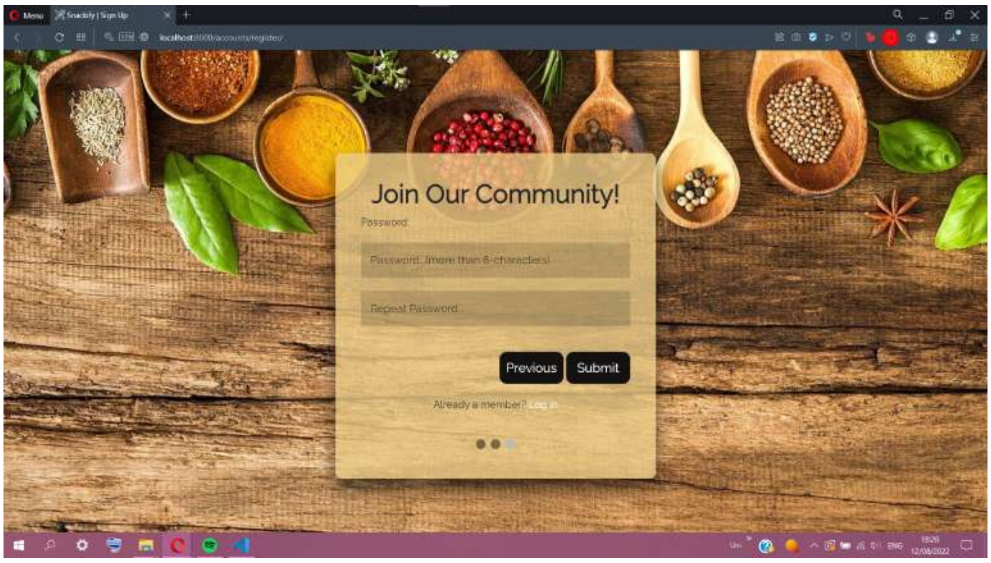


Figure 4.9: Registration Form - Step 3.

4.2.2 Home page

Once successfully logged in, the default redirection sends the user to the home page of our website 4.10. At the top, we can see the header bar, where the name and logo of our website are prominent. On the right is a functional search bar, and on the left is a button that opens the side navigation panel.

In the middle of the screen, the user finds another search bar with the prompt to find a recipe. Typing anything there provides the user with the recipes most relevant to the search query.

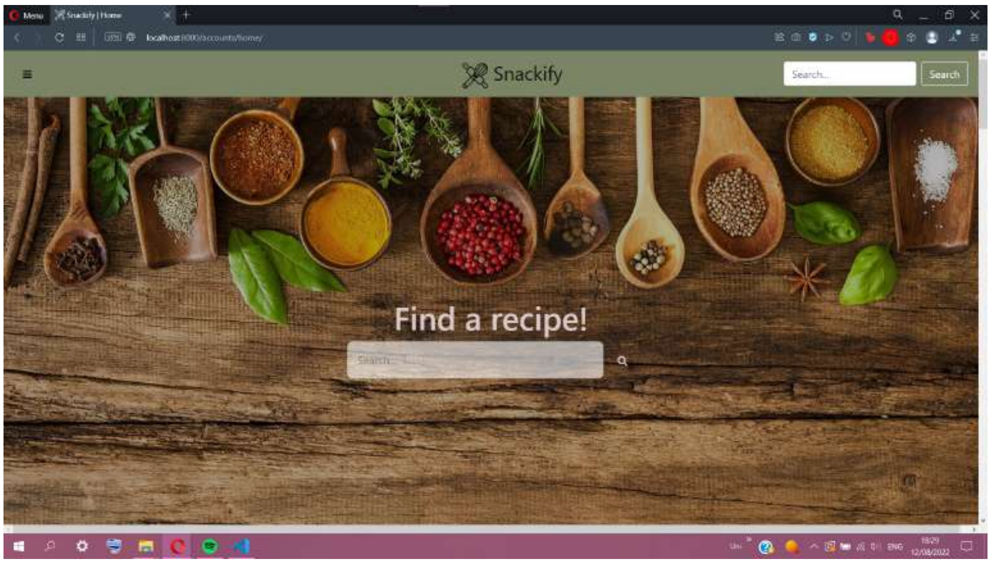
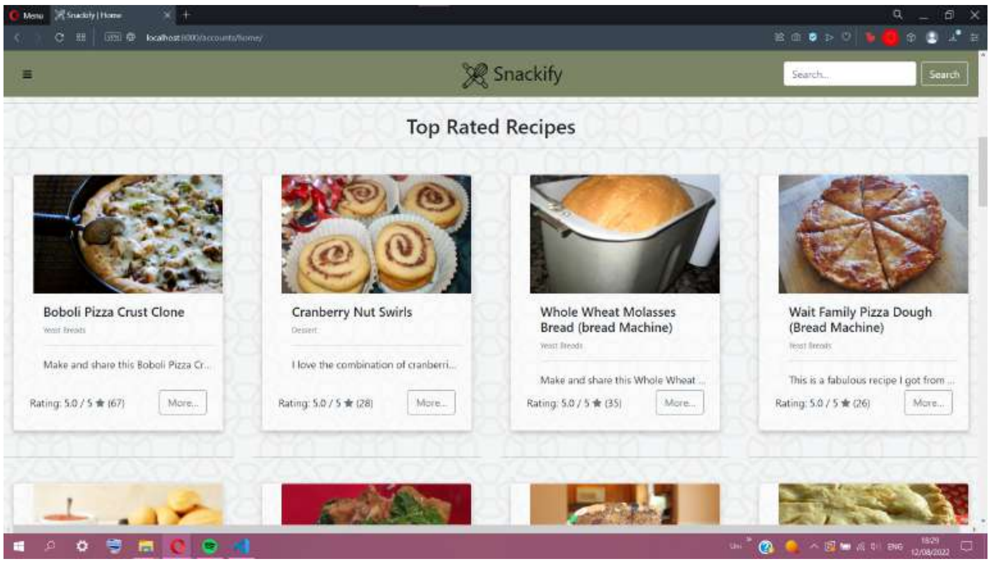


Figure 4.10: Home page.

Scrolling past that, the user finds the first few recommended recipes, comprised of the Top Rated recipes in our database. These 'Top Recipes' 4.11 are an assortment of the highest-rated ones and those with the most reviews. They are thirty-two (32) in total.



Top Rated Recipes

Boboli Pizza Crust Clone
Yeast Breads
Make and share this Boboli Pizza Cr...
Rating: 5.0 / 5 ★ (67) More...

Cranberry Nut Swirls
Dessert
I love the combination of cranberri...
Rating: 5.0 / 5 ★ (28) More...

Whole Wheat Molasses Bread (bread Machine)
Yeast Breads
Make and share this Whole Wheat ...
Rating: 5.0 / 5 ★ (35) More...

Wait Family Pizza Dough (Bread Machine)
Yeast Breads
This is a fabulous recipe I got from ...
Rating: 5.0 / 5 ★ (26) More...

Figure 4.11: Top Rated recipes.

4.2.3 Side Navigation

Clicking on the button on the top left side of the header bar opens the side navigation panel of our website. 4.12 Clicking anywhere on the screen, outside the side panel, closes the side navigation panel once again.

Each of the different sections are links to different parts of the website. When clicked, the first link, ‘Snackify’ (the name of our website), returns the user to the home page. At the bottom of the panel, we find the ‘Logout’ button, which redirects the user to the login page, restricting access to the website until the user enters their credentials once again. In the ‘About Us’ part, we find a page containing details about this website’s creators and how to contact us. Regarding the rest of the sections, we analyze each below in more detail.

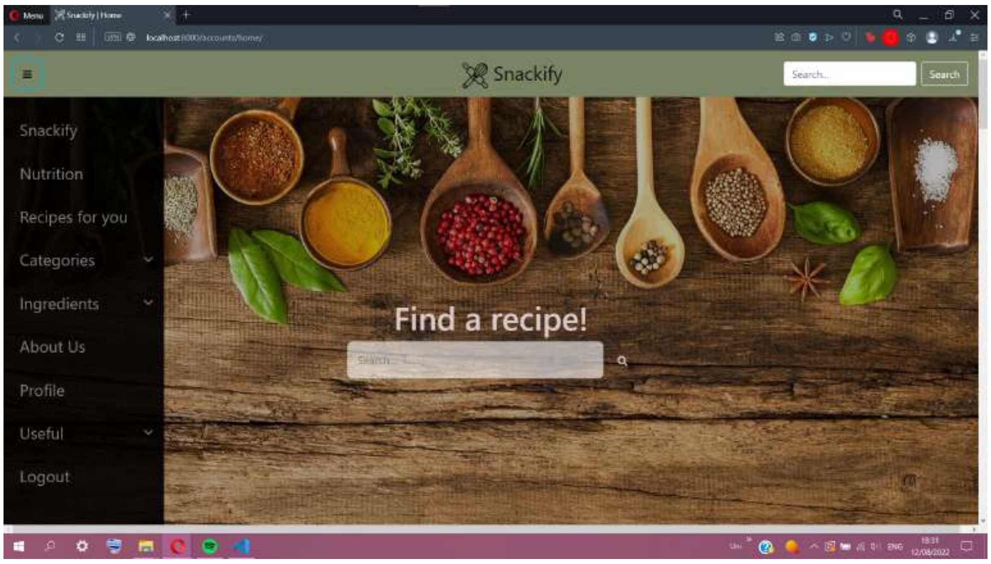


Figure 4.12: Side Navigation panel.

4.2.4 Nutrition

In this section, the user is prompted to enter the name of any recipe on the website 4.13. The recommendation system we built will handle the input and recommend recipes with similar nutritional value to the one the user has provided.

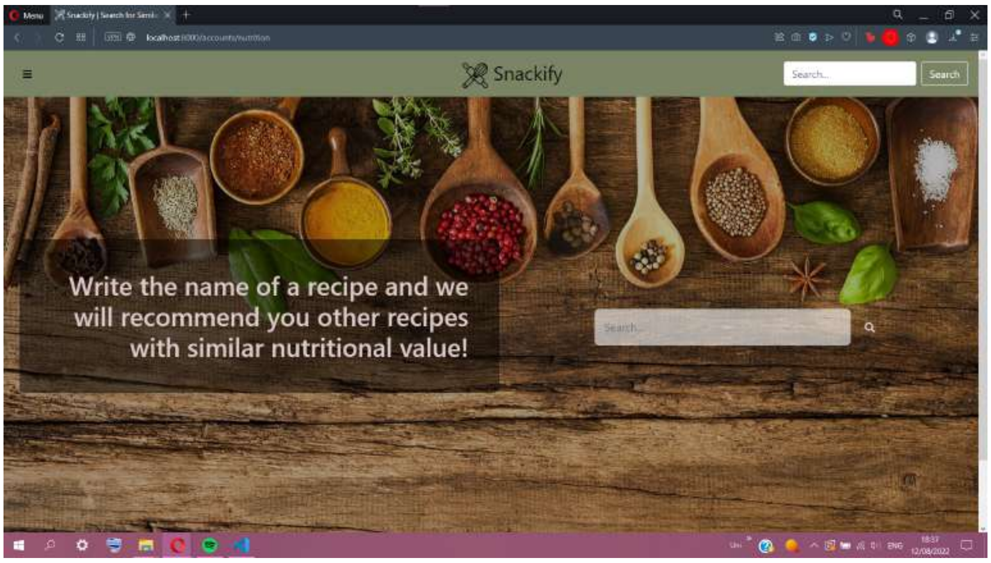
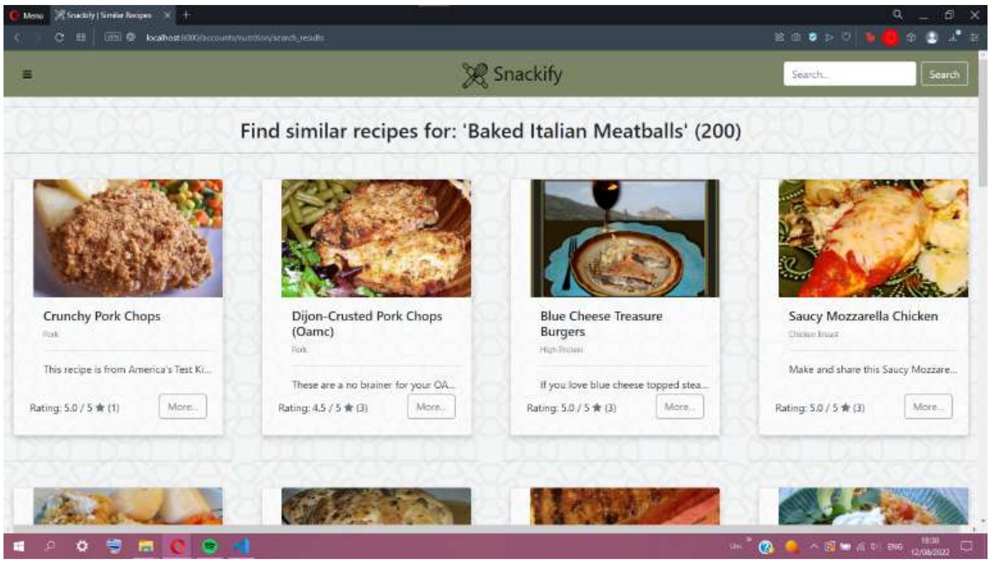

Snackify
Search...
Search
Write the name of a recipe and we will recommend you other recipes with similar nutritional value!
Search...

Figure 4.13: Find recipes with similar calories.

For example, let us say that the user wants to find recipes similar to 'Baked Italian Meatballs'; when they click enter, they see the results below:



Find similar recipes for: 'Baked Italian Meatballs' (200)

Crunchy Pork Chops
Pork
This recipe is from America's Test Ki...
Rating: 5.0 / 5 ★ (1)
More...

Dijon-Crusted Pork Chops (Oamc)
Pork
These are a no brainer for your OA...
Rating: 4.5 / 5 ★ (3)
More...

Blue Cheese Treasure Burgers
High Protein
If you love blue cheese topped stea...
Rating: 5.0 / 5 ★ (3)
More...

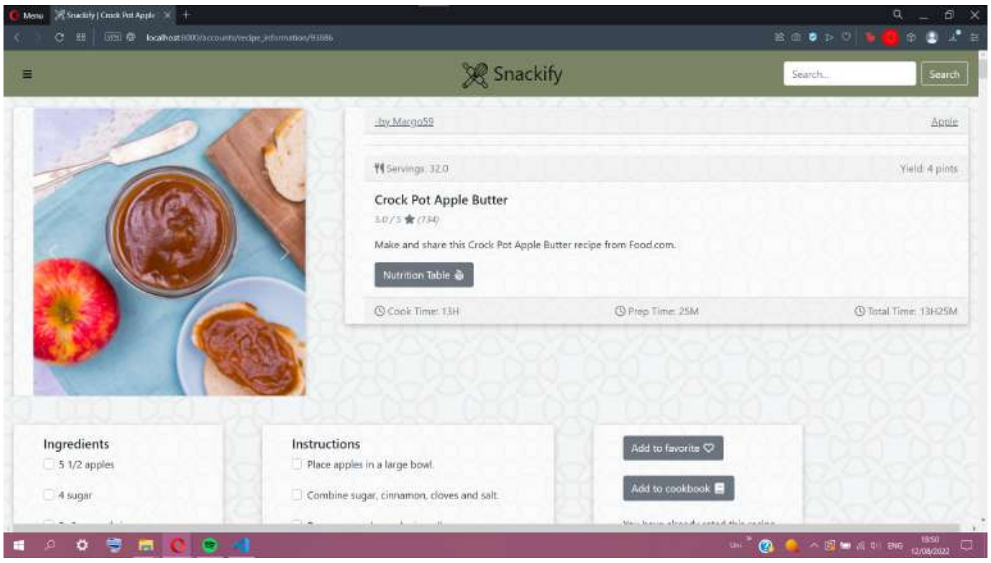
Saucy Mozzarella Chicken
Chicken Breast
Make and share this Saucy Mozzare...
Rating: 5.0 / 5 ★ (3)
More...

Figure 4.14: Recipes with similar calories.

This is only the first row of results. The number of recommended recipes varies with each input.

4.2.5 Recipe Page

After clicking on any available recipes on the site, the user can see the recipe information page 4.15.



-by Margo59 Apple

Servings: 32.0 Yield: 4 pints

Crock Pot Apple Butter

5.0 / 5 ★ (134)

Make and share this Crock Pot Apple Butter recipe from Food.com.

Cook Time: 13H Prep Time: 25M Total Time: 13H25M

Ingredients

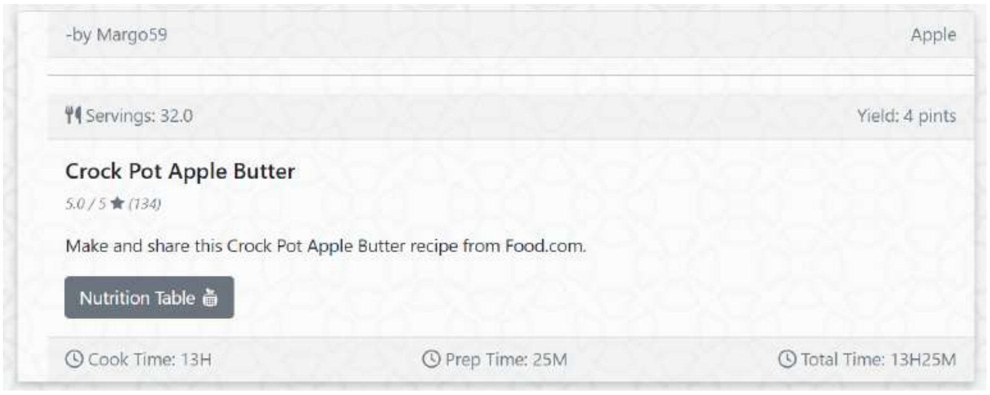
- 5 1/2 apples
- 4 sugar

Instructions

- Place apples in a large bowl.
- Combine sugar, cinnamon, cloves and salt.

Figure 4.15: Recipe page.

At the top of the page, on the left-hand side, we find a carousel of images from the recipe taken by the users that have tried it. On the right-hand side, there is most of the information about the recipe 4.16. Let us look at it a little closer.



-by Margo59 Apple

Servings: 32.0 Yield: 4 pints

Crock Pot Apple Butter

5.0 / 5 ★ (134)

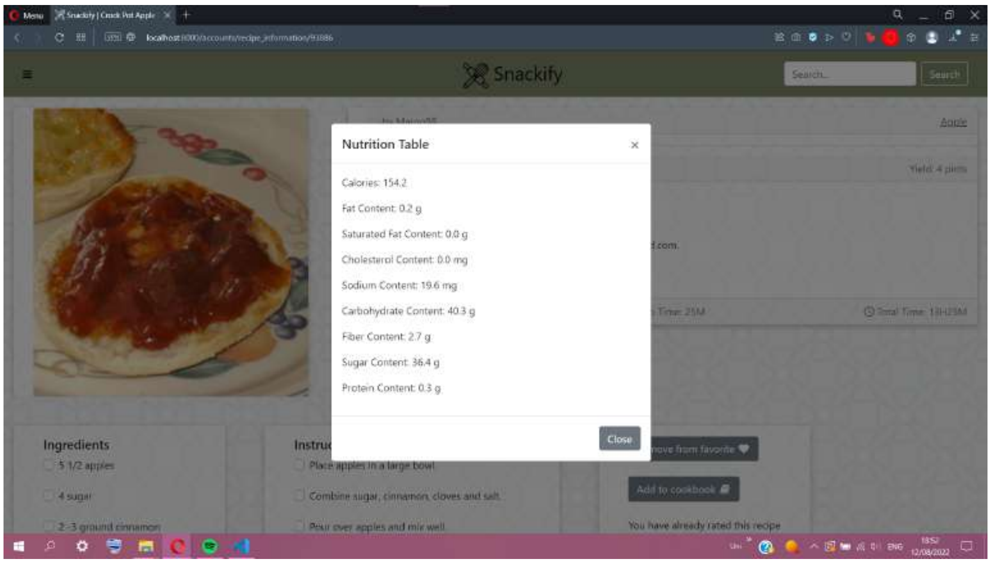
Make and share this Crock Pot Apple Butter recipe from Food.com.

Cook Time: 13H Prep Time: 25M Total Time: 13H25M

Figure 4.16: Recipe information.

At the top of the card, we have the recipe author's username and the recipe category, with links that redirect to the user's profile and the recipes of the specific category. In the row below, we can find the recipe's servings and the yield (in this example, the servings are 32, and the recipe yields 4 pints of apple butter). Next is the recipe's name with a larger text size to catch the eye of the user immediately. In this area, the user can also find the overall rating of the recipe, the number of reviews used to create this rating, along with a description of it provided by the user. Finally, as a footer, we provide the time the recipe needs to prepare and cook and the total time a user will need to make it.

The last thing we can find is the nutrition table 4.17 for each recipe.



Nutrition Table

Calories: 154.2

Fat Content: 0.2 g

Saturated Fat Content: 0.0 g

Cholesterol Content: 0.0 mg

Sodium Content: 19.6 mg

Carbohydrate Content: 40.3 g

Fiber Content: 2.7 g

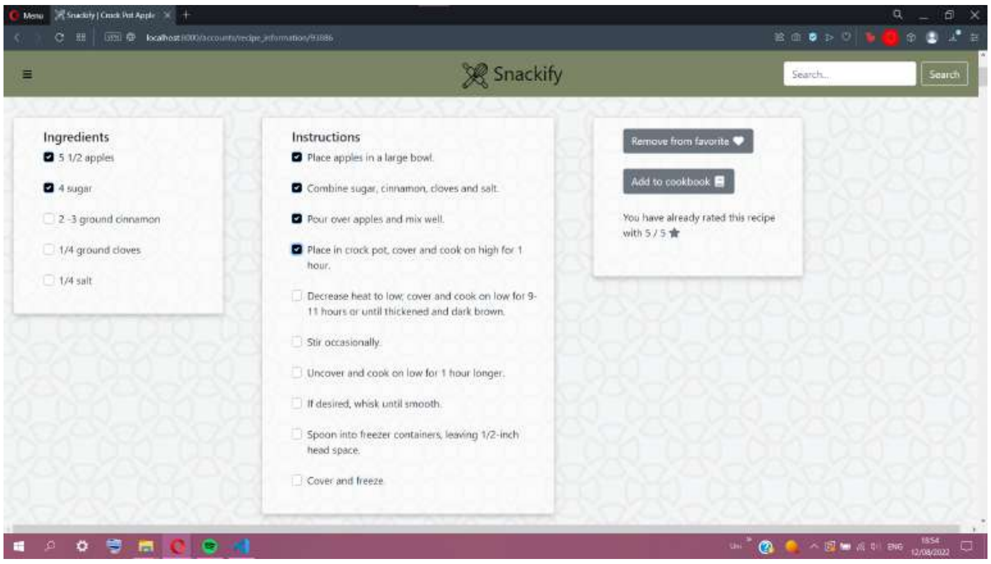
Sugar Content: 36.4 g

Protein Content: 0.3 g

Figure 4.17: Nutrition table.

From there, a user can be informed about the calories, fat content, calories, and other information they may want to know. The nutrition information is enclosed inside a modal pop-up, ensuring that users who do not want to know are not overwhelmed with more data than necessary.

Closing the modal and scrolling further in the page, the user finds space dedicated to recipe making 4.18.



Ingredients

- [x] 5 1/2 apples
- [x] 4 sugar
- [ ] 2 -3 ground cinnamon
- [ ] 1/4 ground cloves
- [ ] 1/4 salt

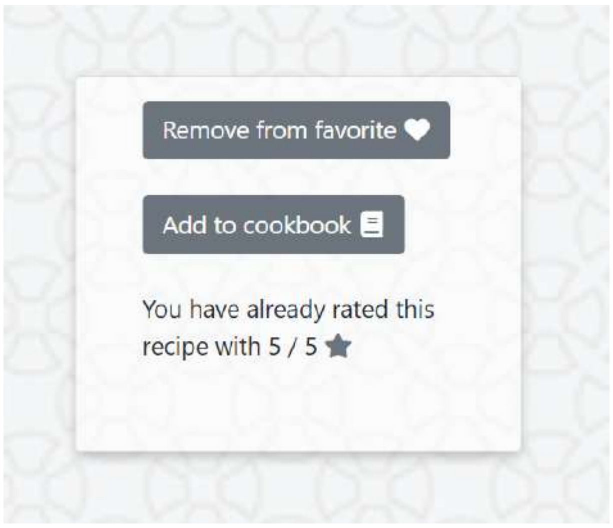
Instructions

- [x] Place apples in a large bowl.
- [x] Combine sugar, cinnamon, cloves and salt.
- [x] Pour over apples and mix well.
- [x] Place in crock pot, cover and cook on high for 1 hour.
- [ ] Decrease heat to low; cover and cook on low for 9-11 hours or until thickened and dark brown.
- [ ] Stir occasionally.
- [ ] Uncover and cook on low for 1 hour longer.
- [ ] If desired, whisk until smooth.
- [ ] Spoon into freezer containers, leaving 1/2-inch head space.
- [ ] Cover and freeze.

You have already rated this recipe with 5 / 5 ★

Figure 4.18: Recipe ingredient and instruction list.

On the left card is a list of the ingredients with their corresponding amount and a list of instructions broken down into steps. Next to each ingredient and instruction step is a clickable checkbox that is helpful when making a complicated recipe and does not want to miss a step or forget an ingredient.



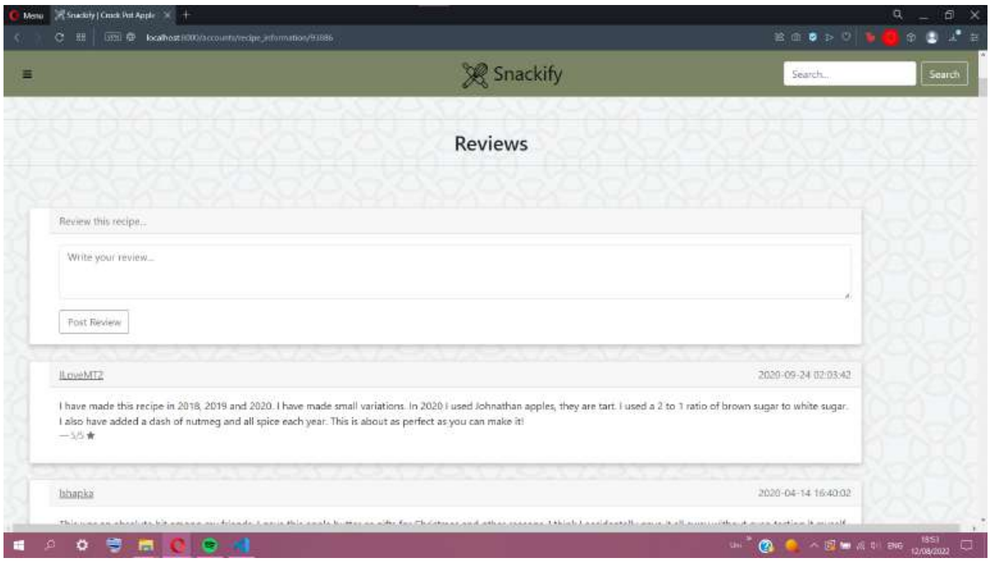
You have already rated this recipe with 5 / 5

Figure 4.19: Rating and action buttons.

On the right side, we find a couple of buttons and a form 4.19. The two buttons are to add the recipe to the user's favourite recipes or their cookbook (or both) accordingly. The favourite button is for the recipes the user has made and loved and wants others to know this recipe was good. The cookbook button is for the recipes the user liked and would like to make again or for new recipes they want to try. The rating form is with a star system out of five (5), with:

- ⋆ ⟶ the recipe was awful.
- ⋆⋆ ⟶ the recipe was bad.
- ⋆ ⋆ ⋆ ⟶ the recipe was ok.
- ⋆ ⋆ ⋆⋆ ⟶ the recipe was good.
- ⋆ ⋆ ⋆ ⋆ ⋆ ⟶ the recipe was awesome.

Once the user has rated a recipe, a message appears that they have already rated this recipe. The star rating appears as well.



Reviews

Review this recipe...

Write your review...

Post Review

ILoveMT2 2020-09-24 02:03:42

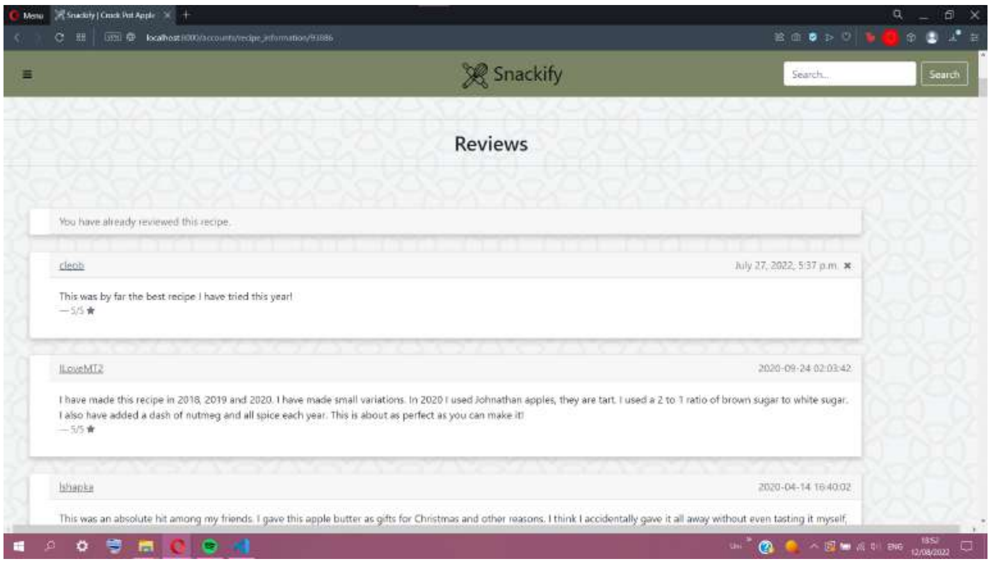
I have made this recipe in 2018, 2019 and 2020. I have made small variations. In 2020 I used Johnathan apples, they are tart. I used a 2 to 1 ratio of brown sugar to white sugar. I also have added a dash of nutmeg and all spice each year. This is about as perfect as you can make it!
— 5/5 ★

lshapka 2020-04-14 16:40:02

Figure 4.20: Recipe reviews.

At the bottom of the page, the user can find all the available reviews from other users that have tried the specific recipe 4.20. At the top left of each review is the author's username with a link to view their profile.

If the user has not written a review for this recipe, a box prompting to leave a review appears at the top.



Reviews

You have already reviewed this recipe.

cleob July 27, 2022, 5:37 p.m. ✖

This was by far the best recipe I have tried this year!
— 5/5 ★

ILoveMT2 2020-09-24 02:03:42

I have made this recipe in 2018, 2019 and 2020. I have made small variations. In 2020 I used Johnathan apples, they are tart. I used a 2 to 1 ratio of brown sugar to white sugar. I also have added a dash of nutmeg and all spice each year. This is about as perfect as you can make it!
— 5/5 ★

lshapka 2020-04-14 16:40:02

This was an absolute hit among my friends. I gave this apple butter as gifts for Christmas and other reasons. I think I accidentally gave it all away without even tasting it myself,

Figure 4.21: User review.

On the other hand, if the user has left a review, a message that the user has already left a review (and can not leave another) appears at the top right below the user's review 4.21. Next, their username appears at the top left, with the link to their profile. Finally, an 'x' button at the top right, next to the date, deletes the review, permitting the user to write another.

4.2.6 Recommended Recipes

In this page, the user can find their personalised recommendations 4.22.

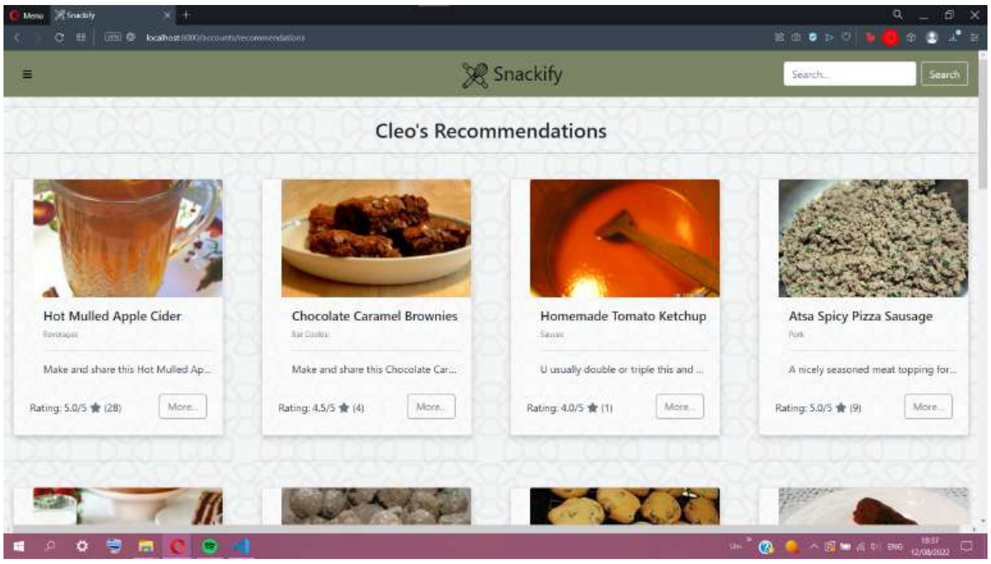


Figure 4.22: User Recommendations.

These recommendations are based on the recipes the user placed in their favourites, randomly selecting twenty (20) to present to the user. We discuss more the process of generating the recommendations below. 4.3

4.2.7 Categories

In the side navigation panel, when the user clicks on the ‘Categories’ button, a drop-down panel appears 4.23. The user has the choice among three popular recipe categories (‘Dessert’, ‘Breakfast’ and ‘Lunch/Snacks’) or clicking the ‘More...’ button,

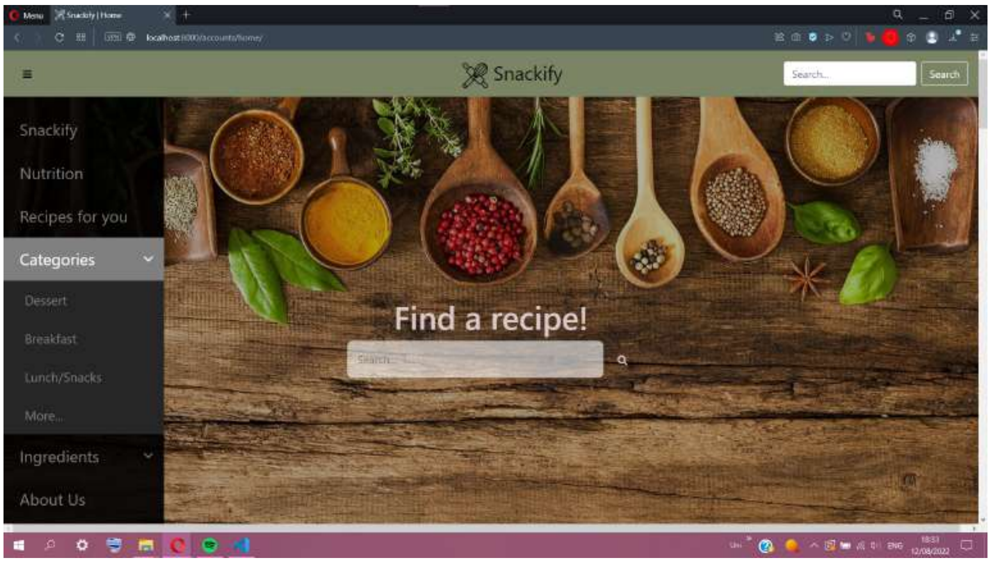


Figure 4.23: Categories side panel.

which redirects the user to a page with all of the categories available.

On the categories page, we can see all of the available ones separately 4.24, and a search bar.

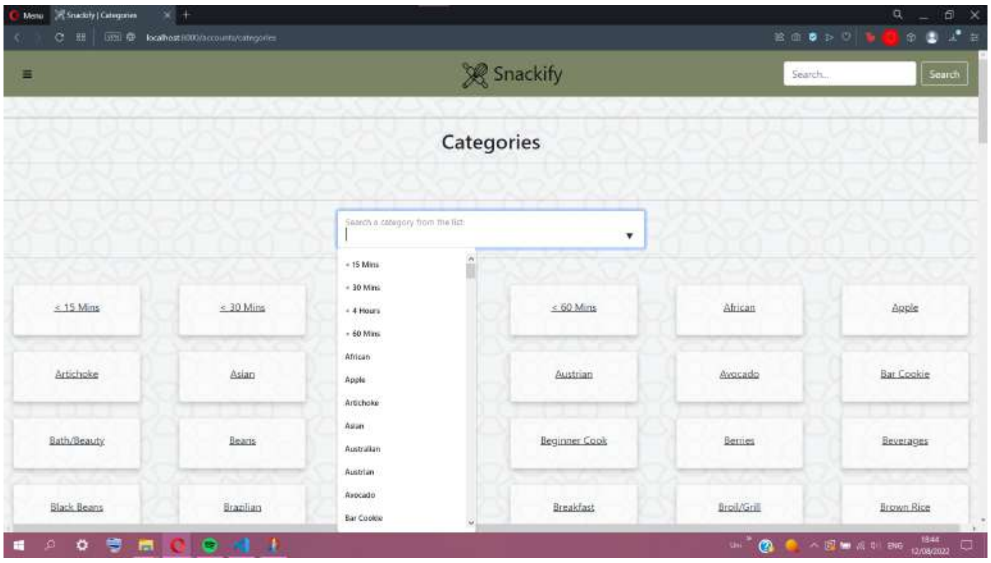


Figure 4.24: Categories.

The search bar contains all category names, and as the user types, it limits the results only to show the ones containing the input. So, for example, if the user writes ‘ag’ into the search bar, the results are limited to ‘Beverages’, ‘Punch Beverage’ and ‘Spaghetti’ 4.25.

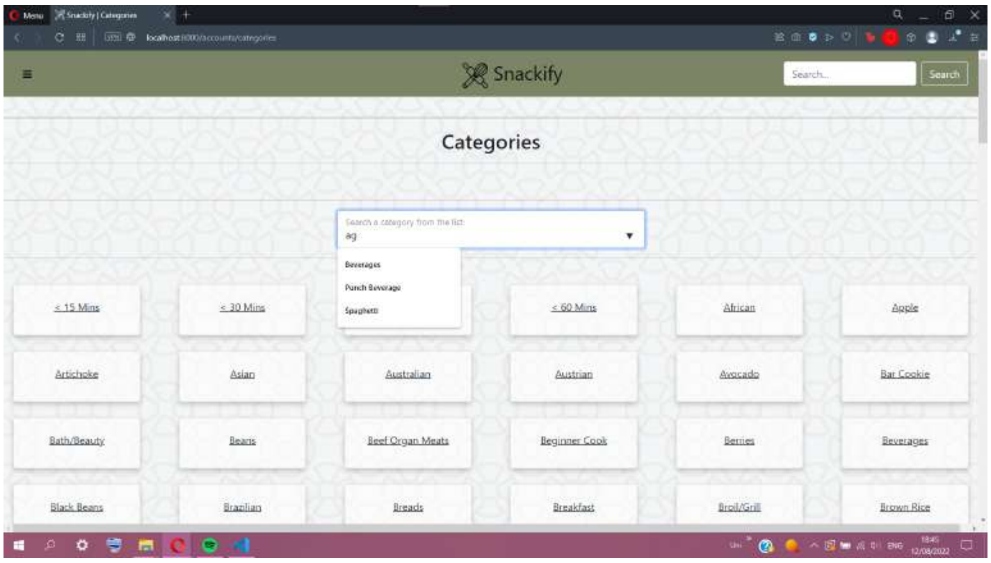


Figure 4.25: Categories limiting search.

After writing or selecting the category the user wants, an 'enter' from the keyboard redirects to the result page.

For instance, after selecting the category '< 15 Mins', we can see the recipes tagged as such 4.26. The number next to the category name inside the parenthesis indicates the number of recipes in the category. Below we can see the first row of results.

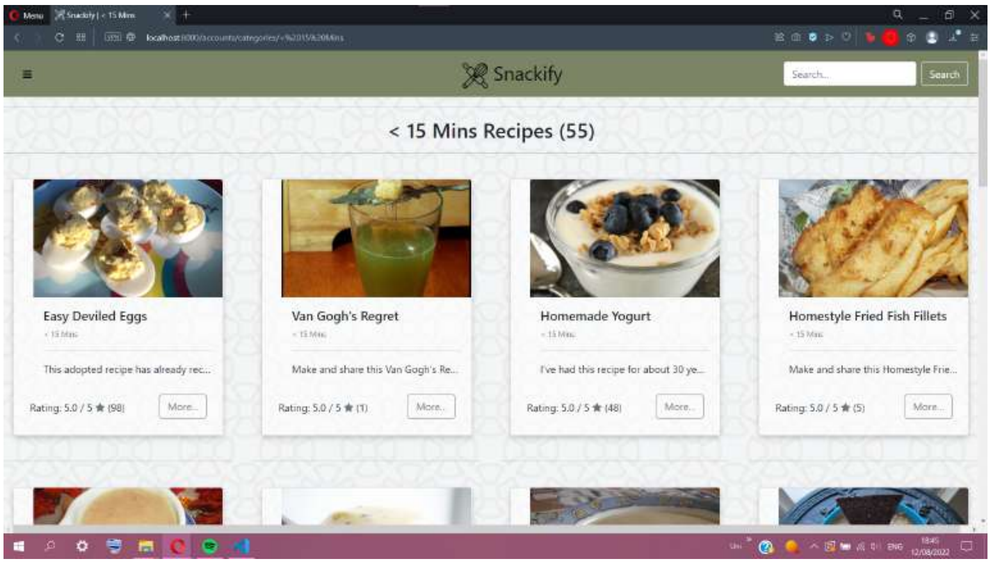
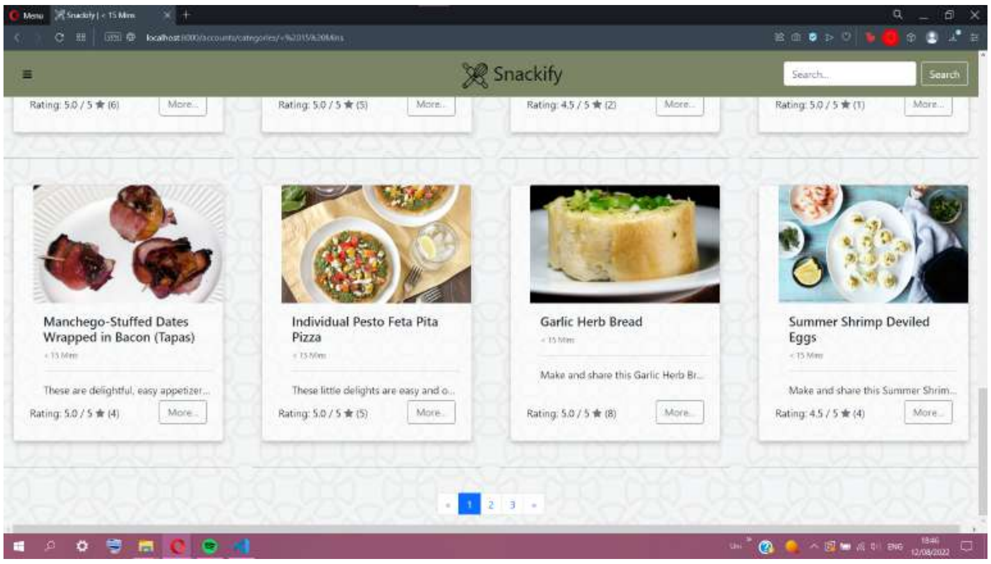


Figure 4.26: Category Recipes.

Scrolling to the bottom, we find the pagination 4.27. From there, the user can move between all the available recipes in a manner that is not overly confusing with too many results on each page. Every paginator page holds 20 results, so in our example, the 55 recipes in the

‘< 15 Mins’ category are separated into three pages, where the third page holds the remaining 15 results instead of 20.



Snackify
Search...
Search
Manchego-Stuffed Dates Wrapped in Bacon (Tapas)
These are delightful, easy appetizer...
Rating: 5.0 / 5 ★ (4)
More...
Individual Pesto Feta Pita Pizza
These little delights are easy and o...
Rating: 5.0 / 5 ★ (5)
Garlic Herb Bread
Make and share this Garlic Herb Br...
Rating: 5.0 / 5 ★ (8)
Summer Shrimp Deviled Eggs
Make and share this Summer Shrim...
Rating: 4.5 / 5 ★ (4)
« 1 2 3 »

Figure 4.27: Categories pagination.

4.2.8 Ingredients

In the side navigation panel, when the user clicks on the ‘Ingredients’ button, a drop-down panel appears 4.28. The user has the choice among three popular recipe ingredients (‘Chicken’, ‘Chocolate’ and ‘Potatoes’) or clicking the ‘More...’ button,

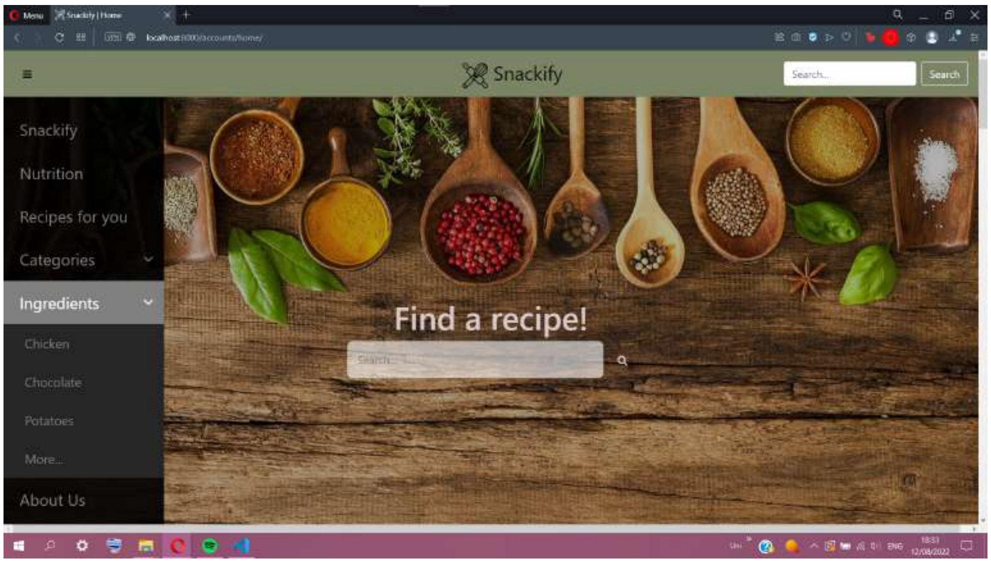


Figure 4.28: Ingredients side panel.

which redirects the user to a page with a search bar 4.29 and a few recommended ingredients.

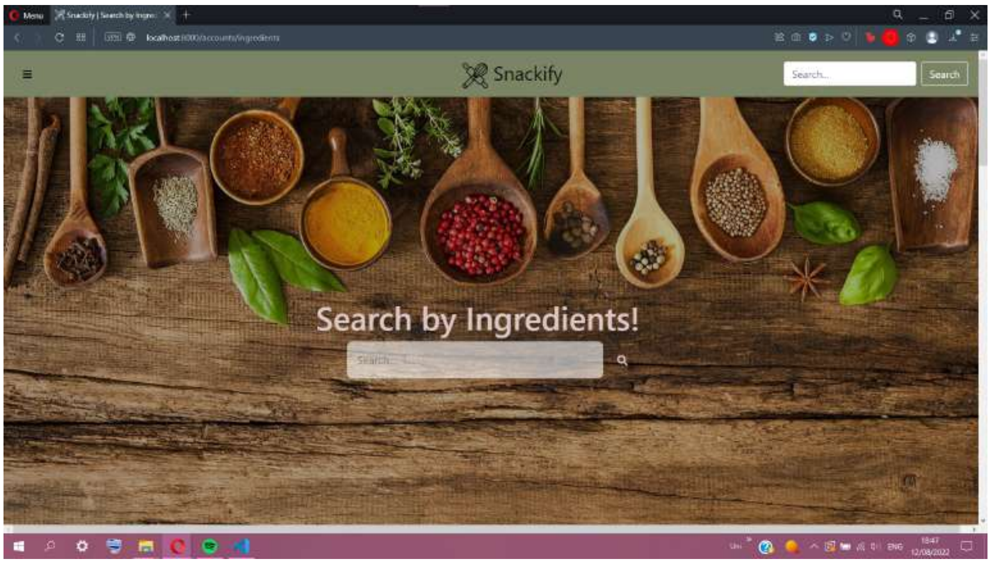
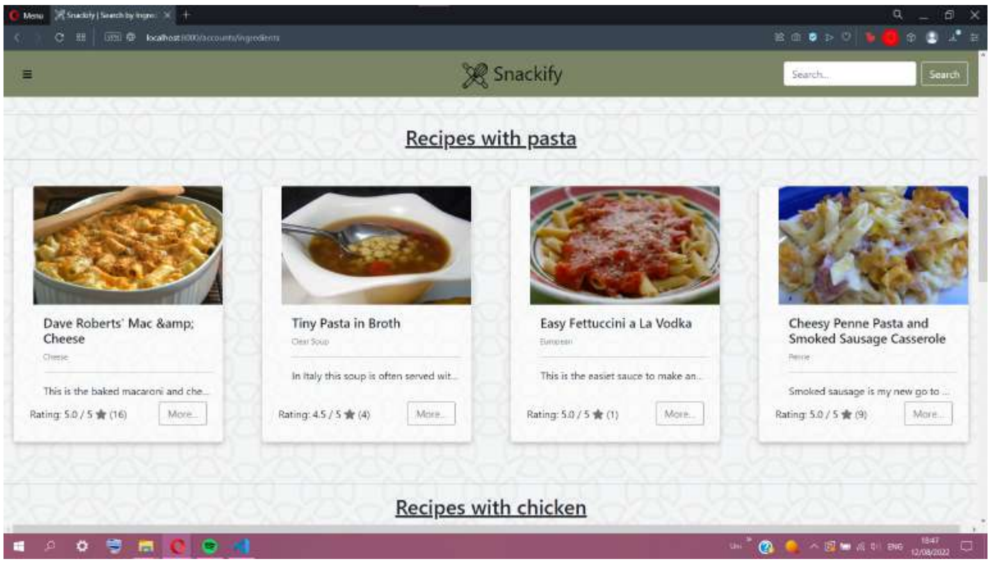

Menu
Snackify | Search by Ingre
localhost:8000/accounts/ingredients
Snackify
Search...
Search
Search by Ingredients!
Search...

Figure 4.29: Search by Ingredients.

As they scroll down, they encounter a few recipes from four recipe ingredients that we think are the most common for people to search for. Such as:

- pasta


Menu
Snackify | Search by Ingre
localhost:8000/accounts/ingredients
Snackify
Search...
Search
More...
More...
More...
More...

Recipes with pasta

Dave Roberts' Mac &amp; Cheese
Cheese
This is the baked macaroni and che...
Rating: 5.0 / 5 ★ (16)

Tiny Pasta in Broth
Clear Soup
In Italy this soup is often served wit...
Rating: 4.5 / 5 ★ (4)

Easy Fettuccini a La Vodka
European
This is the easiet sauce to make an...
Rating: 5.0 / 5 ★ (1)

Cheesy Penne Pasta and Smoked Sausage Casserole
Penne
Smoked sausage is my new go to ...
Rating: 5.0 / 5 ★ (9)

Recipes with chicken

Figure 4.30: Recipes with ‘pasta’ as an ingredient.

- chicken

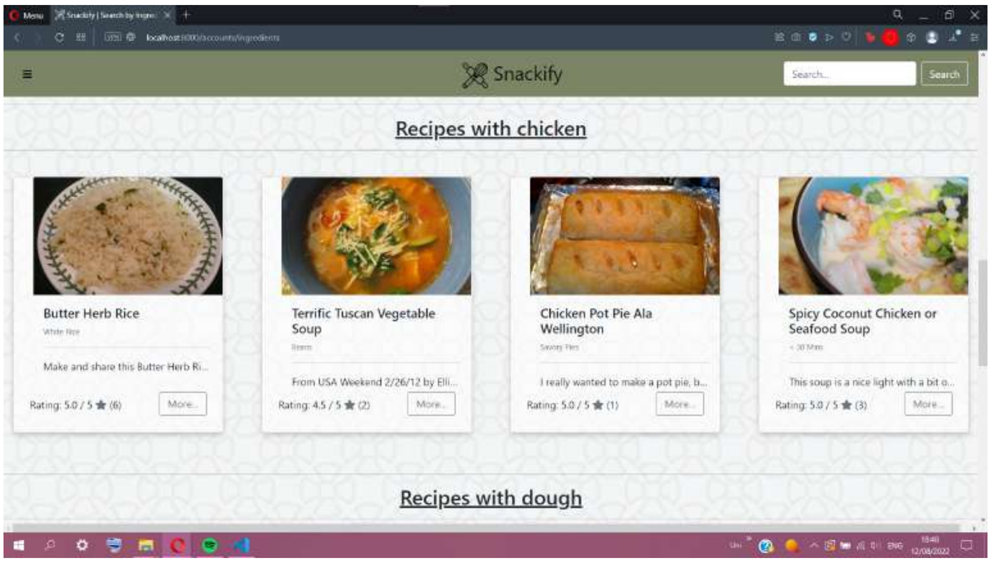

Snackify
Search
Recipes with chicken
Butter Herb Rice
Make and share this Butter Herb Ri...
Rating: 5.0 / 5 ★ (6)
More...
Terrific Tuscan Vegetable Soup
From USA Weekend 2/26/12 by Elli...
Rating: 4.5 / 5 ★ (2)
More...
Chicken Pot Pie Ala Wellington
I really wanted to make a pot pie, b...
Rating: 5.0 / 5 ★ (1)
More...
Spicy Coconut Chicken or Seafood Soup
This soup is a nice light with a bit o...
Rating: 5.0 / 5 ★ (3)
More...
Recipes with dough

Figure 4.31: Recipes with 'chicken' as an ingredient.

- dough

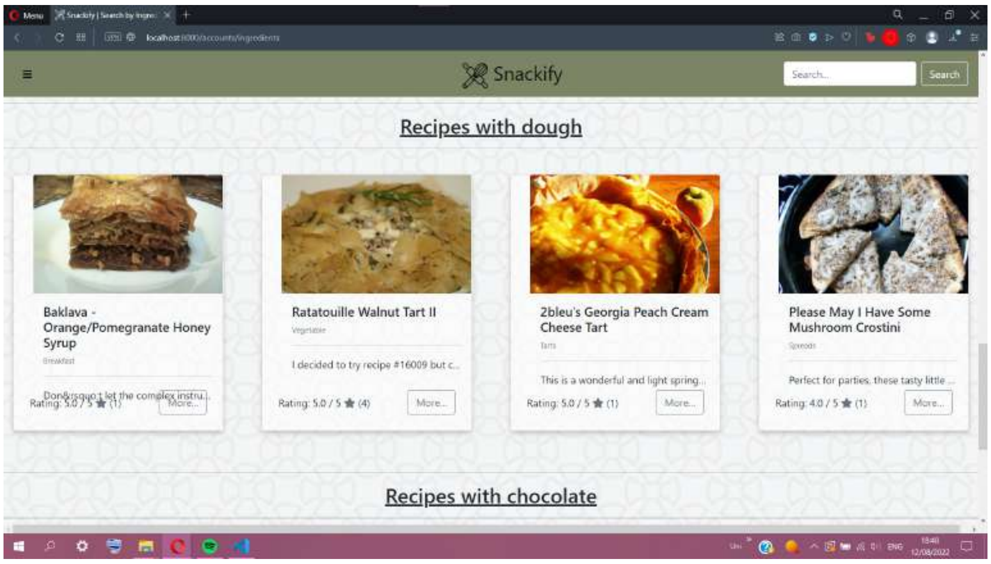

Snackify
Search
Recipes with dough
Baklava - Orange/Pomegranate Honey Syrup
Rating: 5.0 / 5 ★ (1)
More...
Ratatouille Walnut Tart II
I decided to try recipe #16009 but c...
Rating: 5.0 / 5 ★ (4)
More...
2bleu's Georgia Peach Cream Cheese Tart
This is a wonderful and light spring...
Rating: 5.0 / 5 ★ (1)
More...
Please May I Have Some Mushroom Crostini
Perfect for parties, these tasty little ...
Rating: 4.0 / 5 ★ (1)
More...
Recipes with chocolate

Figure 4.32: Recipes with 'dough' as an ingredient.

- chocolate

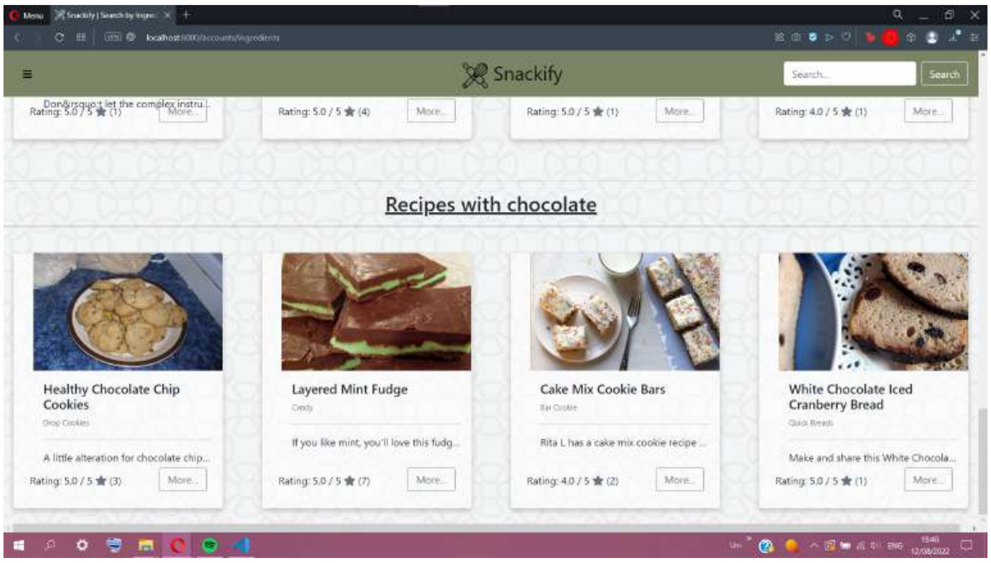

Snackify
Search...
Search
Recipes with chocolate
Healthy Chocolate Chip Cookies
Drop Cookies
A little alteration for chocolate chip...
Rating: 5.0 / 5 ★ (3)
Layered Mint Fudge
Candy
If you like mint, you'll love this fudg...
Rating: 5.0 / 5 ★ (7)
Cake Mix Cookie Bars
Bar Cookie
Rita L has a cake mix cookie recipe ...
Rating: 4.0 / 5 ★ (2)
White Chocolate Iced Cranberry Bread
Quick Breads
Make and share this White Chocola...
Rating: 5.0 / 5 ★ (1)
More...

Figure 4.33: Recipes with ‘chocolate’ as an ingredient.

Returning to the search bar at the top of the page 4.29, there are two ways in which the user can search. The first one, and the most obvious, is to write one ingredient they want to have in a recipe, and the result page holds every recipe with this ingredient 4.34.

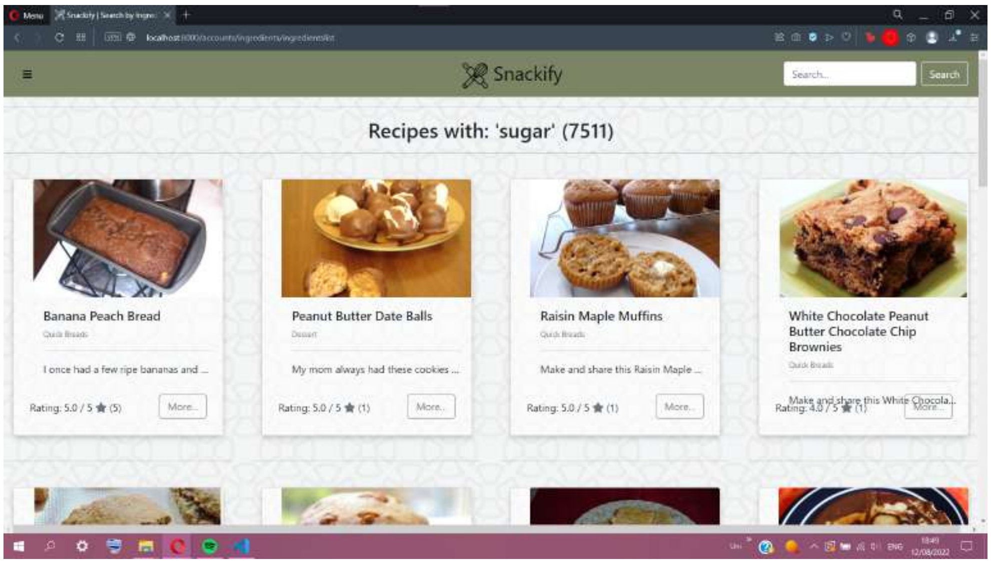

Snackify
Search...
Search
Recipes with: 'sugar' (7511)
Banana Peach Bread
Quick Breads
I once had a few ripe bananas and ...
Rating: 5.0 / 5 ★ (5)
Peanut Butter Date Balls
Dessert
My mom always had these cookies ...
Rating: 5.0 / 5 ★ (1)
Raisin Maple Muffins
Quick Breads
Make and share this Raisin Maple ...
Rating: 5.0 / 5 ★ (1)
White Chocolate Peanut Butter Chocolate Chip Brownies
Quick Breads
Make and share this White Chocola...
Rating: 4.0 / 5 ★ (1)
More...

Figure 4.34: Recipes with only ‘sugar’ as an ingredient.

The second method is for the user to put more than one ingredient in the search bar, separated by commas, and the result page holds every recipe with all the ingredients 4.35.

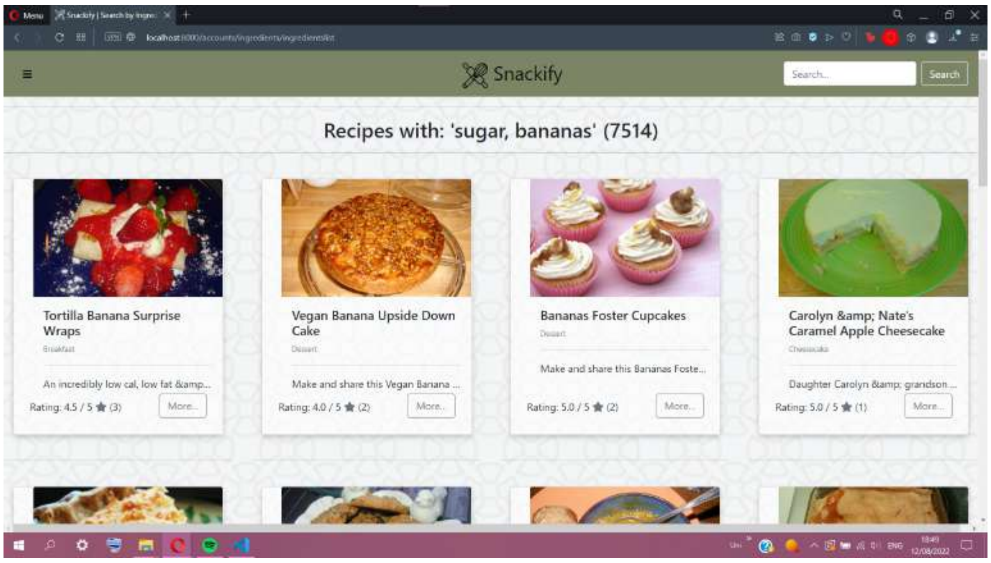

Snackify
Search...
Search
Recipes with: 'sugar, bananas' (7514)
Tortilla Banana Surprise Wraps
Breakfast
An incredibly low cal, low fat &amp...
Rating: 4.5 / 5 ★ (3)
More...
Vegan Banana Upside Down Cake
Dessert
Make and share this Vegan Banana ...
Rating: 4.0 / 5 ★ (2)
More...
Bananas Foster Cupcakes
Dessert
Make and share this Bananas Foste...
Rating: 5.0 / 5 ★ (2)
More...
Carolyn &amp; Nate's Caramel Apple Cheesecake
Cheesecake
Daughter Carolyn &amp; grandson ...
Rating: 5.0 / 5 ★ (1)
More...

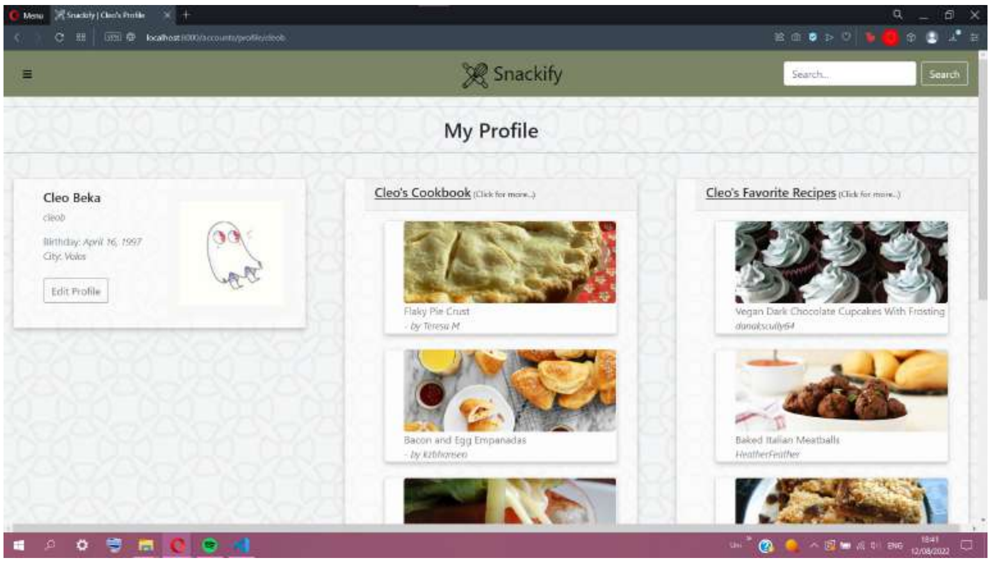
Figure 4.35: Recipes with ‘sugar’ and ‘bananas’ as ingredients.

Of course, under each result page, there is a paginator to avoid an overfilled page with results that confuse the user.

4.2.9 Profile

In the side navigation panel, when the user clicks on the ‘Profile’ button, it redirects to the current user’s profile in session 4.36. On the left side of the screen, the user can see basic information about themselves, such as their first and last name as well as their username exactly below, as they filled them out during the sign-up process, as well as a default profile picture.

Underneath that, in a newly created profile is a dash (‘-’) where the user’s birthday and the city they live in should be. That is because these pieces of information were not in the sign-up form, as some users might not want to disclose that.


Snackify
Search...
Search
My Profile

Cleo Beka
cleob
Birthday: April 16, 1997
City: Volos

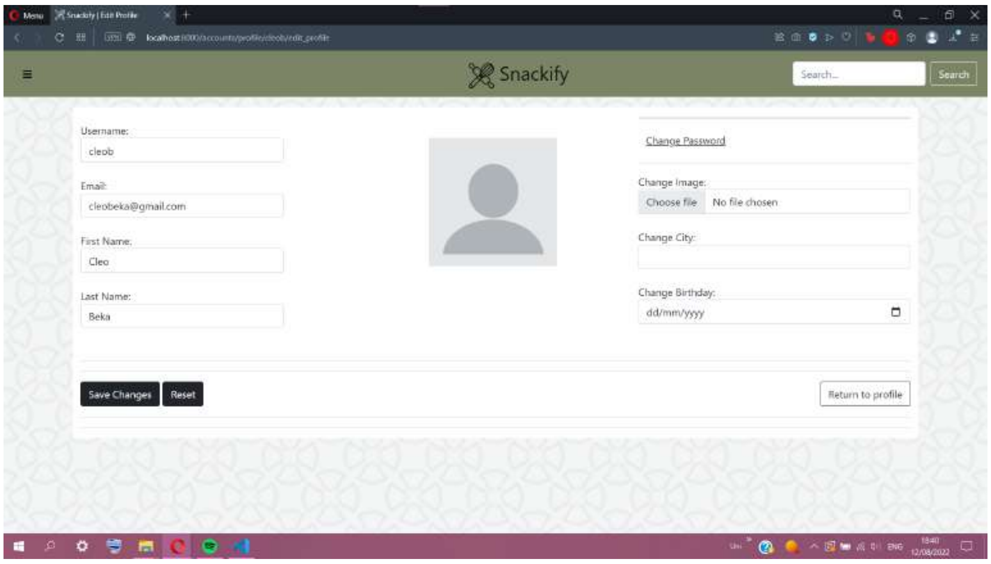
Edit Profile
Cleo's Cookbook (Click for more...)
Cleo's Favorite Recipes (Click for more...)

Flaky Pie Crust - by Teresa M
Bacon and Egg Empanadas - by kibhansen
Vegan Dark Chocolate Cupcakes With Frosting donutsculty64
Baked Italian Meatballs HeatherFeather

Figure 4.36: User profile.

4.2.9.1 Edit Profile

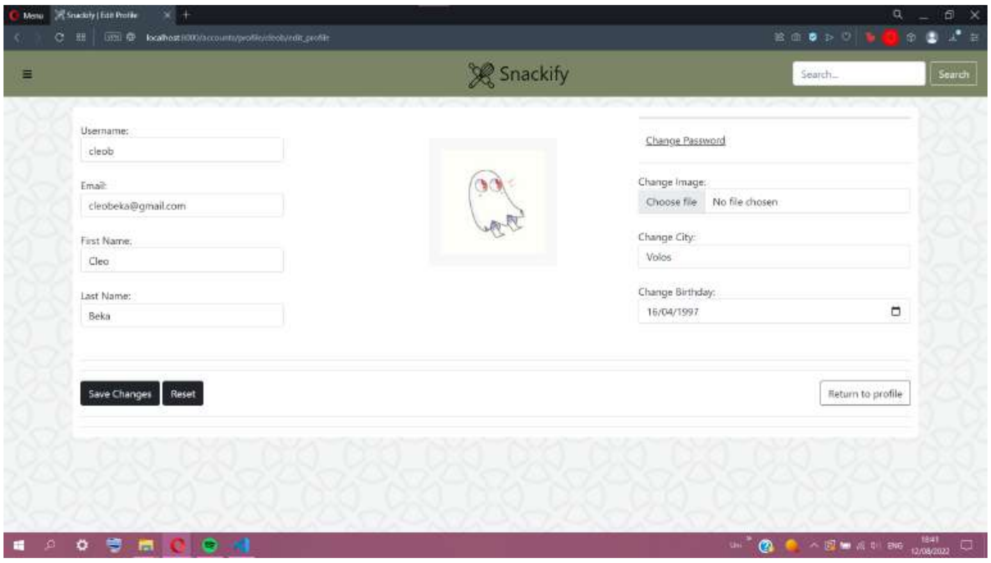
In the event that the user wants to fill them in, or they wrote something wrong while signing up, at the bottom of the card, there is a button named ‘Edit Profile’.



Username: cleob
Email: cleobeka@gmail.com
First Name: Cleo
Last Name: Beka
Change Image: No file chosen
Change City:
Change Birthday: dd/mm/yyyy

Figure 4.37: Edit profile.

After clicking on it, the user is redirected to the page 4.37 where they can edit their information and add their birthday, city, and even a profile picture of their choosing.



Username:
cleob

Email:
cleobeka@gmail.com

First Name:
Cleo

Last Name:
Beka

Change Password

Change Image:
Choose file No file chosen

Change City:
Volos

Change Birthday:
16/04/1997

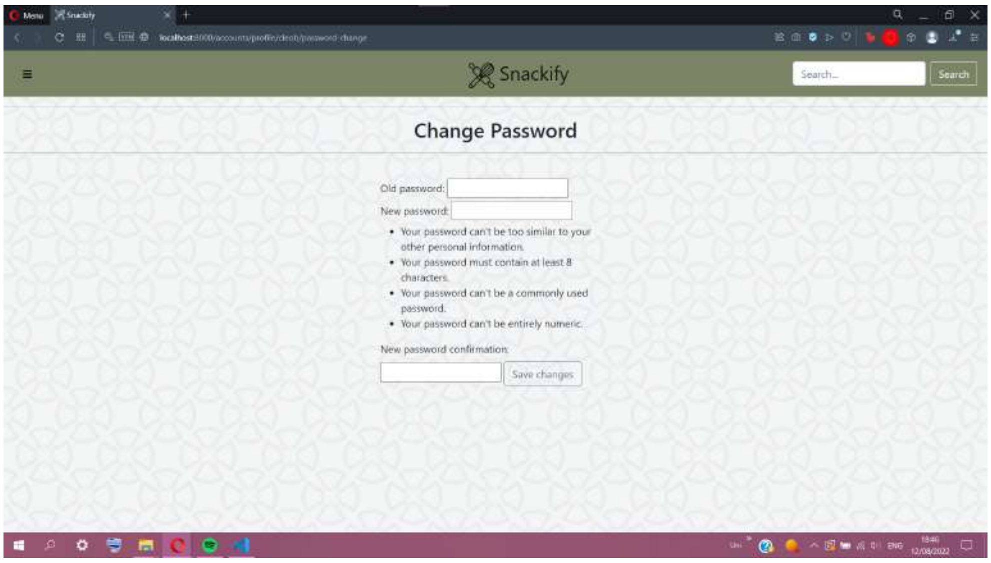
Save Changes Reset Return to profile

Figure 4.38: Edit profile page after filling out the information.

In the event of a mistype during the editing process. Before clicking ‘Save Changes’, the user can click the ‘Reset’ button, effectively returning the information to the last save state.

4.2.9.2 Change Password

If the user thinks their password is too easy or wants a different password, they can change it through the ‘Change Password’ form 4.39.



Change Password

Old password:
New password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation:

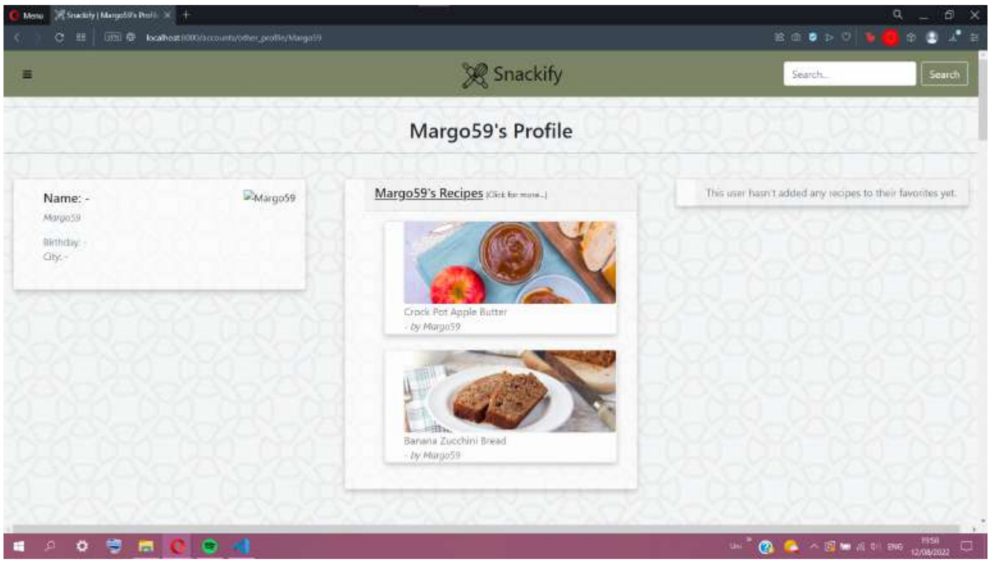
Save changes

Figure 4.39: Change password.

4.2.9.3 Other Users' Profile

In the event a user wants to visit another user's profile, all they have to do is click on their username through a review or a recipe. After that, the user is redirected to a page similar to their profile 4.40.

On the card on the left is the user's personal information that the profile they visit lacks, of course, the 'Edit Profile' button. The middle card holds the recipes the user has uploaded on the site 4.42 instead of the cookbook (if there are none, an error message appears informing us that the user has not uploaded any recipes yet). Finally, the right card shows the recipes the user has favourited.



This user hasn't added any recipes to their favorites yet.

Figure 4.40: Other users' profiles.

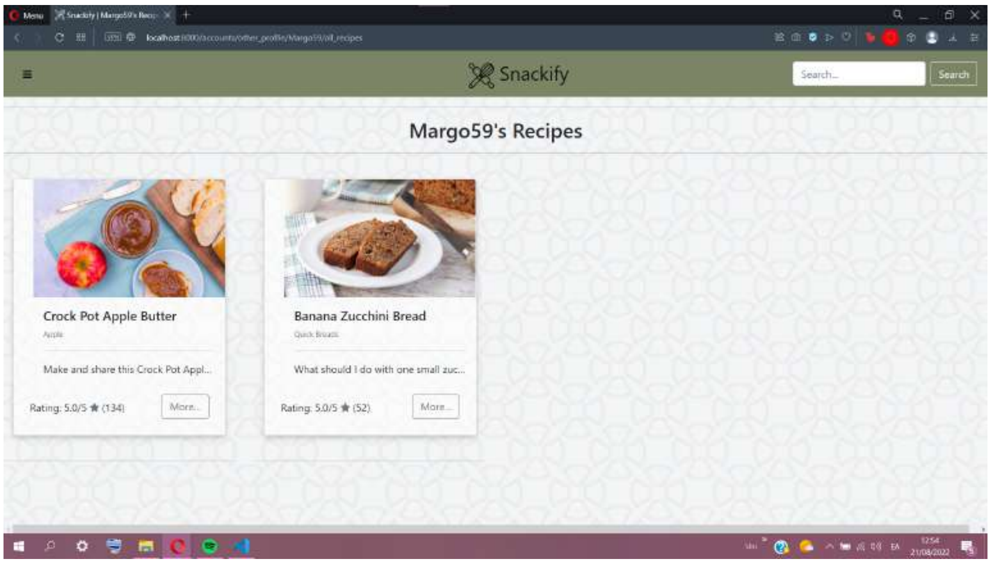
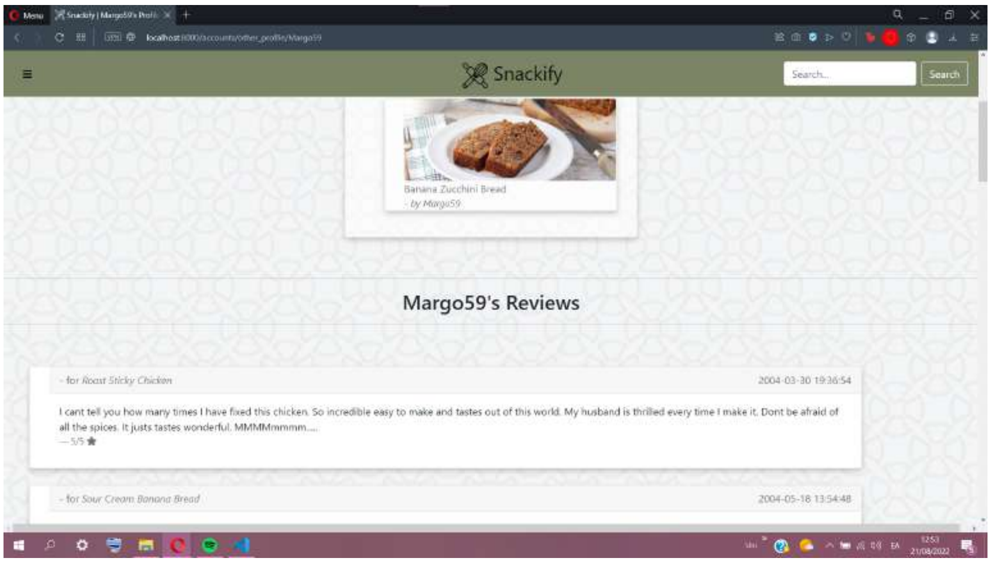


Figure 4.41: Other users' profiles.

At the bottom of the page, a list of the reviews that the user made appears, with the names of the recipes, as well as the rating and the review date 4.41.



- for *Roast Sticky Chicken* 2004-03-30 19:36:54

I cant tell you how many times I have fixed this chicken. So incredible easy to make and tastes out of this world. My husband is thrilled every time I make it. Dont be afraid of all the spices. It justs tastes wonderful. MMMMmmmm.....
— 5/5 ★

- for *Sour Cream Banana Bread* 2004-05-18 13:54:48

Figure 4.42: Other users' profiles.

4.2.10 Useful

In the side navigation panel, when the user clicks on the 'Useful' button, a drop-down panel appears 4.43. The user can choose among three useful converters ('Volume Converter', 'Length Converter' and 'Temperature Converter') that can help during cooking.

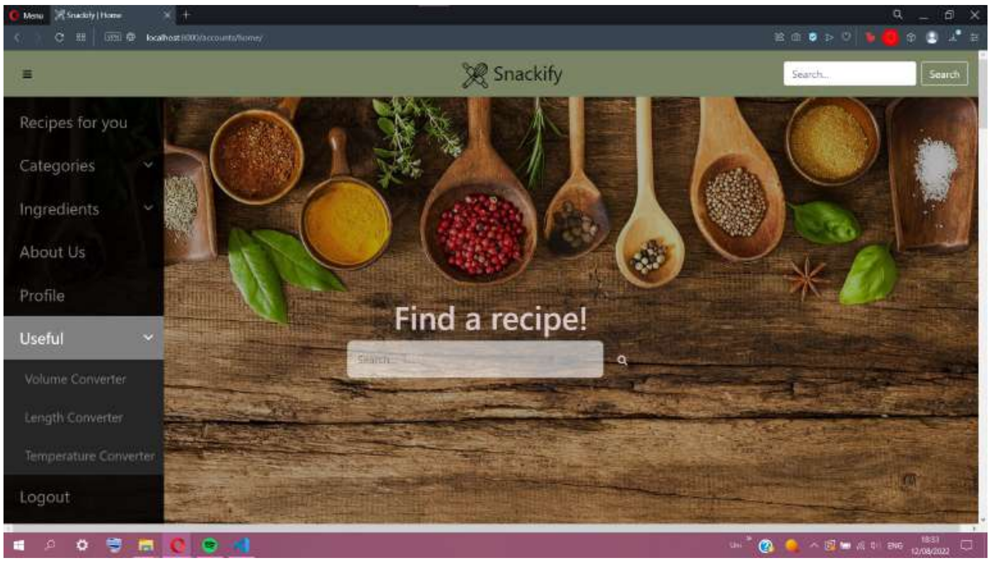
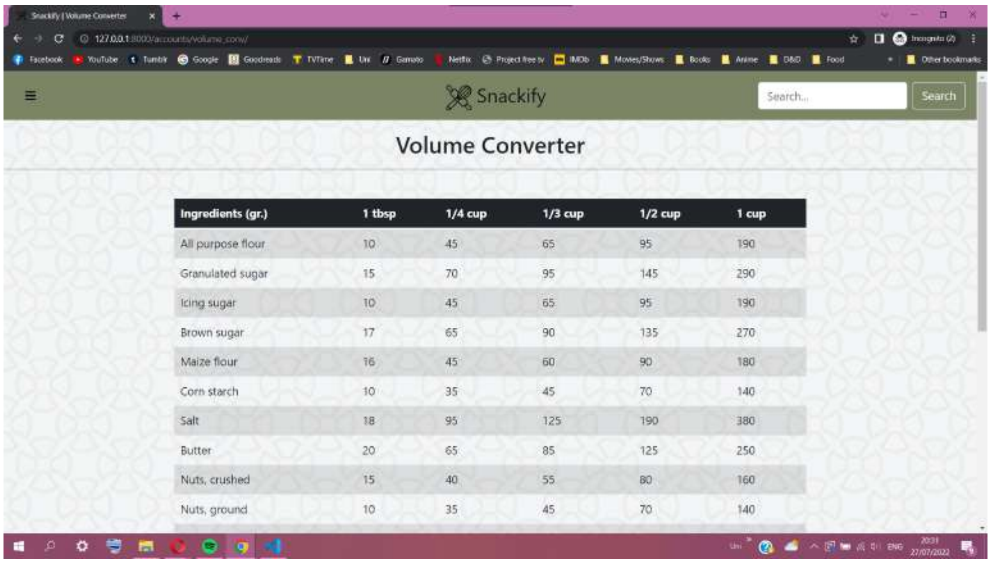


Figure 4.43: Useful side panel.

Let us look at them a little more closely.

- Volume Converter:



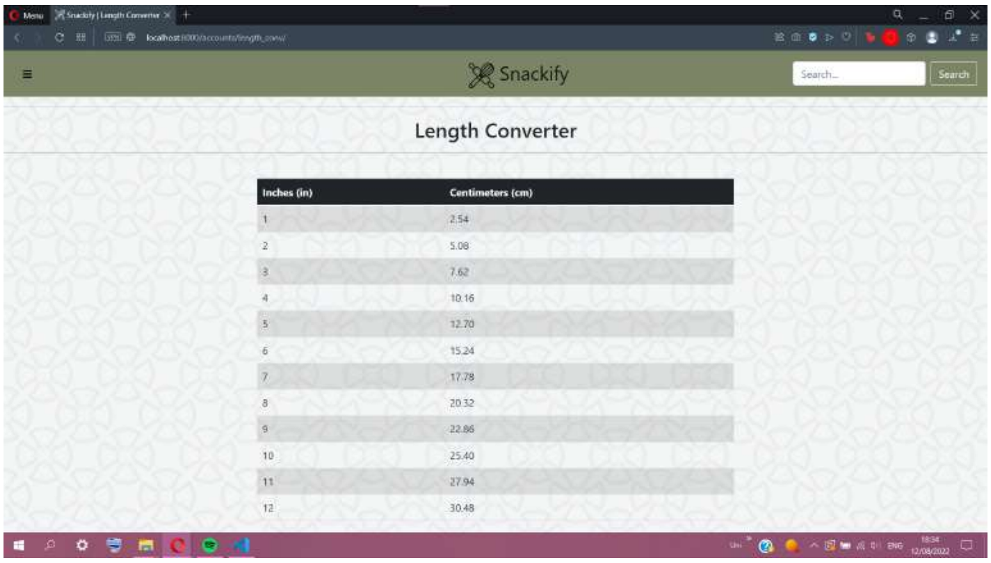
Volume Converter

Ingredients (gr.)	1 tbsp	1/4 cup	1/3 cup	1/2 cup	1 cup
All purpose flour	10	45	65	95	190
Granulated sugar	15	70	95	145	290
Icing sugar	10	45	65	95	190
Brown sugar	17	65	90	135	270
Maize flour	16	45	60	90	180
Corn starch	10	35	45	70	140
Salt	18	95	125	190	380
Butter	20	65	85	125	250
Nuts, crushed	15	40	55	80	160
Nuts, ground	10	35	45	70	140

Figure 4.44: Volume Converter page.

On the ‘Volume Converter’ page, the user can find a table with a list of widely used ingredients measured in grams and then, in each column, the breakdown of how many grams from each ingredient fills each of the measuring utensils. For example, for the ingredient ‘All-purpose flour’, 1 tbsp equals 10 grams of product, 1/4 of a cup equals 45 grams etc.

- Length Converter:



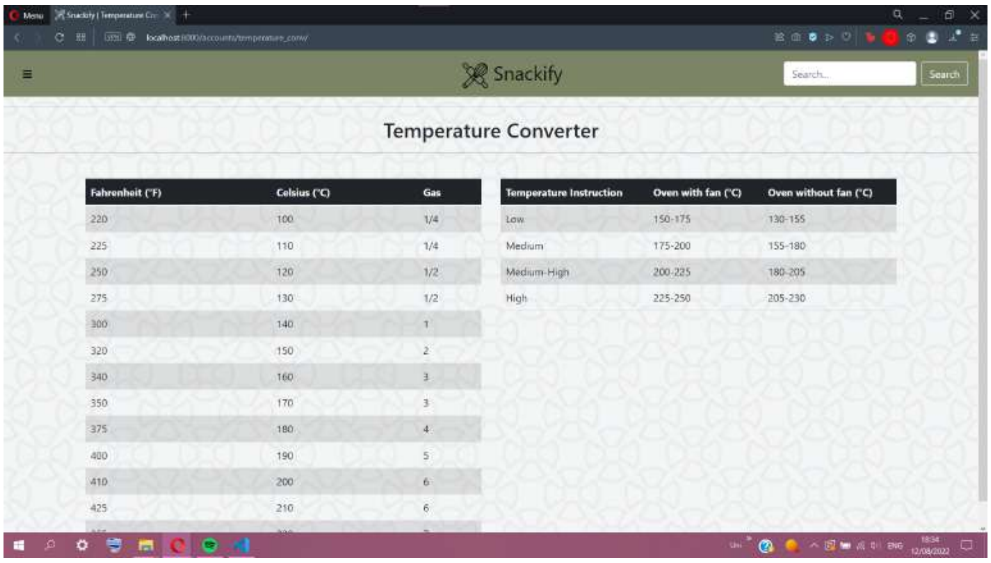
Length Converter

Inches (in)	Centimeters (cm)
1	2.54
2	5.08
3	7.62
4	10.16
5	12.70
6	15.24
7	17.78
8	20.32
9	22.86
10	25.40
11	27.94
12	30.48

Figure 4.45: Length Converter page.

On the 'Volume Converter' page, the user can find a table with a list of the conversion from inches 1-12 to centimetres and back. Those measurements are mainly used for the size of pans during baking.

- Temperature Converter:



Temperature Converter

Fahrenheit (°F)	Celsius (°C)	Gas
220	100	1/4
225	110	1/4
250	120	1/2
275	130	1/2
300	140	1
320	150	2
340	160	3
350	170	3
375	180	4
400	190	5
410	200	6
425	210	6

Temperature Instruction	Oven with fan (°C)	Oven without fan (°C)
Low	150-175	130-155
Medium	175-200	155-180
Medium-High	200-225	180-205
High	225-250	205-230

Figure 4.46: Temperature Converter page.

The user can find two tables with information on the 'Temperature Converter' page. The left table holds the conversion of the most used temperatures in a commercial oven with Fahrenheit or Celsius units and an equivalent for a gas indication. The table on the right has an estimated average temperature for ovens with a fan and ovens without a fan in Celsius units for when the recipe asks for a vague ‘Medium’ or ‘Medium-High’ temperature.

4.3 Back-End

4.3.1 Data and Storage

As mentioned, we have two (2) CSV-type files: ‘recipes’ and ‘reviews’. After we reduced the size of the ‘recipes’ dataset with the methods mentioned in 4.1, we stored them in the PostgreSQL database we created. Using Django to create our website, we have advantages like the Django authentication system, which handles both authentication and authorization. Briefly, authentication checks if a user is whom they claim to be, and authorization decides what an authenticated user is permitted to do. The term authentication is used to refer to both tasks.

The auth system consists of:

- Users
- Permissions: Binary (yes/no) flags determine whether a user can perform a specific task.
- Groups: A method of applying labels and permissions to more than one user.
- etc.

Our database consists of two types of tables. The first type is tables created by Django and its authentication system, and they are the following:

- auth_group
- auth_permission
- auth_group_permission
- auth_user
- auth_user_groups
- auth_user_user_permission

- django_admin_log
- django_content_type
- django_migrations
- django_session

The second type of table is those that we created to store our data, and they are the following:

- accounts_interactions: This table keeps our users' reviews and ratings.
- accounts_profile: This one has inside the profile of each user and their personal information. Every time a new user signs up, a new row is created at this table containing the data from the 'auth_user' table. Then, the particular user can edit the profile data and add more information about themselves.
- accounts_profile_cookbook: The cookbook of each user that contains the recipes they want to find easily and make them.
- accounts_profile_favorite: The favourite recipes of each user
- accounts_recipes_info: Here, we saved the 'recipes' dataset mentioned in 4.1, so we can have access to it without difficulty.
- accounts_reviews: The 'reviews' dataset mentioned in 4.1 is stored again to access it more easily.

For a better understanding of the database and its tables, we depicted it in an Entity Relationship Diagram (ERD) in the following figure:

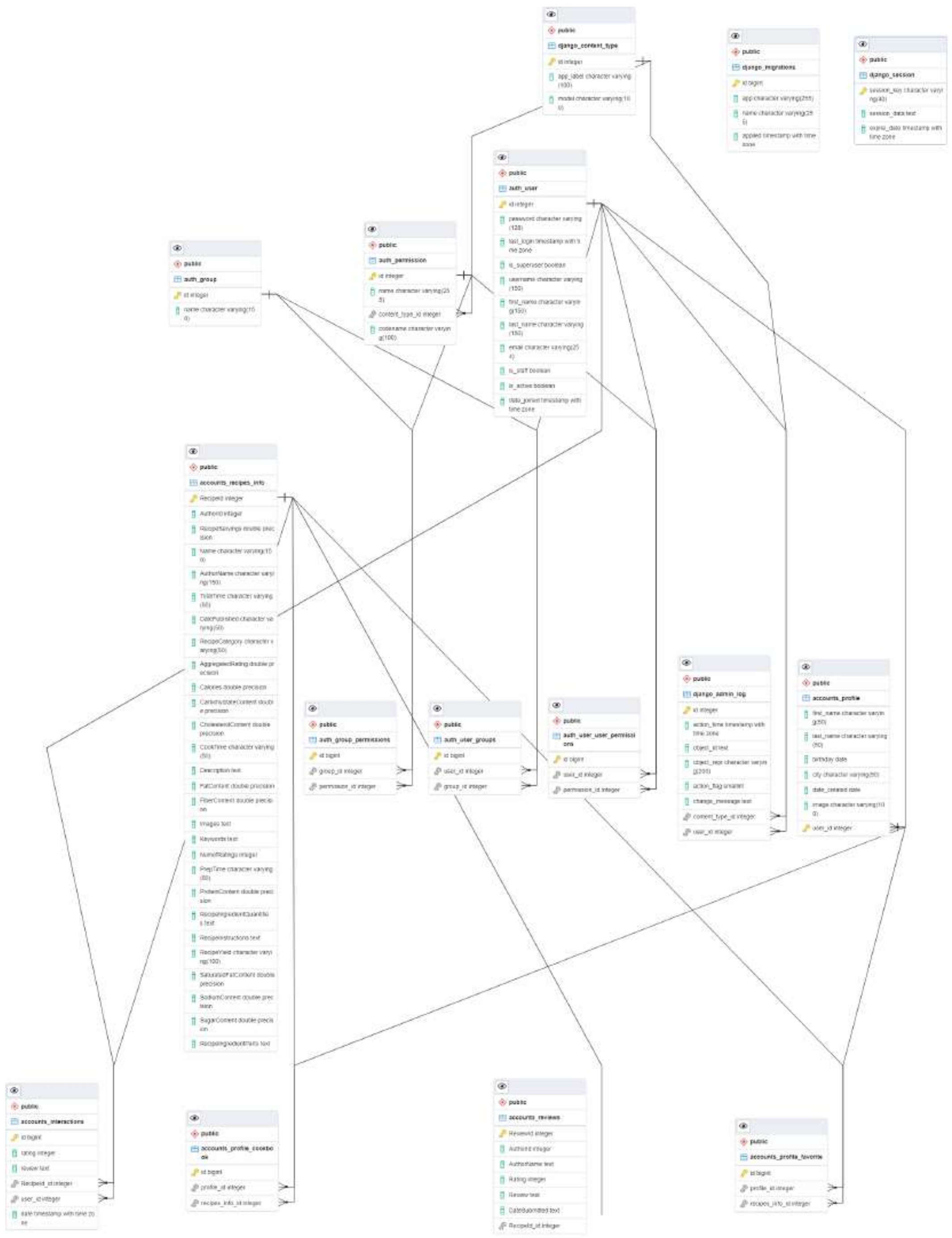


Figure 4.47: ERD

In Figure 4.47 there are the tables we mentioned above and the relationship between them. The types of relationship are the following:

- one-to-one: For example, a recipe can be written by only one user.
- one-to-many: For instance, a recipe can have many reviews.
- many-to-many: For example, a recipe can have many ingredients and an ingredient may be in many recipes.

According to the file 'models.py' which is explained with details in the next subsection, the tables of the database are created with the corresponding relationship between them.

4.3.2 Models

In Django, there is a python file named 'models', and every model class generated there has. As a result, the creation of a new table in the database. So, each of the following models is also a table in the database:

- interactions: It contains the following data:
  - RecipeId
  - UserId
  - Review
  - Rating
- recipes_info: Contains all the columns that the 'recipes' CSV file contains, mentioned in 4.1
- reviews: From the 'reviews' dataset mentioned in 4.1, this model has the same columns as it.
- profile: It contains the personal information of a user:
  - UserId
  - first_name
  - last_name

- birthday
- city
- date_created
- image

We created this database so new users can create a profile and have the opportunity to make their cookbook and favourite list and get recommendations for recipes they may like. Apart from this, we made models and saved the CSV files mentioned in 4.1, so we could have access to them more accessible and avoid reading the CSV files every time we needed them.

4.4 Recommendation Systems

There are three (3) recommendation systems: two Content-Based systems and one Collaborative Filtering system. One Content-Based and the Collaborative Filtering are combined to give results on one page.

4.4.1 Content-Based System I

The implementation of this Item Based Content Filtering system is based on correlation. Correlation describes the statistical relationship between two entities. By that, we mean it is how two variables move concerning one another. Correlation is a value between -1 and +1. However, correlation is not causation! There are three (3) types of correlations:

- **Positive Correlation :** A positive correlation is a value between zero (0.0) and one (1.0). A correlation of one (1.0) shows that if the first variable moves up, the second one will also increase. This relationship is not so strong if the correlation is less than one (1.0).

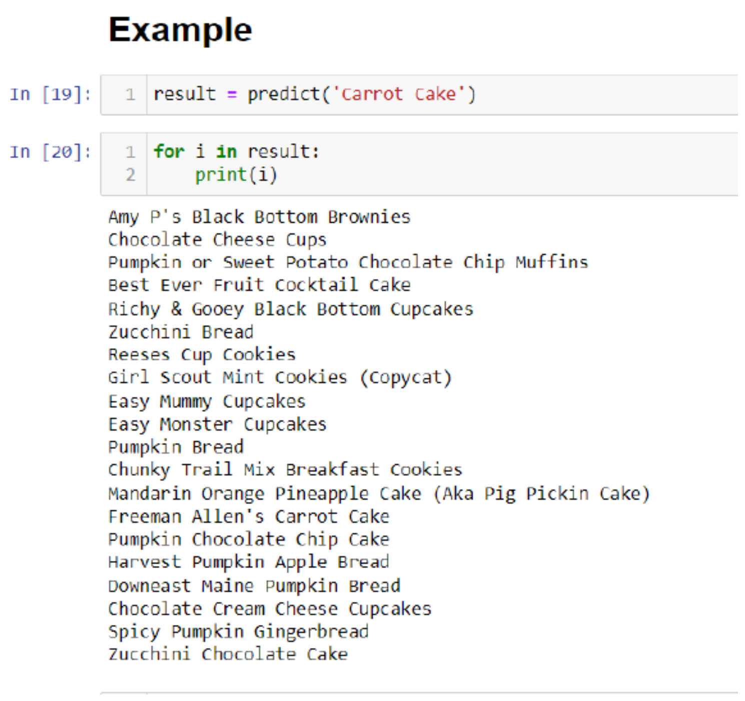
- **Negative Correlation :** A negative correlation is a value between zero (0.0) and minus one (-1.0). A negative correlation shows that the two variables have opposite behaviour. Thus, if the first one moves up, the second one moves down.

- **Zero or no correlation :** A zero correlation shows no relationship between the first variable and the second one. If the first one moves up, the second one may do anything else.

The process we followed to create this recommendation system is:

1. From the original dataset, we keep only the name of the recipes and the nutrition values, which are:
    - FatContent
    - SaturatedFatContent
    - CholesterolContent
    - SodiumContent
    - CarbohydrateContent
    - FiberContent
    - SugarContent
    - ProteinContent
2. A matrix is created with ‘ pivot_table() ’ from the ‘pandas’ library. That creates the spreadsheet-style pivot table as a DataFrame, with the name as a column and the others as values.
3. We create a function that takes as input: the name of the recipe we want to predict similar ones, the correlation lower limit that we want between the recipe and the recommended recipes, and the ‘corrwith()’ function from ‘pandas’ library to compute the correlation between the recipes.
4. We create a new DataFrame with the recipes and the correlations and return a sorted list with the names of the recipes that are acceptable to recommend to the user.

The Figure 4.48 below is an example of this recommendation system. It represents the call of the function mentioned above with the name of the recipe we want to find similar ones. Then, the names of the recipes that are going to be recommended are shown.



Example

```
In [19]: result = predict('Carrot Cake')
```

```
In [20]: for i in result:
             print(i)
```

```
Amy P's Black Bottom Brownies
Chocolate Cheese Cups
Pumpkin or Sweet Potato Chocolate Chip Muffins
Best Ever Fruit Cocktail Cake
Richy & Gooey Black Bottom Cupcakes
Zucchini Bread
Reeses Cup Cookies
Girl Scout Mint Cookies (Copycat)
Easy Mummy Cupcakes
Easy Monster Cupcakes
Pumpkin Bread
Chunky Trail Mix Breakfast Cookies
Mandarin Orange Pineapple Cake (Aka Pig Pickin Cake)
Freeman Allen's Carrot Cake
Pumpkin Chocolate Chip Cake
Harvest Pumpkin Apple Bread
Downeast Maine Pumpkin Bread
Chocolate Cream Cheese Cupcakes
Spicy Pumpkin Gingerbread
Zucchini Chocolate Cake
```

Figure 4.48: Content-Based I example

4.4.2 Content-Based System II

The second Content-Based system uses Cosine Similarity to compute the similarity between recipes.

The process here is different from the process we followed on the previous Content-Based system:

1. We clean the following columns of the 'recipes' DataFrame:
   - **RecipeCategory :** Some special characters are removed like '<, &, /', and the numbers are replaced by their written form. Subsequently, we return a list of the category.
   - **RecipeIngredientParts :** The ingredients are split and returned in a list.
   - **Name :** The recipe's name is returned in a list.
   - **Keywords :** Each recipe's keywords are cleaned like the RecipeCategory, and after being split, we return them in a list.
2. A function that creates a mix of these lists is created and applied to the 'Name' column of the recipes DataFrame, creating a new column named 'total'.

3. We create another new column named 'total parsed' that contains the 'total' data without the commas.

4. Because this procedure requires some time, we save the DataFrame we created in a CSV file, and we read the file every time a recommendation has to be generated by the system. So, when the saved CSV file is read, 'CountVectorizer' from the 'sklearn' library is used to convert the collection of text documents to a matrix of token counts.

5. At this point, 'cosine similarity', also from the 'sklearn' library, is used to compute the cosine similarity between samples in X and Y.

According to [29], the cosine similarity of two non-zero vectors can be computed with the help of the Euclidean dot product formula:

$$A \cdot B = ||A|| \cdot ||B|| \cdot cos(\theta)$$

Consider two vectors of attributes, A and B, the cosine similarity, $cos(\theta)$, is expressed using a dot product and magnitude as in the following figure:

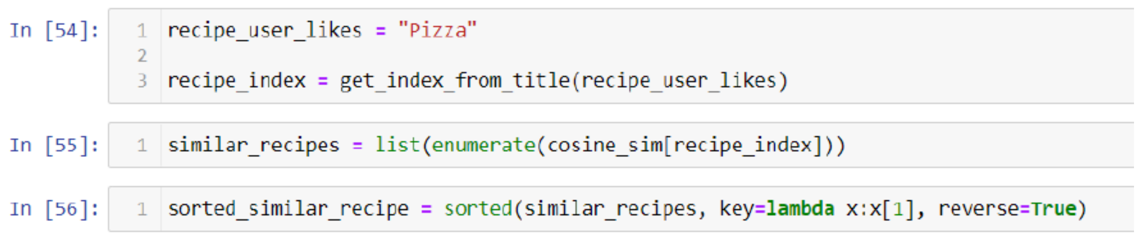
$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 4.49: Cosine Similarity Equation

where A_i and B_i are components of vector A and B respectively.

So, the values of the results from the equation among the characteristics of the two recipes are between zero (0) and one (1). Therefore, if it is zero (0), there is no similarity between the recipes; on the other hand, if it is one (1), the list of characteristics of the recipes is the same.

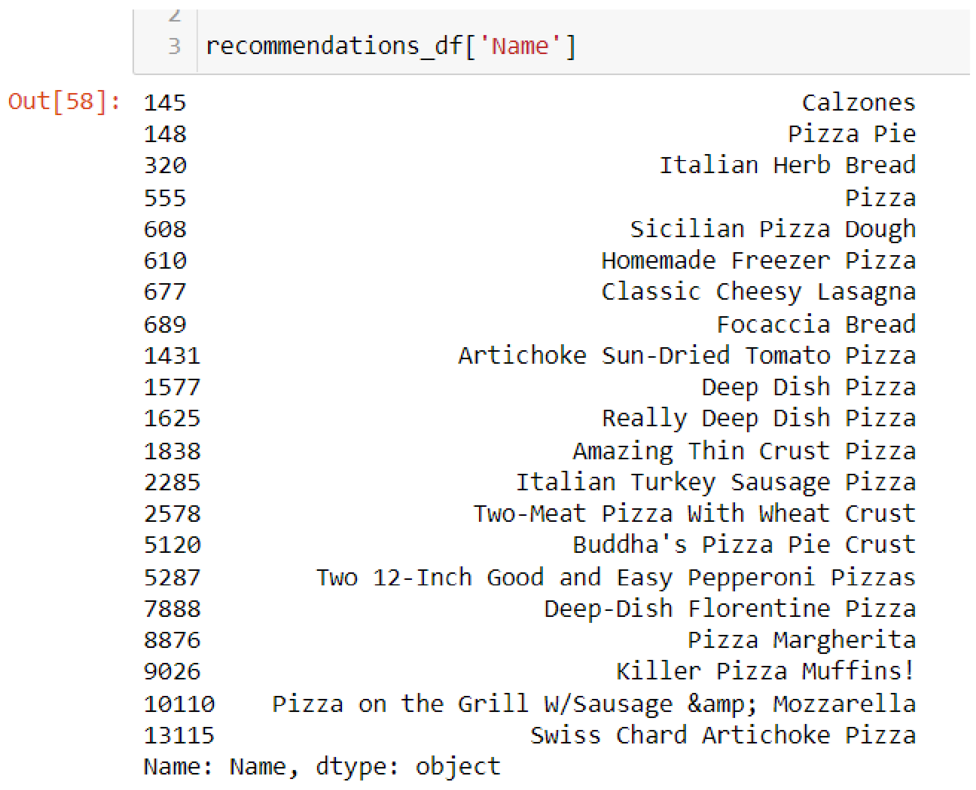
Here, in Figure 4.50 is an example of this recommendation system:



```
In [54]: recipe_user_likes = "Pizza"

         recipe_index = get_index_from_title(recipe_user_likes)

In [55]: similar_recipes = list(enumerate(cosine_sim[recipe_index]))

In [56]: sorted_similar_recipe = sorted(similar_recipes, key=lambda x:x[1], reverse=True)
```

Figure 4.50: Content-Based II example I

The figure above shows the name of the similar recipe we want to find. After that, we find the index of the recipe in the DataFrame and then, using the cosine similarity, we find the similar recipes and sort them in a list. In Figure 4.51, the recommendations are depicted.



```
         recommendations_df['Name']

Out[58]: 145                                      Calzones
         148                                     Pizza Pie
         320                            Italian Herb Bread
         555                                         Pizza
         608                           Sicilian Pizza Dough
         610                          Homemade Freezer Pizza
         677                          Classic Cheesy Lasagna
         689                                 Focaccia Bread
         1431                Artichoke Sun-Dried Tomato Pizza
         1577                               Deep Dish Pizza
         1625                        Really Deep Dish Pizza
         1838                       Amazing Thin Crust Pizza
         2285                      Italian Turkey Sausage Pizza
         2578                   Two-Meat Pizza With Wheat Crust
         5120                        Buddha's Pizza Pie Crust
         5287          Two 12-Inch Good and Easy Pepperoni Pizzas
         7888                     Deep-Dish Florentine Pizza
         8876                              Pizza Margherita
         9026                             Killer Pizza Muffins!
         10110     Pizza on the Grill W/Sausage &amp; Mozzarella
         13115                     Swiss Chard Artichoke Pizza
         Name: Name, dtype: object
```

Figure 4.51: Content-Based II example II

4.4.3 Collaborative-Filtering System

This recommendation system is considerably different from the other ones. The steps to create it are the following:

1. From the 'reviews' dataset, we kept only the following columns:

   - AuthorId

- RecipeId
- Rating

The first one, 'AuthorId', is renamed as 'UserId'.

2. Apart from this dataset ('reviews'), we take the ratings of our users from our database and add them at the end of the previous DataFrame. That is because the target is to find similar users to the current user, according to the recipes he has rated. Hence, we need the ratings of the users we create on our website to gain the ability to recommend recipes to them.

3. The dimension of the DataFrame has to be reduced to avoid any memory errors. So, we keep only recipes with more than twenty (20) recipes and users that have rated more than one (1) recipe. Naturally, the system is more accurate if the users have rated more recipes, but we wanted to allow new users to have recommendations.

4. Then, we defined from the 'surprise' library, 'BaselineOnly' algorithm and fitted it to our data with some options we will mention after the steps.

5. We created a function named 'recipes_to_predict' and returns the recipes that the target user has not rated yet so that the recommendations do not have already known recipes for the user.

6. Another function is created, named 'get_recommendations' that takes as input the 'UserId' of the target user, the DataFrame we made above with the 'UserId', 'RecipeId' and 'Rating', the 'recipes' dataset and a threshold. The basic idea of this algorithm is that it predicts the rating a user would give to a particular recipe. The threshold shows the lower limit of the rating, meaning that the recipes with a predicted rating more or equal to the threshold value will be recommended to the target user. So, this function estimates the rating values and returns the DataFrame with the most accurate recipes.

From [30] the algorithm predicting the baseline estimate for a given user and item is computed as follows:

$$r_{ui} = b_{ui} = \mu + b_u + b_i$$

If user u is unknown, then the bias b_u is assumed to be zero. The same applies to item ‘i’ with b_i.

According to [31] the algorithm we used tries to minimize the following regularized squared error:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - (\mu + b_u + b_i))^2 + \lambda(b_u^2 + b_i^2)$$

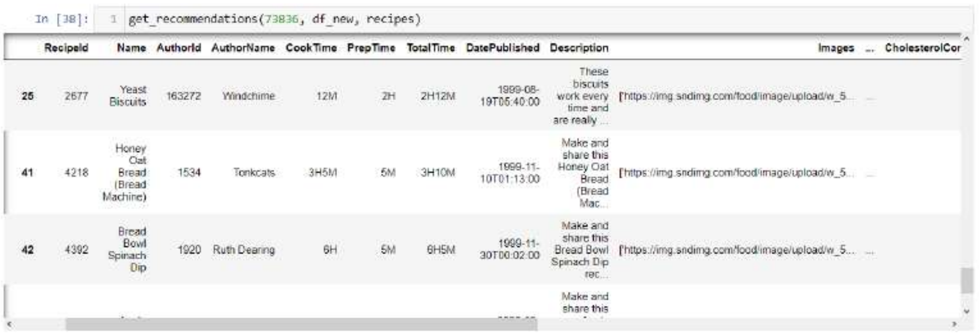
Baselines can be estimated in two ways:

- with Stochastic Gradient Descent (SGD)
- with Alternative Least Squares (ALS)

In our case, using the ‘bls_options’ parameter, we chose:

- ALS to compute baselines
- ‘reg_i’ value is five (5), which is the regularization parameter for items
- ‘reg_u’ value is twelve (12), which is the regularization parameter for users
- ‘n_epochs’ value is five (5) that is the number of iterations of the ALS procedure

Below, there is Figure 4.52 that shows the DataFrame with the predicted recipes of a random ‘UserId’:



```
In [38]:  1 get_recommendations(73836, df_new, recipes)
```

	RecipeId	Name	AuthorId	AuthorName	CookTime	PrepTime	TotalTime	DatePublished	Description	Images	...	CholesterolCor
25	2677	Yeast Biscuits	163272	Windchime	12M	2H	2H12M	1999-08-19T05:40:00	These biscuits work every time and are really ...	['https://img.sndimg.com/food/image/upload/w_5...	...	
41	4218	Honey Oat Bread (Bread Machine)	1534	Tonkcats	3H5M	5M	3H10M	1999-11-10T01:13:00	Make and share this Honey Oat Bread (Bread Mac...	['https://img.sndimg.com/food/image/upload/w_5...	...	
42	4392	Bread Bowl Spinach Dip	1920	Ruth Dearing	6H	5M	6H5M	1999-11-30T00:02:00	Make and share this Bread Bowl Spinach Dip rec...	['https://img.sndimg.com/food/image/upload/w_5...	...	
									Make and share this			

Figure 4.52: Collaborative-Filtering example

4.4.4 Hybrid System

As mentioned before, 4.4.2 and 4.4.3 are combined to give results on the recommendations page of our website. For example, consider user A that has his list of favourite recipes and can rate any recipe he likes.

1. **First case :** User A has not rated any recipe and does not have any favourite recipe.

   In the 4.4.3 section, we mentioned that for the Collaborative-Filtering system, we keep only the users that have rated at least one recipe for the system to generate recommendations. This makes sense because the system can not find similar users as user A has not rated any recipes. Apart from that, in 4.4.2 section finds similar recipes from the list of favourites in the Content-Based system. So, the two systems can not generate recommendations, and the shown recipes are the top-rated recipes based on the number of ratings and the values of the ratings.

2. **Second case :** User A has at least one recipe on his list of favourites but has not rated any recipes.

   In this case, only the 4.4.3 Collaborative-Filtering system can not make any recommendations. Basically, from the 4.4.2 Content-Based system, we get the twenty (20) most similar recipes from each one of the favourite recipes of User A and append them to a list. Then, we use a random function to return twenty (20) recipes to recommend to User A.

3. **Third case :** User A has rated at least one recipe but does not have any recipes on his list of favourites.

   Here, the 4.4.2 Content-Based system cannot make suggestions. So, the 4.4.3 Collaborative-Filtering system finds the recipes that the user would rate highly and lists them. After that, using the random function again, we return twenty (20) recipes to recommend to User A.

4. **Fourth case :** User A has rated at least one recipe and has at least one recipe on his favourites list.

   In this last case, both of the recommendation systems can generate recommendations. So, same as the above, a list of recommended recipes is created, and with the random function, we return only twenty (20) of them to User A as recommendations.

Chapter 5

Experimental Evaluation

In this chapter, the focus point is evaluating our Collaborative-Filtering recommendation system. Apart from the algorithm we used, the whole library 'surprise' is explained in detail. Also, we use figures to display the evaluation results with the corresponding values (i.e. error values).

5.1 Recommendation Algorithm Evaluation

The Collaborative-Filtering system mentioned in the previous chapter is implemented using an algorithm named 'BaselineOnly' from the 'surprise' library. This library consists of multiple algorithms such as: [32]

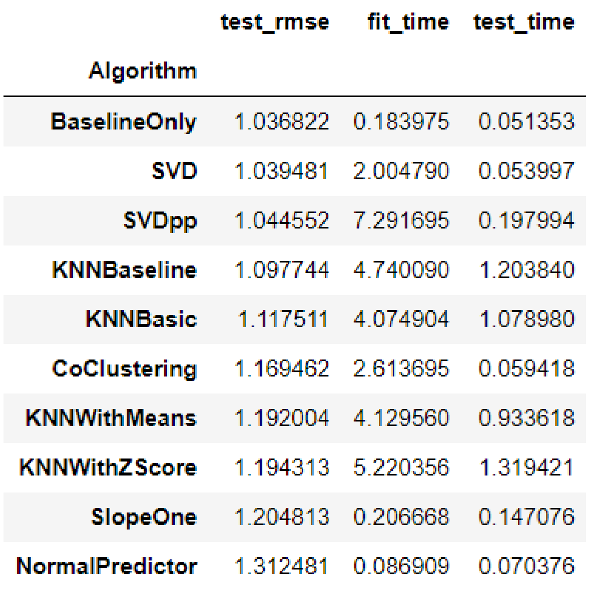
- **SVD :** It is a Matrix-Factorization algorithm, equivalent to Probabilistic Matrix Factorization. [33]
- **SVDpp :** This is the same as the above algorithm with the characteristic that it takes into account the implicit ratings of the users.
- **NMF :** It is a collaborative filtering algorithm based on Non-negative Matrix Factorization. It is identical to SVD.
- **KNNBasic :** It is a fundamental collaborative filtering algorithm.
- **KNNBaseline :** This is a main collaborative filtering algorithm that considers a baseline rating.

- **KNNWithZScore :** It is also a main collaborative filtering algorithm that considers the z-score normalization of each user.
- **KNNWithMeans :** This is also a fundamental collaborative filtering algorithm that considers each user's mean ratings.
- **NormalPredictor :** This is an algorithm that predicts a random rating based on the distribution of the training set, which is supposed to be normal. It is one of the main algorithms that do not do much work.
- **BaselineOnly :** It is an algorithm that predicts the baseline estimate for a given user and item.
- **CoClustering :** It is an algorithm based on Co-Clustering [34]
- **SlopeOne :** This is a straightforward implementation of the SlopeOne algorithm. [35]

Moreover, the 'surprise' library offers the possibility to use cross-validation to find which is the best algorithm to use based on the data the developer has. It tests and trains the dataset taking as input on different iterations. It creates the following sets: [36]

- **training set :** is the part of the data that is known and used to train the algorithm.
- **testing set :** The rest of the dataset that is not in the training set is in this set and is the 'unknown' data that we test the model with.

We used the cross-validation from surprise with all the possible algorithms and RMSE as a measure to find the best algorithm to use to predict our recommendations. In the Figure 5.1, there are the results for our dataset:



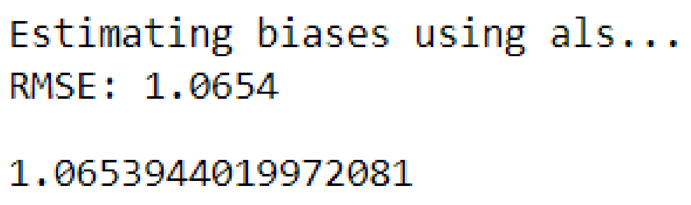
Algorithm	test_rmse	fit_time	test_time
BaselineOnly	1.036822	0.183975	0.051353
SVD	1.039481	2.004790	0.053997
SVDpp	1.044552	7.291695	0.197994
KNNBaseline	1.097744	4.740090	1.203840
KNNBasic	1.117511	4.074904	1.078980
CoClustering	1.169462	2.613695	0.059418
KNNWithMeans	1.192004	4.129560	0.933618
KNNWithZScore	1.194313	5.220356	1.319421
SlopeOne	1.204813	0.206668	0.147076
NormalPredictor	1.312481	0.086909	0.070376

Figure 5.1: Cross Validation results

The function returns:

- **test_rmse :** An array with accuracy values for each test set.
- **fit_time :** An array with the time of training in seconds for each split.
- **test_time :** An array with the testing time in seconds for each split.

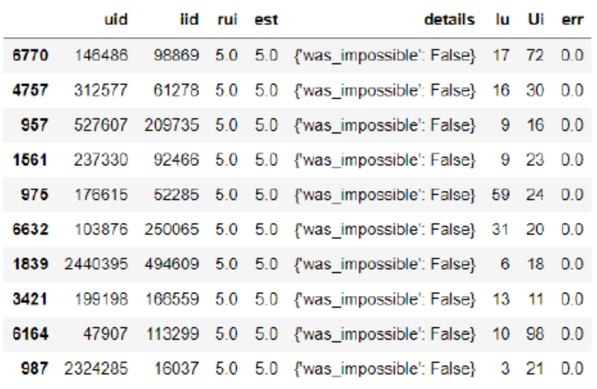
Baseline gave us the best RMSE; therefore, we will train and predict with BaselineOnly and use Alternative Least Squares(ALS). We train the algorithm with our data, and below, in Figure 5.2, there is the conclusive accuracy of the algorithm measured with the RMSE value:



```
Estimating biases using als...
RMSE: 1.0654

1.0653944019972081
```

Figure 5.2: Accuracy of BaselineOnly algorithm

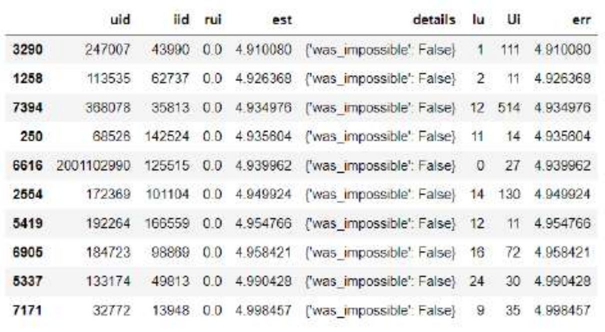
After that, we generated two DataFrames that show the best and the worst predictions. It considers the value of the error, which is the difference between the estimated rating of the algorithm and the actual value of the rating. Below there are the two Figures, 5.3 and 5.4, for the best and the worst predictions, respectively:



	uid	iid	rui	est	details	Iu	Ui	err
6770	146486	98869	5.0	5.0	{'was_impossible': False}	17	72	0.0
4757	312577	61278	5.0	5.0	{'was_impossible': False}	16	30	0.0
957	527607	209735	5.0	5.0	{'was_impossible': False}	9	16	0.0
1561	237330	92466	5.0	5.0	{'was_impossible': False}	9	23	0.0
975	176615	52285	5.0	5.0	{'was_impossible': False}	59	24	0.0
6632	103876	250065	5.0	5.0	{'was_impossible': False}	31	20	0.0
1839	2440395	494609	5.0	5.0	{'was_impossible': False}	6	18	0.0
3421	199198	166559	5.0	5.0	{'was_impossible': False}	13	11	0.0
6164	47907	113299	5.0	5.0	{'was_impossible': False}	10	98	0.0
987	2324285	16037	5.0	5.0	{'was_impossible': False}	3	21	0.0

Figure 5.3: Best predictions of BaselineOnly algorithm

The above are the best predictions, and they are not lucky guesses. However, because Ui (the number of users that have rated the recipe) is between 7 to 35, they are not really small, meaning that a significant number of users have rated the target recipe.



	uid	iid	rui	est	details	Iu	Ui	err
3290	247007	43990	0.0	4.910080	{'was_impossible': False}	1	111	4.910080
1258	113535	62737	0.0	4.926369	{'was_impossible': False}	2	11	4.926369
7394	368078	35813	0.0	4.934976	{'was_impossible': False}	12	514	4.934976
250	68526	142524	0.0	4.935604	{'was_impossible': False}	11	14	4.935604
6616	2001102990	125515	0.0	4.939962	{'was_impossible': False}	0	27	4.939962
2554	172369	101104	0.0	4.949924	{'was_impossible': False}	14	130	4.949924
5419	192264	166559	0.0	4.954766	{'was_impossible': False}	12	11	4.954766
6905	184723	98869	0.0	4.958421	{'was_impossible': False}	16	72	4.958421
5337	133174	49813	0.0	4.990428	{'was_impossible': False}	24	30	4.990428
7171	32772	13948	0.0	4.998457	{'was_impossible': False}	9	35	4.998457

Figure 5.4: Worst predictions of BaselineOnly algorithm

The worst predictions have a significant error value, so although the estimated rating is high, the real one is relatively low.

Chapter 6

Conclusions

In this chapter, we intend to summarise what we have mentioned in all the previous chapters. Apart from that, we highlight a few ideas we would like to implement or improve in the future regarding our project.

6.1 Summary

This diploma thesis aimed to create a website for recommending recipes to users from scratch. The application we developed not only achieves our target but also provides the user with relevant information about each recipe, other user experiences with the recipe and even photographic documentation of it to better visualise the result, among other features.

We developed three different recommendation systems to give the user more flexibility and control over the recommended recipes. We tested multiple algorithms to figure out the methods that work best for our dataset and selected the one with the best results for each recommendation system. The first, a content-based system, utilises the correlation between items to suggest recipes with similar nutritional values. The second, another content-based system, uses cosine similarity to calculate the similarity between recipes. The third, a collaborative filtering system, finds the users most similar to our user to produce recommendations. To make sure we deliver the best results, in any case, we took into account the outliers (i.e. when the user does not have any recipes in the favourites or they have too many of them) and used the second and third systems to build a hybrid one. This hybrid recommendation system has individual systems to produce recommendations, and only the best ones make it to the final user recommendations.

In Chapter 5, we present the results of our recommendations after an extensive evaluation process where we can see the low value of the produced RMSE. According to those results, we are content with the progress of our project, as, in our allotted time, we have managed to integrate multiple distinct operations into the website.

6.2 Future Work

There are several things to develop further for future work:

- Use an actual user database instead of one out of a CSV file. For our website to work correctly, we needed a large amount of data that we could only get through a CSV file or an API. We would like, in the future, we could run the website without the use of the CSV file.
- Fine-tune the recommendation systems (possibly turn them into a hybrid). Presently, the three recommendation systems on our website are content-based and collaborative filtering. We aim to turn them into hybrid recommendation systems, as they perform best in evaluation scores.
- List of recipes the user did not like. Explicit feedback is always necessary for recommendation systems, and a list with content the user does not like will help us generate more suitable recommendations.
- Develop the ‘Add a recipe’ option for the user. Presently, the user cannot upload their recipes and is only able to view the available recipes from the CSV file. In the future, we would like to implement the feature.

Bibliography

[1] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer, 2016.

[2] Kim Falk. Practical recommender systems, Jan 2019.

[3] Robin van Meteren and Maarten van Someren. Using content-based filtering for recommendation, 2000.

[4] CHRISTINA CHRISTAKOU, SPYROS VRETTOS, and ANDREAS STAFYLOPATIS. A hybrid movie recommender system based on neural networks. *International Journal on Artificial Intelligence Tools*, 16(05):771–792, 2007.

[5] Robin Burke. Hybrid recommender systems: Survey and experiments - user modeling and user-adapted interaction, Nov 2002.

[6] Tulasi K. Paradarami, Nathaniel D. Bastian, and Jennifer L. Wightman. A hybrid recommender system using artificial neural networks. *Expert Systems with Applications*, 83:300–313, 2017.

[7] SongJie Gong, HongWu Ye, and HengSong Tan. Combining memory-based and model-based collaborative filtering in recommender system. In *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, pages 690–693, 2009.

[8] Lipi Shah, Hetal Gaudani, and Prem Balani. Survey on recommendation system. *International Journal of Computer Applications*, 137(7):43–49, Mar 2016.

[9] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Jesús Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012.

[10] Al Rashid, Istvan Albert, Dan Cosley, Shyong Lam, Sean McNee, Joseph Konstan, and John Riedl. Getting to know you: Learning new user preferences in recommender

systems. *International Conference on Intelligent User Interfaces, Proceedings IUI*, 02 2002.

[11] Sarah Bouraga, Ivan Jureta, Stéphane Faulkner, and Caroline Herssens. Knowledge-based recommendation systems. *International Journal of Intelligent Information Technologies*, 10(2):1–19, 2014.

[12] Ladislav Peska. Using the context of user feedback in recommender systems - arxiv, 2016.

[13] Douglas W. Oard and Jinmook Kim. Implicit feedback for recommender systems, 1998.

[14] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, jan 2004.

[15] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 257–260, New York, NY, USA, 2010. Association for Computing Machinery.

[16] Félix Hernández del Olmo and Elena Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35(3):790–804, 2008.

[17] Guy Shani and Asela Gunawardana. Evaluating recommendation systems, Oct 2010.

[18] Rocío Cañamares, Pablo Castells, and Alistair Moffat. Offline evaluation options for recommender systems. *Information Retrieval Journal*, 23(4):387–410, 2020.

[19] Suyash Maheshwari and Manas Chourey. Recipe recommendation system using machine learning models. *International Research Journal of Engineering and Technology (IRJET)*, 2019.

[20] Tsuguya Ueta, Masashi Iwakami, and Takayuki Ito. Implementation of a goal-oriented recipe recommendation system providing nutrition information. In *2011 International Conference on Technologies and Applications of Artificial Intelligence*, pages 183–188, 2011.

[21] Hetal Gaudani. Personalized recipe recommendation system using hybrid approach. *International Journal of Innovative Research in Computer and Communication Engineering*, 5:192–197, 06 2016.

[22] Zhengxian Li, Jinlong Hu, Jiazhao Shen, and Yong Xu. A scalable recipe recommendation system for mobile application. *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, 2016.

[23] djangoproject.com, 2022. Available at `https://www.djangoproject.com/start/overview/`.

[24] Mark Otto and Bootstrap contributors. Bootstrap, 2022. Available at `https://getbootstrap.com/docs/5.1/getting-started/introduction/`.

[25] Anaconda: the world's most popular data science platform, 2022. Available at `https://www.anaconda.com`.

[26] Postgresql, 2022. Available at `https://www.postgresql.org`.

[27] pgadmin - postgresql tools. Available at `https://www.pgadmin.org`.

[28] Kaggle: Food.com - recipes and reviews. Available at `https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews`.

[29] Wikipedia - cosine similarity. Available at `https://en.wikipedia.org/wiki/Cosine_similarity`.

[30] Baselineonly-surprise. Available at `https://surprise.readthedocs.io/en/stable/basic_algorithms.html#surprise.prediction_algorithms.baseline_only.BaselineOnly`.

[31] Baseline estimates configuration-surprise. Available at `https://surprise.readthedocs.io/en/stable/prediction_algorithms.html#baseline-estimates-configuration`.

[32] Surprise - prediction algorithms. Available at `https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html`.

[33] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1257–1264, Red Hook, NY, USA, 2007. Curran Associates Inc.

[34] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–, 2005.

[35] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, 5, 02 2007.

[36] Daniel Berrar. *Cross-Validation.* Data Science Laboratory, Tokyo Institute of Technology, 01 2018.

[37] The hitchhiker's guide to python! Available at `https://docs.python-guide.org`.

[38] Python.org. Available at `https://www.python.org/about/`.

[39] Pipenv and virtual environments. Available at `https://docs.python-guide.org/dev/virtualenvs/#lower-level-virtualenv`.

[40] How to get django. Available at `https://www.djangoproject.com/download/`.

[41] How to install django on windows. Available at `https://docs.djangoproject.com/en/4.1/howto/windows/`.

[42] Postgresql downloads. Available at `https://www.postgresql.org/download/`.

[43] pgadmin - getting started. Available at `https://www.pgadmin.org/docs/pgadmin4/development/getting_started.html`.

[44] Evangelia-Alkistis Lemonaki and Kleopatra Beka. Github code, 2022. Available at `https://github.com/kbeka/thesis-snackify/tree/main`.

[45] Postgresql tutorial - import csv file into postgresql table. Available at `https://www.postgresqltutorial.com/postgresql-tutorial/import-csv-file-into-posgresql-table/`.

APPENDICES

Appendix A

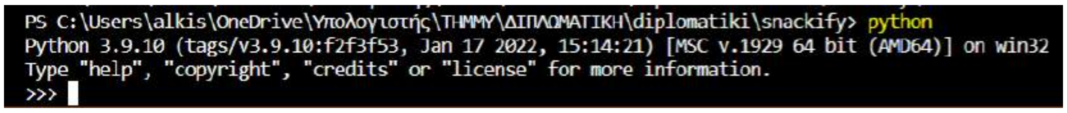
Prerequisites

This appendix is a step-by-step guide on how to download everything needed in order to make the application run correctly.

A.1 Download Python

To run the project, it is necessary to install python 3.x on a personal computer. Python is available for Windows, MacOS, Linux/UNIX and other systems. There are detailed instructions on how to download and install python in [37], and if no instructions are needed, python can be downloaded directly from [38].

Check the python version on a personal computer (PC) with the 'python' command on the terminal like the following figure:



```
PS C:\Users\alkis\OneDrive\Υπολογιστής\ΤΗΜΜΥ\ΔΙΠΛΩΜΑΤΙΚΗ\diplomatiki\snackify> python
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure A.1: Check python version

However, it is easier to download python and other packages if Anaconda [25] is installed first.

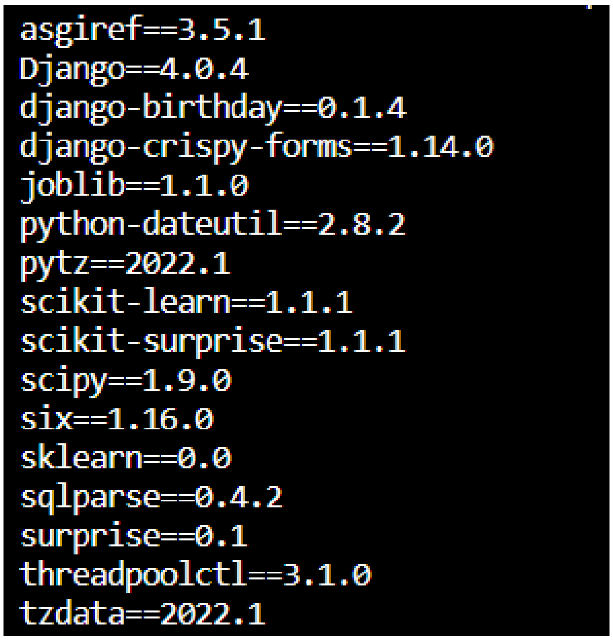
A.2 Create Virtual Environment

To implement our project, we used a virtual environment. The main advantage of using it is to isolate the project and download all the packages needed without changing anything in any other project outside the environment. To create a virtual environment, follow the steps from [39] or if the instructions from [37] are followed, the virtual environment should already be installed.

In our project, we have a virtual environment named ‘diplomatiki’. To create this, you should be inside the project folder, and after creating it, the environment has to be activated according to the instructions mentioned above.

A.3 Download Packages

The packages downloaded to run the project are the following:



```
asgiref==3.5.1
Django==4.0.4
django-birthday==0.1.4
django-crispy-forms==1.14.0
joblib==1.1.0
python-dateutil==2.8.2
pytz==2022.1
scikit-learn==1.1.1
scikit-surprise==1.1.1
scipy==1.9.0
six==1.16.0
sklearn==0.0
sqlparse==0.4.2
surprise==0.1
threadpoolctl==3.1.0
tzdata==2022.1
```

Figure A.2: Downloaded packages

To install the packages above there are two options:

- **Anaconda users :**

  1. > conda install pip

  2. > pip install ‘PackageName’

- **Other users :** > pip install ‘PackageName’

A.4 Download Django

Most Linux and Unix versions have already installed Django. However, if it is not installed, there is a detailed tutorial here [40] for Linux. Moreover, below there is a guide here [41] on how to download and install Django on Windows operating systems.

A.4.0.1 How to create a project

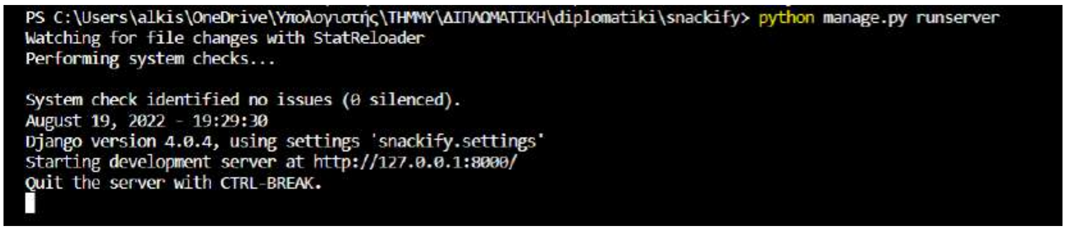
To create a project, it is necessary to activate the virtual environment. After it is activated, the following command must be given:

- > djangoadmin.py startproject ProjectName

Subsequently, to run the server on your personal computer, use the following command:

- > python manage.py runserver

If there are no errors, the result looks like this:



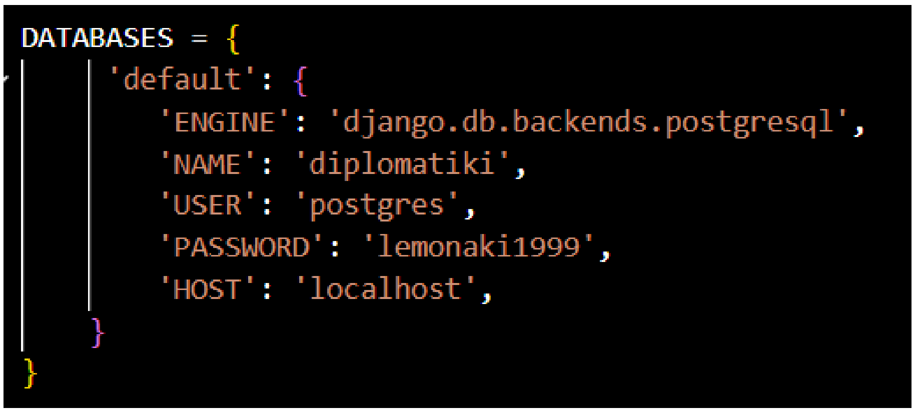
```
PS C:\Users\alkis\OneDrive\Υπολογιστής\ΤΗΜΜΥ\ΔΙΠΛΩΜΑΤΙΚΗ\diplomatiki\snackify> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 19, 2022 - 19:29:30
Django version 4.0.4, using settings 'snackify.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure A.3: Django run server

Finally, by visiting ‘http://127.0.0.1:8000/’ in a browser, the result of the code appears.

A.5 Download pgAdmin4/PostgreSQL

PostgreSQL and pgadmin4 are downloaded together from here [42]. To fully understand how pgadmin4 works, the documentation is here [43]. After the installation, a database has to be created in pgadmin, named ‘diplomatiki’. The username and password will be used in the project code to connect the database with the project. Specifically, in the file named ‘settings.py’, the details of the database are shown in the following figure:



```
DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.postgresql',
            'NAME': 'diplomatiki',
            'USER': 'postgres',
            'PASSWORD': 'lemonaki1999',
            'HOST': 'localhost',
        }
}
```

Figure A.4: Connection of database

The following fields have to be set:

- **USER :** the username you created the database with.
- **PASSWORD :** use the password you created the database with.
- **HOST :** use 'localhost' if it is installed in your personal computer.
- **PORT :** the default port is 5432 and does not have to be written down, but if it is not the default it has to be mentioned

Appendix B

How to use the website

To run the project's code, it is necessary to follow the instructions of the Appendix A and download and install the required tools and packages.

B.1 Download the code

You can download the project's code directly from [44] in zip form. On the website page, by clicking the 'Code' button, there is a 'Download ZIP' button.

B.2 Set up the database

To begin with, we use the following commands to create the tables in the database based on the models of the project:

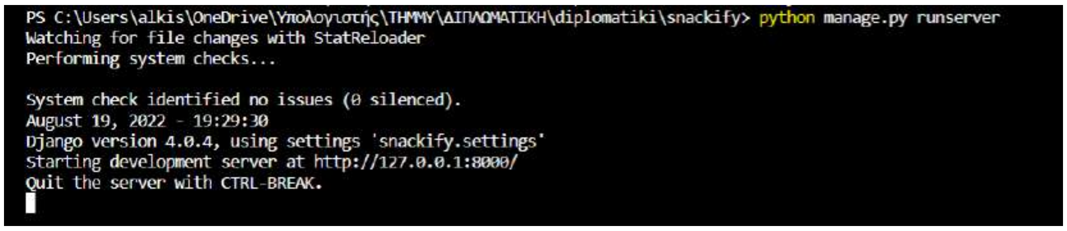
- > python manage.py makemigrations
- > python manage.py migrate

Subsequently, you have to fill the 'recipes_info' and 'reviews' database with the corresponding files from here [28]. In addition, from here [44] you can download the preprocessing of the CSV files from [28] to fill the database with the preprocessed data. Finally, there is an option in pgadmin to fill a table from a CSV file [45], and that is how you will set up the database.

At this point, you can run the server on your personal computer, as mentioned in Appendix A, with the following command:

- > python manage.py runserver

If everything runs smoothly, the result is the following:



```
PS C:\Users\alkis\OneDrive\Υπολογιστής\ΤΗΜΜΥ\ΔΙΠΛΩΜΑΤΙΚΗ\diplomatiki\snackify> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 19, 2022 - 19:29:30
Django version 4.0.4, using settings 'snackify.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure B.1: Project run server